CLEI-2015

2015 XLI Latin American Computing Conference (CLEI), Special Edition
http://clei.org/clei2015

October 19th-23rd, 2015, Arequipa, Peru
Hosted by: SPC, UNSA, UCSM, UCSP, ULS, Arequipa, Peru

Editors:
Hector Cancela
Alex Cuadros-Vargas
Ernesto Cuadros-Vargas

Proceedings

2015 XLI Latin American Computing Conference (CLEI), Special Edition
http://clei.org/clei2015

CLEI-2015

Arequipa, October 19th-23rd, 2015

Local Organizers:

- Sociedad Peruana de Computación
- Universidad Nacional de San Agustín, Arequipa
- Universidad Católica de Santa María, Arequipa
- Universidad Católica San Pablo, Arequipa
- Universidad La Salle, Arequipa
- Universidad Nacional de Ingeniería, Lima
- Universidad Nacional Mayor de San Marcos, Lima
- Universidad Ricardo Palma, Lima
- Universidad Nacional San Cristóbal de Huamanga, Ayacucho
- Universidad Nacional del Altiplano, Puno
- Universidad Nacional Micaela Bastidas, Abancay
- Asociación Peruana de Productores de Software (APESOFT)
- EDDAS, Arequipa
- IEEE Sección Perú
- ACM UCSP Chapter

Silver Supporters:

- Oracle Academy

Abstracts of the 2015 XLI Latin American Computing Conference (CLEI), Special Edition
Arequipa, Peru, October 19th-23rd

2015 XLI Latin American Computing Conference (CLEI), Special Edition
Organizing Committee

Message from Conference and General Program Chairs

CLEI (Centro Latinoamericano de Estudios en Informática) is an organization composed of aproximatly ninety institutions granting Computing degrees from all countries in the Latin American region.

CLEI has promoted annual meetings on Computing in the Latin American region for more than four decades. These meetings promote the exchange of knowledge and progress in research on different fields of Computing. The CLEI events have a high impact in the Latin American region, and also in other parts of the world. They have consolidated a number of symposia in different domains of computing, and have created others in new research areas.

Throughout the years, the successful development of each meeting has enabled the CLEI organization to consolidate a prestigious trajectory in the Computer Science community in Latin America as well as in other regions.

This is the second time Peru hosts the CLEI annual meeting. The first time was in 2004, both in Arequipa. The present edition is a very special one, as it marks the 41st CLEI conference, and it is co-organized by more that 10 Peruvian institutions: ▪ Sociedad Peruana de Computación ▪ Universidad Nacional de San Agustín (Arequipa) ▪ Universidad Católica de Santa María (Arequipa) ▪ Universidad Católica San Pablo (Arequipa) ▪ Universidad La Salle (Arequipa) ▪ Universidad Nacional de Ingeniería (Lima) ▪ Universidad Nacional Mayor de San Marcos (Lima) ▪ Universidad Ricardo Palma (Lima) ▪ Universidad Nacional San Cristóbal de Huamanga (Ayacucho) ▪ Universidad Nacional del Altiplano (Puno) ▪ Universidad Nacional Micaela Bastidas (Abancay) ▪ Asociación Peruana de Productores de Software (APESOFT) ▪ EDDAS (Arequipa) ▪ IEEE Peru Section and ▪ ACM UCSP Chapter showing the cooperation and strenght of the Peruvian academic community.

For the last several years, the Latin American Computing Conference has been organized in a number of symposia: 1) Latin American Symposium on Computer Graphics, Virtual Reality, and Image Processing, 2) Latin American Symposium on Software Engineering, 3) Latin American Symposium on Informatics and Society, 4) Latin American Symposium on Operational Research and Artificial Intelligence, and 5) Latin American Symposium on Infrastructure, Hardware, and Software, 6) Latin American Symposium on Large Scale Information Systems, 7) Latin American Symposium on Data and Information Management, 8) Latin American Symposium on Theoretical Computer Science. 9) XXII Iberoamerican Symposium on Higher Education in Computing (SIESC 2015)

Additionally, CLEI 2015 includes the following associated events: 1) XXI Latin American Master Thesis Competition(CLTM 2015) 2) I Latin American PhD Thesis (CLTD 2015) 3) The VI Latin American Women in Computing Congress (LAWCC 2015) 4) The IV Workshop on Nomenclature and Accreditation of Computing Programs

Over 400 contributions were received from different parts of America and Europe. All contributions were subject to a rigorous review process, resulting in 120 accepted papers. The main proceedings of the conference are included and indexed in IEEE Xplore; at the same time, the best papers presented at CLEI 2015 symposia and SIESC 2015 were invited to be published in the Electronic Notes in Theoretical Computer Science (ENTCS) journal and the CLEI Electronic Journal (CLEIej).

CLEI 2015 is sponsored by EDDAS, APESOFT and Oracle Academy. A conference like CLEI is only possible by the involvement of a large community; we want to thank the CLEI steering committee, members of the program committees and the organizing committee, invited lectures and speakers, authors, teachers, students, administrative staff, and many other people who contributed towards the success of the event.

Héctor Cancela
Universidad de la República (Uruguay)
Chair of the Program Committee

Alex Cuadros-Vargas
Universidad Católica San Pablo (Peru)
Chair of the Program Committee

Ernesto Cuadros-Vargas
Universidad Católica San Pablo (Peru)
Chair of the Organizing Committee

Index

General Index

Next Generation of GPUs

Esteban Clua[1]
[1]Federal Fluminense University. Rio de Janeiro Brazil
email: *esteban@ic.uff.br*

Schedule: Mon 19th@09:00, **Room:** A

GPUs were initially developed for solving real time graphics pipeline. Due to their massively parallel architectures and the creation of a unified architecture, they became an important alternative for high performance computing. Modern GPUs have more than 3000 cores and may achieve up to 7TFlops, which is dozens of times the amount of a CPU based architecture. This talk will briefly show the hardware architecture of a modern GPU, will discuss about trends and the future of the next generation of GPUs and will present how different areas are being enhanced by this technologies, such as video-games, deep-learning and self-driving cars.

Short Biography

Esteban Walter Gonzalez Clua graduated in Computer Science at Universidade de São Paulo and has Master's and PhD degree in Computer Science. He has experience in Computer Science and has been active in the following subjects: game engine architectures, 3D games, GPU Computing, education with games, real time computer graphics and digital entertainment. Today Esteban is associate professor in computer science of Universidade Federal Fluminense, in Rio de Janeiro, and director of UFF Medialab. Esteban is one of the founders of SBGames - Brazilian Symposium of Digital Entertainment and Video Games (the largest conference in the subject in South America), president of the Brazilian Computing Society Game Committee and member of program committees of many conferences in Video Games. In 2007 he received the prize of the personality which most contributed for the growth of the video game industry in Brazil and in 2009 and 2013 received the prize of Young Scientist of the State of Rio de Janeiro. Esteban is the coordinator of the first Latin America NVIdia Center of Excelence, which is in UFF Medialab and is a NVIDIA Fellow professor.

Los procesadores en el final de la ley de Moore

Francisco Tirado[1]
[1]Universidad Complutense. Madrid Spain
email: *ptirado@ucm.es*

Schedule: Mon 19th@14:00, **Room:** A

Desde su aparición, los procesadores han doblado su velocidad cada 18 meses. Esto ha sido debido, por una parte, a los avances en la integración de circuitos que permiten duplicar el número de transistores cada 18 meses y una mayor velocidad del reloj y por otra, a mejoras de su arquitectura (organización interna de los diferentes módulos operativos) orientadas a ejecutar un número cada vez mayor de instrucciones por ciclo. La dedicación de recursos para incrementar el rendimiento de un procesador ha llegado a un punto donde el retorno que se obtiene en rendimiento es pequeño. Hoy en día, factores como la potencia y disipación térmica son factores determinantes en el diseño de los chips. La explotación, por tanto de estos recursos adicionales se ha orientado a la implementación en el chip de múltiples niveles de paralelismo. Además, nos encontramos en el posible fin de la Ley de Moore, y esto plantea problemas adicionales en el escalado tecnológico, dejándose de cumplir la reglas que ha funcionado por más de 50 años

Short Biography

Francisco Tirado es Catedrático de Arquitectura y Tecnología de Computadores en la Universidad Complutense de Madrid desde 1986. Ha trabajado en diferentes áreas de investigación dentro de la arquitectura de sistemas, computación de altas prestaciones y arquitectura del procesador. Hasta la actualidad es coautor de más de 250 publicaciones en revistas y conferencias internacionales. Ha participado en la organización de más de 90 Congresos Internacionales como General Chair, Program Chair, miembro del Comité de Programa, Chair de Sesión, Conferenciante Invitado y revisor de artículos. Ha impartido más 80 conferencias en congresos internacionales y universidades. Es Premio Nacional de Informática 2013 y Doctor Honoris Causa por la Universidades de San Agustín (Perú) y la Nacional de Asunción (Paraguay). Ha ocupado diferentes puestos en organismos de investigación y evaluación. Gestor en el Programa Nacional de I+D, Presidente de la Comisión Nacional de Evaluación de Actividad Investigadora, Director del Parque Científico de Madrid. En la actualidad es Vicerrector de Investigación de la UCM, Presidente de Honor de la Sociedad Científica Informática de España (SCIE) y miembro de la Comisión Ejecutiva de la Confederación de Sociedades Científicas de España (COSCE)

Service Quality Assurance in Context-Aware Systems

Nelly Condori-Fernandez[1]
[1]VU University Amsterdam. The Netherlands
email: *n.condori-fernandez@vu.nl*

Schedule: Tue 20th@09:00, **Room:** A

Context-aware computing and associated technologies are becoming part of daily life. In fact, context-aware computing is considered to be a game-changing opportunity for enterprises to improve both productivity and profits. However, this positive economic growth will also bring a number of issues related to social sustainability (e.g. perceived enjoyment, resilience, privacy).

Given that evidence has corroborated that feelings and emotions dictate to a large extent our actions and decisions, sustainability of context aware systems is approached from the perspective of one of its ingredients, namely social acceptance. In this talk, I will introduce a sustainability assurance framework that exploit emotional information to adjust service quality levels at runtime.

Inteligencia Artificial en el Siglo XXI

Nicolas Kemper[1]
[1]Universidad Nacional Autónoma de México. Mexico Mexico
email: *kemper@unam.mx*

Schedule: Tue 20th@14:00, **Room:** A

En ésta conferencia se trata de proporcionar una perspectiva apropiada y moderna de la inteligencia artificial (IA). Revisamos lo que hemos sido y lo que se ha logrado, se comentan los logros de la IA en el medio siglo pasado en lo que se refiere a búsqueda, representación de conocimiento, razonamiento, aprendizaje, planeación, etc. Se hace un recuento de la IA considerando el paradigma inicial como Inteligencia Artificial Simbólica, pasando por lo que se denomina Inteligencia Computacional y se culmina con el enfoque de Inteligencia Artificial Bioinspirada. También sopesamos nuestras perspectivas actuales bajo la meta de lograr el nivel de una IA más humana. En todo este panorama reflexionamos si la creación de un nivel humano de IA es imposible, o se darán creaciones exitosas en un futuro muy cercano.

Short Biography

Ingeniero Industrial, Universidad Nacional de Trujillo de Perú, Maestría en Ingeniería, especialidad sistemas expertos aplicados en la industria eléctrica y Doctorado en Ingeniería, especialidad en Inteligencia Artificial aplicada en sistemas energéticos, ambos grados en la Universidad Nacional Autónoma de México. Sus líneas de investigación y desarrollo tecnológico son: Desarrollo de sistemas inteligentes aplicados. Gestión del Conocimiento y Organizaciones Inteligentes. Manufactura Inteligente, Instrumentación Virtual Inteligente y Robótica Adaptativa. Minería de datos e Inteligencia de Negocios. Dispositivos de vuelo no tripulados autónomos para monitoreo espacial. Jefe del Departamento de Tecnologías de la Información y del Grupo de Sistemas Inteligentes de este Centro de Investigación. Profesor y Tutor del Postgrado en Ciencias e Ingeniería de la Computación en la misma UNAM. Tiene alrededor de 30 artículos en memorias de eventos y revistas internacionales y regionales, alrededor de 60 reportes técnicos de proyectos de desarrollo tecnológico, ha desarrollado diversos prototipos tecnológicos y software inteligente. Ha publicado el libro "Sistema Inteligente para la Gestión de una Planta Geotermoelétrica", y está en proceso de publicación los libros "Manual para el Desarrollo de Sistemas Expertos", "Desarrollo de Sistemas Difusos", "Inteligencia Artificial Aplicada", "Desarrollo de Proyectos de Minería de Datos" y "Evaluación de Proyectos Energéticos". Miembro del Technical Committee on Artificial Intelligence and Expert System, periodo 2000-2005, de la International Association of Science and Technology for Development (IASTED) de Canadá. Miembro del International Program Committee (IPC) de diversos eventos internacionales. Miembro de la Asociación Mexicana de Economía en Energía; del International Association of Science and Technology for Development (IASTED), Canadá; Miembro Fundador de la Academia Mexicana de Tecnología; Miembro del Institute for Systems and Technologies of Information, Control and Communication de Portugal (INSTICC) y es Miembro Fundador de la Sociedad Peruana de Inteligencia Artificial. Ha tenido diversas distinciones como la Medalla Gabino Barreda por el mejor promedio en sus estudios de maestría en Ingeniería. Así fue honrado con el grado de Doctor Honoris Causa por la Universidad Nacional de Piura, Perú.

Deep Learning for Multimedia Data: Teaching Computers to Sense

Omar Florez[1]
[1]Intel Labs. California USA
email: *omar.u.florez@intel.com*

Schedule: Wed 21st@09:00, **Room:** A

For the past few years, deep learning has been making rapid progress in both techniques and applications; significant performance gains were reported using deep learning in automatic speech recognition and image recognition over hand-optimized feature representations.

Advances in smartphones, tablets, and wearables have made possible to sense a rich collection of user data, examples include audio, images, and video. This information allows us to infer user contexts such as faces, semantic locations, activities, and mood states enabling better and personalized user experiences. This explains the growing interest of industry (Intel, Facebook, Google, NVIDIA, Spotify, Netflix, Baidu, etc.) trying to take advantage of deep learning capabilities in recent years.

In several image and speech tasks, the success of deep learning is due to its ability to learn representations from noisy and unstructured data. Context sensing faces the same problem therefore we believe applications of deep learning in this domain can be advantageous. During this talk we will try bringing together researchers and applicants to discuss some of the deep learning algorithms and capabilities for multimedia and context domains as well as explore possible new research areas.

Short Biography

Dr. Omar U. Florez is a Research Scientist at the Anticipatory Computing Group at Intel Labs (California, USA). He graduated from Universidad Nacional de San Agustin, Peru in 2007 and received his Ph.D. in Computer Science at Utah State University in 2013. He is a recipient of an Innovation Award on Large-Scale Analytics by IBM Research, and the organizer of the NSF-funded Broader Participation in Data Mining workshop at KDD in 2014, which for first time funded the attendance of under-represented researchers worldwide. He is also the co-founder of South Americans in Computing. Dr. Florez's research interests cover statistical machine learning, recommender systems, and deep learning for multimedia data. He has 20+ academic publications and journals in ACM, IEEE, and Springer.

Compact Data Structures

Gonzalo Navarro[1]
[1]Universidad de Chile. Santiago de Chile Chile
email: *gnavarro@dcc.uchile.cl*

Schedule: Wed 21st@14:00, **Room:** A

Compact Data Structures is a new and exciting research area that lies in the intersection of Algorithms and Data Structures, Information Theory, and Data Compression. It has gained much attention in recent years due to the sharp increase in the sizes of the datasets available to applications and the widening gap in the memory hierarchy, which makes it much faster to process data in main memory than in external storage. Compact Data Structures seek for compressed data representations that can be manipulated and queried directly in compressed form, therefore enabling the representation of much larger datasets in main memory. I will describe the main ideas of the area and the two most relevant success stories: the representation of n-node trees in 2n bits and the representation of powerful text indexes within the space of the compressed text.

Short Biography

Gonzalo Navarro completed his PhD in Computer Science in 1998 at the University of Chile, where he is currently full professor. His areas of interest include algorithms and data structures, text searching, compression, and metric space searching. He has directed the Millennium Nucleus Center for Web Research, RIBIDI (an Ibero American project funded by CYTED), and a project funded by Yahoo!

Research, apart from smaller projects. He has participated in various research projects, such as the Millennium Institute for Cell Dynamics and Biotechnology, an ECOS/CONICYT project (Chile-France cooperation), AMYRI (a CYTED project), and a Fondef project. Currently, he participates in the Millennium Nucleus Information and Coordination in Networks and in the Center for Biotechnology and Bioengineering.

He has been PC (co-)chair of several conferences: SPIRE 2001, SCCC 2004, SPIRE 2005, SIGIR 2005 Posters, IFIP TCS 2006, a track of ENC 2007, SISAP 2008, SISAP 2012, and LATIN 2016. He co-created SISAP on 2008, and was Steering Committee member of SPIRE, LATIN, and SISAP. He is a member of the Editorial Board of the journals Information Retrieval, ACM Journal of Experimental Algorithmics, and Information Systems. He has been guest editor of special issues in ACM SIGSPATIAL and Journal of Discrete Algorithmics. He has been PC member of more than 50 international conferences and reviewer for about 40 international journals. He has given around 50 invited talks in several universities and international conferences, including 10 plenary talks and 3 tutorials in international conferences. He created in 2005 the Workshop on Compression, Text, and Algorithms, which has become a permanent satellite of SPIRE.

He has coauthored a book published by Cambridge University Press, about 20 book chapters, 7 proceedings of international conferences (editor), more than 130 papers in international journals, and more than 200 in international conferences. He is one of the most prolific and highly cited authors in Latin America.

Combining Matching Dependencies and Machine Learning via Datalog for Entity Resolution in Databases

Leopoldo Bertossi[1]
[1]Carleton University. Ottawa Canada
email: *bertossi@scs.carleton.ca*

Schedule: Thu 22st@09:00, **Room:** A

In this presentation we give an introduction to the combination of matching dependencies for entity resolution and support-vector machines, a classification method in machine learning. We show that they can be used to address the problems of identifying duplicate representations in data sources and merging them into single representations. The logical glue is provided by Datalog, a logical query language for relational databases, in its incarnation as LogiQL, which has been developed by LogicBlox. This presentation emphasizes the current trend of logically combining different techniques for addressing different interrelated problems in data management.

The presentation will be given in Spanish.

Short Biography

Leopoldo Bertossi has been Full Professor at the School of Computer Science, Carleton University (Ottawa, Canada) since 2001. He is also a Faculty Fellow of the IBM Center for Advanced Studies (IBM Toronto Lab). He has been a professor at the Department of Computer Science, PUC-Chile (until 2001); and also the President of the Chilean Computer Science Society (SCCC). His research interests include data management in general, database theory, business intelligence, data quality, semantic web data, logic-based knowledge representation, and machine learning. He obtained a PhD in Mathematics from the PUC-Chile in 1988.

Learning to select learning algorithms

Andre de Carvalho[1]
[1]University of Sao Paulo. São Carlos-SP Brazil
email: *andre@icmc.usp.br*

Schedule: Thu 22st@14:00, **Room:** A

A large number of learning algorithms have been developed in the last decades and they have been applied to several tasks in different application domains, with different performance levels. According to empirical and theoretical results, no single algorithm can outperform the others in every task. Thus, when using learning algorithms to solve a new task, we are faced with the question of which algorithm to use. Metalearning provides a general framework for the selection of the most suitable algorithm for a new task. This talk will discuss how metalearning can be used for algorithm selection in different learning tasks.

Short Biography

André C. P. L. F. de Carvalho is Full Professor in the department of Computer Science, University of São Paulo, Brazil. His main research interests are data mining, data science and machine learning. Prof. André de Carvalho has more than 300 peer reviewed publications, including 10 best papers awards from conferences organized by ACM, IEEE and SBC. He is a member of the International Association for Statistical Computing (IASC) Council and director of the Center of Machine Learning in Data Analysis of the University of São Paulo.

XML Semantic Disambiguation: Background, Applications, and Ongoing Challenges

Richard Chbeir[1]
[1]University of Pau. Anglet France
email: *richard.chbeir@univ-pau.fr*

Schedule: Thu 22st@17:15, **Room:** A

Since the last two decades, XML has gained momentum as the standard for Web information management and complex data representation. Also, collaboratively built semi-structured information resources such as Wikipedia have become prevalent on the Web and can be inherently encoded in XML. Yet most methods for processing semi-structured information in general and XML in particular handle mainly the syntactic properties of the data, while ignoring the semantics involved. To devise more intelligent applications, one needs to augment syntactic features with machine-readable semantic meaning. This can be achieved through the computational identification of the meaning of data in context, also known as automated semantic analysis and disambiguation, which is nowadays one of the main challenges at the core of the Semantic Web. XML semantic-analysis processing and disambiguation become crucial in an array of applications ranging over semantic-aware query rewriting, semantic document clustering and classification, schema matching, as well as blog analysis and event detection in social networks and tweets. This talk provides a concise and comprehensive review of the methods related to XML-based semi-structured semantic analysis and disambiguation. It will be composed of four parts. First, I will briefly cover traditional word sense disambiguation methods for processing flat textual data. Second, I will describe and categorize disambiguation techniques developed and extended to handle semi-structured and XML data. Third, I will describe current and potential application scenarios that can benefit from XML semantic analysis. Fourth, I will discuss ongoing challenges and future directions.

Short Biography

Richard Chbeir received his PhD in Computer Science from the University of INSA DE LYON-FRANCE in 2001 and then his Habilitation degree in 2010 from the University of Bourgogne. He is currently a Full Professor in the Computer Science Department in IUT de Bayonne in Anglet -France. His current research interests are in the areas of social networks, multimedia semantics, XML and RSS similarity, and digital ecosystems. Richard Chbeir has published in international journals, books, and conferences, and has served on the program committees of several international conferences. He is currently the Chair of the French Chapter ACM SIGAPP.

Testing Based on Finite State Machines: Past, Present and Future

Adenilso Simao[1]
[1]University of Sao Paulo. São Carlos-SP Brazil
email: *adenilso@icmc.usp.br*

Schedule: Fri 23th@09:00, **Room:** A

State machines are among the simplest behavioral models which can be used for system modelling. Nonetheless, they are very expressive and have a long history in computing science. The first testing techniques based on this kind of models are from 50ties. Since then, several contributions have been proposed, including some recent results. In this talk, we are going to introduce the main concepts of testing based on state machines, in special, finite state machines. The main generation methods will be discussed. The state-of-the-art methods will be presented and future directions on this research area will be pointed out.

Short Biography

Possui graduação em Bacharelado em Ciência da Computação pela Universidade Estadual de Maringá (1997), mestrado em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo (2000) e doutorado em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo (2004). Realizou estágio de pós-doutoramento no Centre de Recherche Informatique de Montreal (2008-2010). Desde 2004 é Professor da Universidade de São Paulo (atualmente, é Professor Associado Nível 2). Tem experiência na área de Ciência da Computação, com ênfase em Engenharia de Software, atuando principalmente nos seguintes temas: teste de software, métodos formais e linguagens de programação.

Cloud Security and Forensics

George J. Proeller[1]
[1]Colorado Technical University. Colorado Springs USA
email: *gproeller@comcast.net*

Schedule: Fri 23th@10:15, **Room:** A

Cloud Computing offers an opportunity to increase capability with a promise of decreasing overall IT costs. The Cloud is also increasingly coupled with the ubiquity of mobile computing and its advantages. Both are now a part of everyday life - Amazon, Google, Dropbox, iTunes Cloud, RealAudio Cloud, etc. However the "Benefits.ºf the Cloud cause us to forfeit aspects of control, visibility, and tracking data origin, source, and attribution. Traditional Computer Forensics requires physical access to systems for processes such as disk imaging something not readily available when working in the Cloud. Further the physical location of data can be very difficult to determine in a private cloud and a near impossible task in a public cloud where the data may not even reside in the same country as the user. This talk discusses the challenges and opportunities within the emerging areas of cloud and mobile forensics.

Short Biography

Thirty-plus years' experience in Information Technology Systems (ITS) focusing on life-cycle Information Assurance/Computer Network Security for the Enterprise. He holds multiple security certifications including the Certified Information Systems Security Professional (CISSP) credential, the Certified Information Security Manager, and the GIAC Security Leadership Certification and Certified Information Forensics Investigator and is a pioneer in the transition of information security to academia. His awards include being named a Distinguished Fellow of the Information Systems Security Association (ISSA) and to the ISSA Hall of Fame.

Planificación Óptima de Sistemas Multinúcleo con restricciones de calidad de servicio, recursos y energía

Rodrigo Santos[1]
[1]Universidad Nacional del Sur - CONICET. Bahía Blanca Argentina
email: *ierms@criba.edu.ar*

Schedule: Tue 20th@15:15, **Room:** A

Dos hechos fundamentales contribuyeron al desarrollo de los sistemas móviles. Por un lado el abaratamiento del hardware y la expansión de la memoria disponible. Por el otro la evolución de las arquitecturas. De hecho practicamente cualquier microcontrolador, aún aquéllos económicos tienen características multicore. Si bien puede que la unidad de procesamiento sea única, cuentan con dispositivos periféricos para las comunicaciones, el tratamiento de señales de entrada/salida y el manejo de tiempos como mínimo.

La consolidación de las tecnologías de comunicación inalámbrica permitió el establecimiento de redes de sensores y actuadores que miden, evalúan y controlan procesos a distancia. Estos procesos abarcan áreas tan diversas como la industria de procesos continuos al monitoreo ambiental, las redes sociales, la administración del sistema de transporte público o la atención temprana de incidentes urbanos o ambientales. En esta clase de sistemas la validación temporal de la información administrada y la coherencia de los mismos son aspectos centrales para el funcionamiento del sistema.

Un adecuado análisis del comportamiento temporal de esta clase de sistemas requiere de políticas de planificación que puedan garantizar el cumplimiento de los requerimientos. Se suma al problema de la planificación, las restricciones que la calidad de servicio requerida y el consumo de energía imponen en la actualidad sobre los sistemas.

En el curso se presenta la teoría de planificación clásica, los mecanismos de verificación y validación de las mismas, los criterios de optimalidad para sistemas de cómputo y comunicaciones. El modelado adecuado de esta clase de problemas permite transferir resultados de un área a otra.

Short Biography

Dr. Rodrigo Santos es Prof. del Departamento de Ingeniería Eléctrica y de Computadoras de la Universidad Nacional del Sur e Investigador Adjunto del CONICET. Su área de interés es la planificación de sistemas de tiempo real, el análisis de requerimientos, los sistemas colaborativos y los sistemas embebidos. Fue presidente de CLEI entre 2008 y 2012, miembro del IEEE. Autor de varios artículos en revistas y congresos de la especialidad.

Seguridad en la Web

Ernst Leiss[1]
[1]University of Houston. Houston USA
email: *coscel@cs.uh.edu*

Schedule: Wed 21st@15:15, **Room:** A

Empezamos con métodos para la protección de contenido digital, desde criptografía hasta marcas de agua digitales. Después hablamos sobre control de acceso, incluyendo contraseñas y métodos biométricos. De ahí pasamos a crímenes que involucran o dependen de la web. Terminamos con algunos comentarios sobre privacidad y los efectos de la tecnología en el mundo cotidiano.

Short Biography

Ernst Leiss obtuvo grados académicos en ingeniería (Dipl.-Ing., TU Viena, 1975), ciencia de la computación (M. Math., U. Waterloo, 1974), y matemática (Dr. techn., TU Viena, 1976). Trabajó como post doc en la Universidad de Waterloo (1976/7) y como profesor investigador en la Universidad de Chile en Santiago (1978). Empezó en el departamento de ciencias de la computación en 1979 donde todavía trabaja. Sus intereses incluyen teoría de lenguajes formales, computación de alto desempeño, y seguridad. Ha sido supervisor de 17 disertaciones doctorales y de mas de 100 tesis de maestría. Es autor de seis libros y mas de 160 trabajos científicos en conferencias y revistas.

Leiss ha participado en cada conferencia CLEI desde 1992. Se involucró con LANC, para definir un modelo sostenible para la conferencia. Ha participado en numerosos comités de programa, en particular como General Program Chair de la conferencia en Quito en 2011, donde se usó por primera vez el modelo de simposios. Ha participado en un ECI curso en Buenos Aires y dos ERTIC cursos en

Asuncion. Además, ha dictado cursos en Siria, Brasil, Italia, España, Alemania, Finlandia, Tunisia, y Marruecos. Hasta la fecha, ha dado seminarios en 32 diferentes países.

Leiss ha participado en la acreditación de programas en informática. Ha sido un ACM Distinguished Lecturer desde 1991. Ha ocupado puestos administrativos en su Universidad y actuado como presidente de varios comités; en particular, en 1994 como presidente del Senado de su universidad, inició una mayor reestructuración administrativa de la Universidad de Houston. En 2014, fue premiado con la Distinción CLEI al Mérito Latinoamericano en Informática.

Digital Disruption

Juan Pablo Guizado Serrano[1]
[1]Oracle. Lima Peru
email: *juan.guizado@oracle.com*

Schedule: Fri 23th@10:15, **Room:** B

Existe una latente necesidad de innovación de las organizaciones para garantizar su éxito en la era digital, y hacer de la disrupción digital una oportunidad para la innovación.

Tendencias como el Cloud Computing, Big Data, el Internet de las cosas, redes sociales y la movilidad se han revelado como los principales ejes de la transformación de las empresas. Por ello consideran que el uso de estas tecnologías está ocasionando una disrupción digital masiva en las empresas. Debemos estar consientes que la tecnología aún no ha llegado a su más alto nivel de desarrollo, al contrario estamos recién entrando al momento en donde el despliegue correcto de ésta mostrará su verdadero valor.

Short Biography

Con casi 15 años de experiencia en Tecnologías de la Información, el Ingeniero Juan Pablo Guizado desde hace 3 años es Consultor en Soluciones de Capa Media en Oracle Perú, especialista en productos de Automatización de Proceso de negocio y Optimización de integración de sistemas empresariales, anteriormente ha sido Administrador de Plataforma de Misión Crítica para Aplicaciones Java y Arquitecto de Aplicaciones multiplataforma, conocedor además de construcción de sistemas participando en múltiples proyectos de envergadura nacional.

Conditional Monte Carlo with Intermediate Estimations for simulation of Markovian systems

Héctor Cancela
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
cancela@fing.edu.uy

Leslie Murray
FCEIA
Universidad Nacional de Rosario
Rosario. Argentina
leslie@eie.fceia.unr.edu.ar

Gerardo Rubino
INRIA–Rennes
Bretagne Atlantique, Campus de Beaulieu
Rennes, France
Gerardo.Rubino@inria.fr

***Abstract*—For systems that are suitable to be modelled by continuous Markov chains, dependability analysis is not always straightforward. When such systems are large and complex, it is usually impossible to compute their dependability measures exactly. An alternative solution is to estimate them by simulation, typically by Monte Carlo simulation. But for highly reliable systems standard simulation can not reach satisfactory accuracy levels (measured by the variance of the estimator) within reasonable computing times. Conditional Monte Carlo with Intermediate Estimations (CMIE) is a simulation method proposal aimed at making accurate estimations of dependability measures on highly reliable Markovian systems. The basis of CMIE is introduced, the unbiasedness of the corresponding estimator is proven, and its variance is shown to be lower than the variance of the standard estimator. A variant of the basic scheme, that applies to large and highly reliable multicomponent systems, is introduced. Some experimental results are shown.**

***Keywords*—dependability, simulation, rare event, Conditional Monte Carlo.**

I. Introduction

We consider systems which can be modelled by a continuous time homogeneous Markov chain X irreducible on the finite state space S (see [1], [2], [3] or Chapter 6 in [4]). The chain can also be absorbing and the techniques described here still work, but they are easier to present in the irreducible case. In this context, some measures of dependability need the evaluation of the probability $\gamma = \mathbb{P}\{\tau_D < \tau_{\mathbf{u}}\}$, where the times $\tau_{\mathbf{u}}$ and τ_D are defined as follows. The state space of the Markov chain is partitioned as $S = U \cup D$, such that in U the system is *up* and in D the system is *down*. The process X starts at some initial state $\mathbf{u} \in U$. Define $\tau_{\mathbf{u}}$ as the *return time to* $\mathbf{u}$, that is, $\tau_{\mathbf{u}} = \inf\{t > 0 \colon X(t) = \mathbf{u}$ and $X(t^-) \neq \mathbf{u}\}$, and τ_D as the *hitting time of* D, that is, $\tau_D = \inf\{t > 0 \colon X(t) \in D\}$.

The simplest and most basic dependability metric is the Mean Time To Failure (MTTF), defined as the expected life–time of the system, that is, the mean time until the system enters the subset D: MTTF = $\mathbb{E}\{\tau_D\}$. This metric admits the well–known representation $\mathrm{MTTF} = \mathbb{E}\{\min(\tau_D, \tau_{\mathbf{u}})\}/\gamma$.

Since we focus on the estimation of γ, we can just collapse all D into a single state $\mathbf{d}$, made absorbing. As before, event $\{\tau_{\mathbf{d}} < \tau_{\mathbf{u}}\}$ means that X gets absorbed at $\mathbf{d}$ before coming back to $\mathbf{u}$. For systems with a large (or infinite) number of states, the exact computation of γ is not feasible, and the standard *Monte Carlo* simulation will work, unless $\gamma \ll 1$, in which case we are facing a *rare event* problem, a context in which acceptable values of the estimator's variance can only be achieved at the expense of a very high number of replications. *Monte Carlo* methods must therefore be improved and adapted to address efficiently this *rare event* problem. Research has resulted in a large number of solutions in this regard, most of which derive from two well known families of techniques named, respectively, *Splitting* [5], [6], [7], [8], [9] and *Importance Sampling* [1], [10], [11].

Some applications of *Splitting* in the context of highly reliable systems can be found in [12] and [13], where the *reliability* and availability estimations of repairable systems are analysed using a variant called RESTART. Recently, some results in the context of static systems have been published in [14] and [15]. Some methods derived from *Importance Sampling*, like Zero-Variance [16], [17], [18] and Cross-Entropy [19], [20] have been successfully applied in the simulation of systems affected by *rare events*.

Conditional Monte Carlo [15], [21] is a classic variance reduction technique that has not given rise to many methods in the field of rare events applied to reliability estimation. However, some applications can be found in [22], [23], [24], [25], but most of them are aimed at the *rare events* probability estimation in models that deal with heavy–tailed distributions. This article addresses the problem of reliability estimation in the model so far defined and introduces a *Conditional Monte Carlo* simulation scheme, suitable for the estimation of γ.

The rest of this paper is organized as follows. Section II shows a basic application of Conditional Monte Carlo on Markovian systems. Section III, the core of this paper, introduces modifications to the basic Conditional Monte Carlo algorithm, in order to make it usable and efficient. Sections IV, V and VI discuss some properties and features of the proposed method. Section VII shows how to apply it to the particular case of Markovian multicomponent systems. Some experimental results are included in Sections VI and VII. A comparison with *Splitting* is shown in Section VIII. Conclusions and future directions can be found in Section IX.

II. Conditional Monte Carlo algorithm

There are different simulation methods to estimate value of γ. In the crude or standard simulation, N_1 replications start at state $\mathbf{u}$ and they are simulated until they either come back to

$\mathbf{u}$, in time $\tau_{\mathbf{u}}$, or hit state $\mathbf{d}$, in time $\tau_{\mathbf{d}}$. Let I be the indicator random variable of the event $\{\tau_{\mathbf{d}} < \tau_{\mathbf{u}}\}$:

$$I = \begin{cases} 1 & \text{w.p.} \quad \gamma, \\ 0 & \text{w.p.} \quad 1-\gamma. \end{cases} \tag{1}$$

Then, $\gamma = \mathbb{E}\{I\}$. Its standard estimator, $\widehat{\gamma}_s$, is:

$$\widehat{\gamma}_s = \frac{1}{N_1}\sum_{j=1}^{N_1} I^{(j)}, \tag{2}$$

where $I^{(j)}, j = 1, 2, \ldots, N_1$ are N_1 independent values sampled from distribution (1).

Let $C = \{\mathbf{d}, k, \mathbf{u}\}$, where k is any state in S, other than $\mathbf{d}$ or $\mathbf{u}$, and let X_C be a random variable defined as *the first state in C hit by a replication started at* $\mathbf{u}$:

$$X_C = \begin{cases} \mathbf{d} & \text{w.p.} \quad p_{\mathbf{d}}, \\ k & \text{w.p.} \quad p_k, \\ \mathbf{u} & \text{w.p.} \quad p_{\mathbf{u}}. \end{cases}$$

See that $p_{\mathbf{d}} \leq \gamma$, because γ is the probability that any replication that starts at $\mathbf{u}$ reaches $\mathbf{d}$ before coming back to $\mathbf{u}$, whereas $p_{\mathbf{d}}$ is the same probability, provided that "the path does not contain k". Similarly, $p_{\mathbf{u}} \leq 1-\gamma$.

The expectation of I, conditioned on the values of X_C, is given by the following expressions: $\mathbb{E}\{I \mid X_C = \mathbf{d}\} = 1$, $\mathbb{E}\{I \mid X_C = k\} = \gamma_k$ and $\mathbb{E}\{I \mid X_C = \mathbf{u}\} = 0$ (γ_k is the probability that a replication that starts at state k, hits state $\mathbf{d}$ before it hits state $\mathbf{u}$). Thus, $\mathbb{E}\{I \mid X_C\}$ is a random variable with the following probability distribution:

$$\mathbb{E}\{I \mid X_C\} = \begin{cases} 1 & \text{w.p.} \quad p_{\mathbf{d}}, \\ \gamma_k & \text{w.p.} \quad p_k, \\ 0 & \text{w.p.} \quad p_{\mathbf{u}}, \end{cases} \tag{3}$$

and the following expectation:

$$\mathbb{E}\{\mathbb{E}\{I \mid X_C\}\} = \mathbb{E}\{I\} = 1 \times p_{\mathbf{d}} + \gamma_k \times p_k + 0 \times p_{\mathbf{u}} = \gamma.$$

The expected value of both random variables, I and $\mathbb{E}\{I \mid X_C\}$, is γ. As a consequence, another estimator of γ —namely, a *Conditional Monte Carlo* estimator— is:

$$\widehat{\gamma}_c = \frac{1}{N_1}\sum_{j=1}^{N_1} \mathbb{E}\{I \mid X_C^{(j)}\}, \tag{4}$$

where $\mathbb{E}\{I \mid X_C^{(j)}\}, j = 1, 2, \ldots$ are N_1 independent random variables sharing distribution (3). The samples $\mathbb{E}\{I \mid X_C^{(j)}\}$ are obtained in two steps: first, $X_C^{(j)}$ is sampled and then, the corresponding value $\mathbb{E}\{I \mid X_C^{(j)}\}$ is *computed*. In this introductory example the only three possible values of $X_C^{(j)}$ to be sampled are $\{\mathbf{u}, k, \mathbf{d}\}$, whereas the exact values of $\mathbb{E}\{I \mid X_C^{(j)}\}$ associated with them are, respectively, $\{1, \gamma_k, 0\}$.

If the set C includes more intermediate states besides k, the method applies as well. If, for example, $C = \{\mathbf{d}, 1, 2, \ldots, n, \mathbf{u}\}$, the distribution of $\mathbb{E}\{I \mid X_C\}$ becomes:

$$\mathbb{E}\{I \mid X_C\} = \begin{cases} 1 & \text{w.p.} \quad p_{\mathbf{d}}, \\ \gamma_1 & \text{w.p.} \quad p_1, \\ \gamma_2 & \text{w.p.} \quad p_2, \\ \vdots & \\ \gamma_n & \text{w.p.} \quad p_n, \\ 0 & \text{w.p.} \quad p_{\mathbf{u}}, \end{cases} \tag{5}$$

where γ_i is the probability that a replication that starts at state i hits state $\mathbf{d}$ before it hits state $\mathbf{u}$. Now:

$$\begin{aligned} \mathbb{E}\{\mathbb{E}\{I \mid X_C\}\} &= \mathbb{E}\{I\} = 1 \times p_{\mathbf{d}} + \sum_{i=1}^{n} \gamma_i \, p_i + 0 \times p_{\mathbf{u}} \\ &= \sum_{i=0}^{n} \gamma_i \, p_i = \gamma, \end{aligned}$$

where the notation $\gamma_0 = 1$ and $p_0 = p_{\mathbf{d}}$ is included for simplicity. The estimator given in Expression (4) remains valid, with the only difference of sampling from the distribution (5) instead of (3).

Figure 1 depicts the set of probabilities so far defined and shows the nomenclature used to refer to them in the rest of this article (as $\gamma_{\mathbf{u}} = 0$, the term $p_{\mathbf{u}} \times \gamma_{\mathbf{u}}$ equals 0, reason why it is shown in Figure 1 but does not appear in any further expression).

For any given set $C = \{\mathbf{d}, 1, 2, \ldots, n, \mathbf{u}\}$, call $\widetilde{C} = C \setminus \{\mathbf{d}, \mathbf{u}\}$, i.e. the subset formed only by the intermediate states, that is, $\widetilde{C} = \{1, 2, \ldots, n\}$.

The variance of the *Conditional Monte Carlo* estimator in (4) is:

$$\begin{aligned} \mathbb{V}\{\widehat{\gamma}_c\} &= \frac{1}{N_1}\left(\mathbb{E}\{\mathbb{E}\{I \mid X_C\}^2\} - \mathbb{E}\{\mathbb{E}\{I \mid X_C\}\}^2\right) \\ &= \frac{1}{N_1}\left(\sum_{i=0}^{n} p_i \gamma_i^2 - \gamma^2\right). \end{aligned} \tag{6}$$

On the other hand, the variance of the standard estimator given in (2) is known to be:

$$\mathbb{V}\{\widehat{\gamma}_s\} = \frac{1}{N_1}\left(\gamma - \gamma^2\right) = \frac{1}{N_1}\left(\sum_{i=0}^{n} p_i \gamma_i - \gamma^2\right). \tag{7}$$

Comparing expressions (6) and (7) and considering that $\gamma_i \leq 1,\ i = 0, \ldots, n$, because all these values are probabilities, it is clear that:

$$\sum_{i=0}^{n} p_i \gamma_i^2 \leq \sum_{i=0}^{n} p_i \gamma_i,$$

what means that the variance of the *Conditional Monte Carlo* estimator given in (6), is never larger than the *Standard Monte Carlo* estimator variance given in (7). This is, of course, a general fact on *Conditional Monte Carlo* methods, but it is worth making it explicit in our context.

III. Conditional Monte Carlo with Intermediate Estimations

The main problem in the use of *Conditional Monte Carlo*, as it was introduced so far, is the fact that the values γ_1, γ_2, …, γ_n are unknown, and that may be even as hard to evaluate as the exact value of γ itself. To work around this problem, these values will be now replaced by estimators.

It will be shown that after such replacement, the method is still unbiased. This is the core of the proposal introduced in this article and the basis of the so–called *Conditional Monte Carlo with Intermediate Estimations* (*CMIE*) method. The method

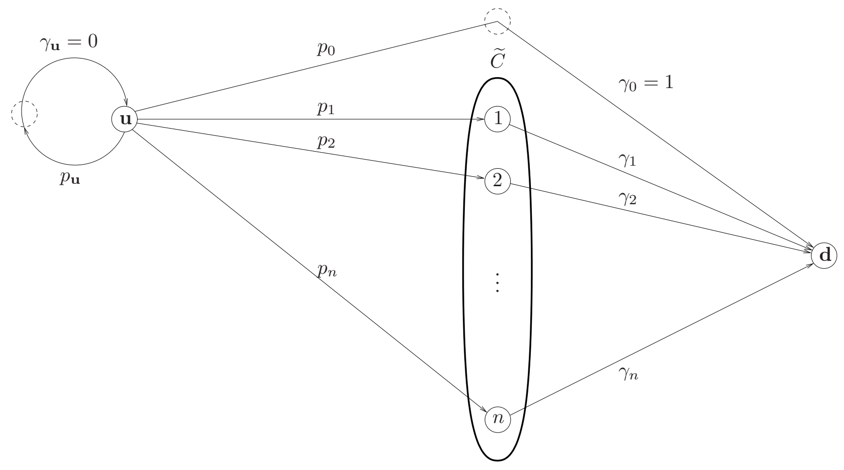


Fig. 1: The set of probabilities used in all calculations.

will now be described and, at the end of this section, the variance of the corresponding estimator will be determined.

To address the following calculation, it is better to express $\widehat{\gamma}$ in terms of the random vector $\bar{I} = (I_0, I_1, \ldots, I_{n+1})$, whose components are dependent binary random variables such that one and only one has value 1:

$$\bar{I} = \left\{ \begin{array}{lll} (1,0,0,\ldots,0,0) & \text{w.p.} & p_{\mathbf{d}}, \\ (0,1,0,\ldots,0,0) & \text{w.p.} & p_1, \\ (0,0,1,\ldots,0,0) & \text{w.p.} & p_2, \\ & \vdots & \\ (0,0,0,\ldots,1,0) & \text{w.p.} & p_n, \\ (0,0,0,\ldots,0,1) & \text{w.p.} & p_{\mathbf{u}}. \end{array} \right. \tag{8}$$

Then, being $\gamma_0 = 1$, the standard estimator, $\widehat{\gamma}_s$, is:

$$\begin{aligned} \widehat{\gamma}_s &= \frac{1}{N_1} \sum_{j=1}^{N_1} \; I_0^{(j)} \times \gamma_0 + I_1^{(j)} \times \gamma_1 + I_2^{(j)} \times \gamma_2 + \ldots \\ &\quad + I_n^{(j)} \times \gamma_n + I_{n+1}^{(j)} \times 0 \\ &= \frac{1}{N_1} \sum_{j=1}^{N_1} \sum_{k=0}^{n} I_k^{(j)} \times \gamma_k \end{aligned} \tag{9}$$

In (9), the samples $I_0^{(j)}$, $I_1^{(j)}$, …, $I_n^{(j)}$ are obtained by the simulation, whereas the values γ_1, γ_2, …, γ_n must be calculated. However, if such calculation is too hard, or simply impossible, these values can be replaced by standard estimators. In order to do this, every time the simulation reaches a state $i \in \widetilde{C}$, N_2 independent replications must be started at i and simulated until they either reach $\mathbf{d}$ (and accumulate 1) or $\mathbf{u}$ (and accumulate 0). Once these N_2 replications started at i are completed, a standard estimator $\widehat{\gamma}_i$ can be evaluated and used in place of γ_i. To compute these estimations, define the set of Bernoulli random variables $\{J_i\}_{i=1}^{n}$, with the following probability distribution:

$$J_i = \left\{ \begin{array}{lll} 1 & \text{w.p.} & \gamma_i, \\ 0 & \text{w.p.} & 1-\gamma_i, \end{array} \right. \qquad i = 1,2,\ldots,n. \tag{10}$$

The samples of J_i are obtained from the actual simulation of the Markov chain, which is the same as sampling them from distribution (10) ($J_0 = 1$ w.p. 1). Then, if γ_k is replaced by the estimator $\widehat{\gamma}_k$ in (9), the standard estimator is transformed into the *CMIE* estimator $\widehat{\gamma}_{cie}$:

$$\begin{aligned} \widehat{\gamma}_{cie} &= \frac{1}{N_1} \sum_{j=1}^{N_1} \left(\sum_{k=0}^{n} I_k^{(j)} \times \frac{1}{N_2} \sum_{i=1}^{N_2} J_k^{(i)} \right) \\ &= \frac{1}{N_1 N_2} \sum_{j=1}^{N_1} \sum_{k=0}^{n} \sum_{i=1}^{N_2} I_k^{(j)} J_k^{(i)}. \end{aligned}$$

It is simple to show that $\widehat{\gamma}_{cie}$ is unbiased:

$$\begin{aligned} \mathbb{E}\left\{\widehat{\gamma}_{cie}\right\} &= \frac{1}{N_1 N_2} \; \mathbb{E}\left\{ \sum_{j=1}^{N_1} \sum_{k=0}^{n} \sum_{i=1}^{N_2} I_k^{(j)} J_k^{(i)} \right\} \\ &= \frac{1}{N_1 N_2} \; \sum_{j=1}^{N_1} \sum_{k=0}^{n} \sum_{i=1}^{N_2} p_k \gamma_k \\ &= \sum_{k=0}^{n} \; p_k \gamma_k \;\; = \;\; \gamma. \end{aligned}$$

To determine the the variance of $\widehat{\gamma}_{cie}$, let $\bar{I}^{(x)}$ be any possible replication of $\bar{I}$, what means that $I_0^{(x)}$, $I_1^{(x)}$, …, $I_n^{(x)}$ are the components of this replication. Using the variance decomposition formula, the variance of the estimator can be written as:

$$\mathbb{V}\{\widehat{\gamma}_{cie}\} = \underbrace{\mathbb{V}\{\mathbb{E}\{\widehat{\gamma}_{cie} \mid \bar{I}^{(x)}\}\}}_{A} + \underbrace{\mathbb{E}\{\mathbb{V}\{\widehat{\gamma}_{cie} \mid \bar{I}^{(x)}\}\}}_{B}.$$

Terms A and B are analysed separately and after some algebra (see [26]), we obtain:

$$\begin{aligned} \mathbb{V}\{\widehat{\gamma}_{cie}\} = A + B &= \frac{1}{N_1} \left(\sum_{k=0}^{n} p_k \gamma_k^2 - \gamma^2 \right) + \\ &\quad \frac{1}{N_1 N_2} \left(\gamma - \sum_{k=0}^{n} \; p_k \gamma_k^2 \right). \end{aligned} \tag{11}$$

Term A is the value of the variance of the *Conditional Monte Carlo* estimator when the values γ_1, γ_2, …, γ_n are known exactly (see (6)). Term B is the variance increase due to the fact that the values γ_1, γ_2, …, γ_n are replaced by estimators.

IV. Multiple Sets of Intermediate States

The key to the application of our *Conditional Monte Carlo* to Markov chains (as described in Section II) —call it *pure Conditional Monte Carlo*— is the knowledge of the probabilities $\gamma_1, \gamma_2, \ldots, \gamma_n$. The lack of these values makes it necessary to use estimations instead (as described in Section III). This technique is the heart of the *CMIE* method proposed in this article. As shown, the estimators $\widehat{\gamma}_1, \widehat{\gamma}_2, \ldots, \widehat{\gamma}_n$ can be obtained by standard simulation started every time one of the intermediate states 1, 2, …, n is reached. But these values can be estimated more accurately, applying the same *Conditional Monte Carlo* method recursively, in the following way.

Suppose that two sets of intermediate states, $\widetilde{C}_1$ and $\widetilde{C}_2$, are defined, instead of one, as shown Figure 2. Assume that $\widetilde{C}_1 \cap \widetilde{C}_2 = \emptyset$ and $\mathbf{u}, \mathbf{d} \notin \widetilde{C}_1, \widetilde{C}_2$. Then, once a state $i \in \widetilde{C}_1$ is reached, N_2 replications must be started at state i, and they must be simulated until they either hit a state in $\widetilde{C}_2$, go back to $\mathbf{u}$, or get absorbed at $\mathbf{d}$. This can be considered the second recursive level of the simulation. It is intended to obtain the values $\gamma'_1, \gamma'_2, \ldots, \gamma'_{n_1}$, which indicate the probability that each of these N_2 replications get absorbed at $\mathbf{d}$. These values are not estimated by means of standard simulation, they are estimated more accurately by this recursive level of *Conditional Monte Carlo* simulation that makes use of $\widetilde{C}_2$ as the set of intermediate states. It is simple to extend this mechanism to more recursive levels (with more sets of intermediate states).

The variance analysis can be extended to the case of two sets of intermediate states, $\widetilde{C}_1$ and $\widetilde{C}_2$, in a straightforward manner. The probabilities involved are shown in Figure 2. The variance, computed in [26], is the following:

$$\begin{aligned}\mathbb{V}\{\widehat{\gamma}_{cie}\} = & \frac{1}{N_1}\left(\sum_{l=0}^{n_1} p_l\, {\gamma'_l}^2 - \gamma^2\right) + \\ & \frac{1}{N_1}\sum_{l=0}^{n_1} p_l \left(1/N_2\left(\sum_{k=0}^{n_2} p_{lk}\gamma_k^2 - {\gamma'_l}^2\right) + \right. \\ & \left. \frac{1}{N_2 N}\left(\gamma'_l - \sum_{k=0}^{n_2} p_{lk}\,\gamma_k^2\right)\right).\end{aligned}$$

Given this expression, it follows that the variance obtained in a model with two sets of intermediate states, $\widetilde{C}_1$ and $\widetilde{C}_2$, is lower than or equal to the variance obtained in a model with the single intermediate set of states $\widetilde{C}_1$.

V. Comparative Analysis of Variances

The variance of the *CMIE* estimator for the case of only one set of intermediate states, $\mathbb{V}\{\widehat{\gamma}_{cie}\}$, was derived in Section III. In this section, this variance is compared to the variance of other estimators, namely, the variance of the standard estimator, $\mathbb{V}\{\widehat{\gamma}_s\}$, shown in (7), and the variance of the *pure Conditional Monte Carlo* estimator, $\mathbb{V}\{\widehat{\gamma}_c\}$, derived in (6).

As $N_2 \to \infty$, $\mathbb{V}\{\widehat{\gamma}_{cie}\} \to \mathbb{V}\{\widehat{\gamma}_c\}$. Clearly, if the number of replications used in the estimation of the probabilities $\gamma_i,\ i = 0, \ldots, n$, is infinite, the estimators converge to the corresponding exact values, and the method becomes the *pure Conditional Monte Carlo*.

At the end of Section II it has been shown that $\mathbb{V}\{\widehat{\gamma}_c\} \le \mathbb{V}\{\widehat{\gamma}_s\}$, meaning that the accuracy of *pure Conditional Monte Carlo* is never less than the accuracy of *Standard Monte Carlo*. It is clear that $\mathbb{V}\{\widehat{\gamma}_c\} \le \mathbb{V}\{\widehat{\gamma}_{cie}\}$. We now prove that $\mathbb{V}\{\widehat{\gamma}_{cie}\} \le \mathbb{V}\{\widehat{\gamma}_s\}$, meaning that the proposed estimator is never less accurate than *Standard Monte Carlo* estimator.

$$\begin{aligned}\mathbb{V}\{\widehat{\gamma}_{cie}\} &= \frac{\sum\limits_{k=0}^{n} p_k\gamma_k^2 - \gamma^2}{N_1} + \frac{\gamma - \sum\limits_{k=0}^{n} p_k\gamma_k^2}{N_1 N_2} \\ &\le \frac{\sum\limits_{k=0}^{n} p_k\gamma_k^2 - \gamma^2}{N_1} + \frac{\gamma - \sum\limits_{k=0}^{n} p_k\gamma_k^2}{N_1} = \frac{\gamma - \gamma^2}{N_1} \\ &\le \mathbb{V}\{\widehat{\gamma}_s\}.\end{aligned}$$

The inequality holds, no matter the values of N_1 and N_2. This means that the proposed estimator, $\widehat{\gamma}_{cie}$, is never less accurate than *Standard Monte Carlo* estimator, $\widehat{\gamma}_s$, even for a low number of replications N_1 and N_2.

Finally, the three variances involved are related as follows: $\mathbb{V}\{\widehat{\gamma}_c\} \le \mathbb{V}\{\widehat{\gamma}_{cie}\} \le \mathbb{V}\{\widehat{\gamma}_s\}$, which means that *CMIE* is always more accurate than crude or *Standard Monte Carlo*, but never as accurate as *pure Conditional Monte Carlo*, in which the exact values $\gamma_i,\ i = 0, \ldots, n$, are used.

VI. Intermediate States Analysis

The variance reduction capacity of *CMIE* depends on the choice of the set of intermediate states. In this section two properties of the sets of intermediate states are considered. Their proofs can be seen in [26]. The first one states that after adding a new state to an existing set, the variance of the estimator never increases and, therefore, a variance reduction may be expected. The second one says that if we compare the variance reduction obtained by two disjoint sets, the highest variance reduction comes from the cut that is somehow "closer" to the initial state, $\mathbf{u}$.

These two properties are consistent because, if the addition of one state to an existing set of intermediate states yields a variance that is lower than, or at worst equal to, the variance before the addition, the set that yields the least variance is the one composed of all the states: $C = S$. But, from the implementation point of view, the set $C = S$ is equivalent to the set formed by the adjacent states to $\mathbf{u}$, because, if $C = S$, for any replication started at the initial state $\mathbf{u}$, the only reachable states are the adjacent ones.

In the case of two or more sets of intermediate states, the choice of the second, and the consecutive ones, must be somehow similar to the choice of the first one with respect to the initial state $\mathbf{u}$. Whenever possible, the second set (between the existing one and state $\mathbf{d}$) must be formed by the adjacent states to the existing set. However, this is not straightforward and must be analyzed for every particular model.

These two properties are now tested on a continuous time Markov chain proposed and used by Juneja and Shahabuddin in [3] and shown in Figure 3. The system has 2 components of class A and 4 components of class B. The components can only be *operational* or *failed*. The state is the pair (N_A, N_B), where N_i indicates the number of *failed* components in class i. Failure rates of classes A and B are, respectively, $\epsilon/2$ and

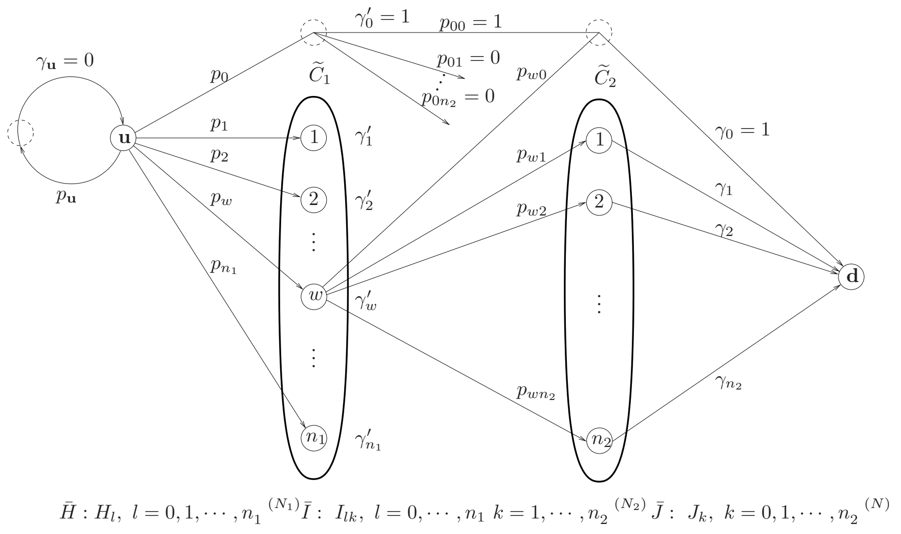


Fig. 2: The case of two sets of Intermediate States, $\widetilde{C}_1$ and $\widetilde{C}_2$

ϵ. The system fails if all components of all classes fail. Group repair (all *failed* components of a class are repaired simultaneously) begins if two components of the same class fail. Group repair rates for both classes are equal to 1. There is one repair-person in the system, and class A gets preemptive priority over class B.

Tables I, II and III show the results obtained by *CMIE* simulations over this model. The sets $\widetilde{C}_1$, $\widetilde{C}_2$ and $\widetilde{C}_3$ are all cuts between $\mathbf{u}$ and $\mathbf{d}$, and they are referred to, making use of the numbers placed above each state in Figure 3. Table III shows the ratio $\widehat{\mathbb{V}}\{\widehat{\gamma}_s\}/\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\}$ when *CMIE* and standard simulation run the same execution time.

The *CMIE* method was programmed in the `C` language, using the `gcc` compiler. The estimator $\widehat{\gamma}_{cie}$ and an unbiased estimator of its variance were calculated as follows:

$$\widehat{\gamma}_{cie} = \frac{1}{N_1}\sum_{j=1}^{N_1} \gamma^{(j)} \quad \text{and}$$

$$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\} = \frac{1}{N_1 - 1}\left(\frac{1}{N_1}\left(\sum_{j=1}^{N_1} {\gamma^{(j)}}^2\right) - \widehat{\gamma}_{cie}^2\right). \qquad (12)$$

The results in Table I show that the cut that attains the lowest variance is the one formed by the adjacent states to $\mathbf{u}$. The variances of the estimators whose associated cut is close to state $\mathbf{u}$ are also low and quite similar. But, when the cuts are "far" from state $\mathbf{u}$, the variance greatly increases. In these experiments the number of replications started at state $\mathbf{u}$ was 10,000 and the number of replications launched from the intermediate states was also 10,000.

The experiment whose results are in Table II show that as the number of cuts increases, the variance of the estimator $\widehat{\gamma}_{cie}$ decreases. In these experiments the number of replications started at state $\mathbf{u}$ was 10,000 and the number of replications launched from the intermediate states was 100 for all cases.

The experiments in Table III are included to briefly show the variance reduction capacity of the *CMIE* method. The number of replications launched from intermediate states was 1,000 for all cases; the number of replications starting at state $\mathbf{u}$ was adjusted so that the total execution time of each of the four simulations was $t = 500$ sec. This time was fixed in advance and equal for all methods in order to have a fair comparison of the accuracy that was obtained by the different experiments.

TABLE I: Model in Figure 3, $\epsilon = 0.01$

$\widetilde{C}_1$	$\widehat{\gamma}_{cie}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\}$
1–5	6.18E−06	6.28E−14
2–6–10	6.23E−06	6.88E−14
3–7–11	6.36E−06	6.90E−14
4–8–12	6.34E−06	1.53E−13
9–13	6.93E−06	6.56E−12

TABLE II: Model in Figure 3, $\epsilon = 0.01$

$\widetilde{C}_1$	$\widetilde{C}_2$	$\widetilde{C}_3$	$\widehat{\gamma}_{cie}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\}$
1–5	—	—	4.00E−06	4.00E−12
1–5	2–6–10	—	6.19E−06	6.31E−14
1–5	2–6–10	3–7–11	6.13E−06	2.27E−15

VII. Application to Large Systems

Sometimes the state space S of the Markov chain is extremely large and, therefore, the choice of intermediate states is hard to be done explicitly. The idea of *CMIE* fits better to these models if it is adapted in the following way. In every replication, the computed values are samples of the probability

TABLE III: Model in Figure 3, $\epsilon = 0.001$

$\widetilde{C}_1$	$\widetilde{C}_2$	$\widetilde{C}_3$	$\widehat{\gamma}_{cie}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_s\}/\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\}$
4–8–12	—	—	6.53E−09	4.60E−20	87
3–7–11	9–13	—	6.38E−09	7.75E−21	516
3–7–11	4–8–12	9–13	6.44E−09	1.28E−21	3125

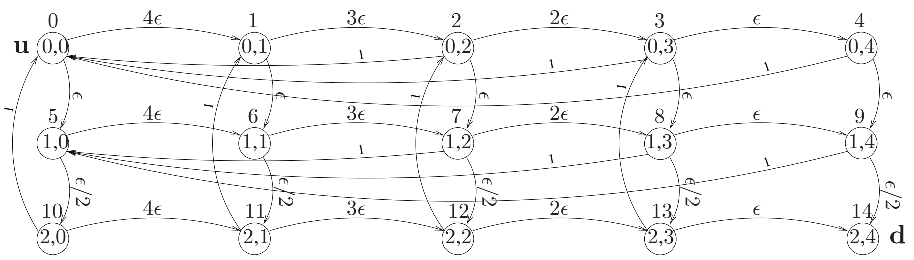


Fig. 3: Continuous time Markov chain used in the experimental variance analysis

of interest, γ, conditioned to the values of the random variable X_C (or to the intermediate states that are reached). In the end, the values of X_C are a function of the path $\pi_{\mathbf{u}}$ followed by the trajectory started at state $\mathbf{u}$. But the value of γ can be conditioned to events related to different functions that take the path $\pi_{\mathbf{u}}$ as their argument. In the following subsections three possible alternatives for these functions are introduced.

A. Forward Steps

For some systems it is possible to detect whether, at every jump, the replications move towards the final state $\mathbf{d}$, or not. In these cases it is possible to make the recursive calls only after moving $D \geq 1$ steps closer to the target state, every time. Proceeding this way, wherever the simulation starts, it must keep moving forwards and backwards until it either comes back to $\mathbf{u}$, or moves D steps forward to $\mathbf{d}$. If $\mathbf{u}$ is reached, the replication terminates; if the simulation moves D steps closer to $\mathbf{d}$, a new replication (recursive call) is launched.

B. Consecutive Failures

In multicomponent systems subject to fails and repairs, every failure produces a forward step, that is a step towards the final state $\mathbf{d}$. For a replication that starts at some state i, there are many ways (paths) to get D steps closer to the target state $\mathbf{d}$. One of them corresponds to the case in which D consecutive failures occur. If the system is composed of more that D components, there will be many different ways in which D consecutive failures may occur, all of them rarer than the case in which the D steps are completed after a zigzag of forward and backward steps. Thus, the indicator random variable I can be conditioned on such a sequence of D consecutive failures.

C. Measure of Rarity

Let $\pi_{i,j}$ be a path that starts at state i and ends at state j, without hitting state $\mathbf{u}$. If this path is composed of the sequence of states i, k, …, l, j, the probability that the simulation goes through it, is: $\mathbb{P}\{\pi_{i,j}\} = p_{ik} \times \ldots \times p_{lj}$, where p_{xy} is the probability of going from state x to the neighbour state y, no matter if this jump is a fail or a repair.

In order to apply *CMIE*, the indicator variable I can be conditioned on the event $\mathbb{P}\{\pi_{i,j}\} \leq B$, where B is a fixed bound. But, as in highly reliable systems most of the probabilities p_{xy} are likely to be low, the values $\mathbb{P}\{\pi_{i,j}\}$ are likely to be extremely low. Therefore, it may be better to apply logarithm as follows: $-\log(\mathbb{P}\{\pi_{i,j}\}) = -\log(p_{ik}) - \ldots - \log(p_{lj})$ and to condition on the event $-\log(\mathbb{P}\{\pi_{i,j}\}) \geq W$ (where $W = -\log(B)$).

TABLE IV: *Tandem computer*, 1st version in [1]

Method	$\widehat{\gamma}_{cie}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\} \times t$
FB	1.33E−06	3.37E−14
SFB	1.27E−06	2.06E−15
SFBP	1.27E−06	2.20E−15
Forward Steps	1.21E−06	4.20E−13
C. Failures	1.19E−06	3.93E−13
M. of Rarity	1.20E−06	5.38E−13

D. Experimental Comparison

The three implementations proposed are now subject to an experimental comparison. All the values —obtained by using the formulas indicated in (12)— are compared to results obtained from published papers.

The model used in the first set of experiments was used by Cancela et. al. in [1]. It is a computer that is composed of a multiprocessor, a dual disk controller, two RAID disk drives, two fans, two power supplies, and one dual interprocessor bus. When a component in a dual fails, the subsystem is reconfigured into a simplex. This tandem computer system requires all subsystems, one fan, and one power supply for it to be *operational*. The failure rates are 5ϵ, 2ϵ, 4ϵ, 0.1ϵ and 3ϵ for the processors, the disk controller, the disks, the fans, the power supplies and the bus respectively, with $\epsilon = 10^{-5}$ failures/hour. There is only one repairman and the repair rates are 30 repairs/hour for all the components, except for the bus, which has repair rate equal to 15 repairs/hour. In the experiments shown in Table IV, the multiprocessor and the disks have two units each, and only one is needed for the system to be working. FB, SFB and SFBP are all *Importance Sampling* methods used in [1]. Table V shows the results obtained for the same system, but with with a four-unit multiprocessor (only one of the four processors is required to have an *operational* system), and with each RAID being composed of 5 drives, only 3 of which are required for the system to be *operational*.

The third system used, also taken from [1], consists of a replicated database in which there are four sites, and each site has a whole copy of the database, on a RAID disk cluster. All clusters are identical, with the same redundancies (7-out-of-9), and with failure rate (for each disk) equal to $\epsilon = 10^{-2}$. There is one repairman per class, and the repair rate is 1. The system is considered *up* if there is at least one copy of the database working. Results are shown in Table VI.

Measure of Rarity is efficient only if failure and repair rates are considerably different. When this is not the case, the

TABLE V: *Tandem computer*, 2nd version in [1]

Method	$\widehat{\gamma}_{cie}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\} \times t$
FB	1.24E−07	1.88E−15
SFB	—	1.57E−16
SFBP	1.25E−07	9.05E−17
Forward Steps	1.19E−07	5.55E−14
C. Failures	1.30E−07	6.17E−14
M. of Rarity	1.24E−07	1.11E−14

TABLE VI: *Replicated database* in [1]

Method	$\widehat{\gamma}_{cie}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\} \times t$
FB	8.54E−13	8.65E−25
SFB	1.16E−12	3.93E−23
SFBP	8.87E−13	2.37E−25
Forward Steps	8.04E−13	4.41E−26
Consecutive Failures	8.10E−13	4.18E−23
Measure of Rarity	—	—

measure of rarity increases significantly at both, failures and repairs and, as a consequence, an increase of such measure is not an indication that the systems is moving towards the target event. In the case of the replicated database, failure and repair rates are, respectively, 10^{-2} and 1. Compared to the rates of the other systems analyzed, these rates are considerably close. This is the reason why *Measure of Rarity* is not computed in Table VI.

In the second set of experiments the models are the ones used by L'Ecuyer et. al. in [18]. In the first case (Example 5 in [18]), the system is composed of two sets of processors with two processors per set, two sets of disk controllers with two controllers per set, and six clusters of disks with four disks per cluster. The failure rates for processors, controllers, and disks are 5×10^{-5}, 2×10^{-5} and 2×10^{-5}, respectively. The repair rate is 1 for each type of component. In each disk cluster, data is replicated, which means that the failure of a single disk does not provoke a system's failure. The system is *operational* if all data is accessible from both processor types, meaning that at least one processor of each type, one controller of each set, and three disks of each cluster are *operational*. Results are shown in Table VII. BFB, SBLR, ZVA(v_0), ZVA(v_1), ZVA(v_2), and ZVA(v_3) are all *Importance Sampling* methods used in [18].

The last example is the one referred to as Example 6 in [18]. The system is composed of 20 types of components numbered from 0 to 19, with 4 components of each type. All repair rates are assumed to be 1, but component's failure rates differ: type–i components have failure rate $\lambda_i = (1+i/10)\epsilon$ for $0 \leq 1 \leq 9$ and $\lambda_i = i\epsilon^2/10$ for $10 \leq 1 \leq 19$, where $\epsilon = 10^{-3}$. The system is *failed* whenever a total of 7 components are *failed*. Results are shown in Table VIII.

All the *CMIE* estimations can be considered in the same order of precision and efficiency of the other methods to which the comparisons has been made.

VIII. CMIE vs. Splitting

If the sets of intermediate states are cuts (between $\mathbf{u}$ and $\mathbf{d}$) in the graph that models the Markov chain, there is a formal equivalence between *CMIE* and *Splitting* [5], [6], [7], [9], [27].

TABLE VII: Example 5 in [18] ($\gamma = 5.60$E−05)

Method	$\widehat{\gamma}_{cie}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\} \times t$
BFB	—	4.93E−07
SBLR	—	1.17E−03
ZVA(v_0)	—	6.21E−11
ZVA(v_1)	—	3.90E−11
ZVA(v_2)	—	4.80E−11
Forward Steps	5.59E−05	3.08E−11
C. Failures	5.51E−05	2.83E−11
M. of Rarity	5.28E−05	9.88E−11

TABLE VIII: Example 6 in [18]

Method	$\widehat{\gamma}_{cie}$	$\widehat{\mathbb{V}}\{\widehat{\gamma}_{cie}\} \times t$
BFB	3.10E−11	9.35E−17
SBLR	—	—
ZVA(v_3)	3.00E−11	1.26E−22
Forward Steps	3.03E−11	1.74E−23
C. Failures	2.93E−11	4.28E−22
M. of Rarity	2.38E−11	8.70E−21

When the set $\widetilde{C}$ is a cut, the *CMIE* estimator takes the form:

$$\widehat{\gamma}_{cie} = \frac{1}{N_1}\sum_{j=1}^{N_1}\left(\sum_{k=1}^{n} I_k^{(j)} \times \frac{1}{N_2}\sum_{i=1}^{N_2} J_k^{(i)}\right) = \frac{1}{N_1 N_2}\sum_{j=1}^{N_1}\sum_{k=1}^{n}\sum_{i=1}^{N_2} I_k^{(j)} J_k^{(i)}. \quad (13)$$

A different analysis on the same model shows that, any path starting at state $\mathbf{u}$ has a probability, say P_1, to reach —any state of— the set $\widetilde{C}$ before coming back to $\mathbf{u}$. In the same way, a path starting from any state in the set $\widetilde{C}$ has a probability, say P_2, to reach state $\mathbf{d}$ before coming back to $\mathbf{u}$. The set $\widetilde{C}$ can be seen as a *bound* or *threshold* in the paths going from $\mathbf{u}$ to $\mathbf{d}$ and, therefore, *Splitting* can be applied in the estimation of γ. This *Splitting* estimation takes the form: $\widehat{\gamma} = \widehat{P}_1 \times \widehat{P}_2$, where $\widehat{P}_1$ and $\widehat{P}_2$ are, respectively, standard estimators of P_1 and P_2, as in any ordinary *Splitting* application. Figure 4 shows part of a set of replications, some of which start at $\mathbf{u}$ and goes forward to $\widetilde{C}$, and some others that start at $\widetilde{C}$ and goes forward to $\mathbf{d}$. According to this approach, the estimators of P_1 and P_2 are:

$$\widehat{P}_1 = \frac{1}{N_1}\sum_{j=1}^{N_1}\sum_{k=1}^{n} I_k^{(j)} \quad \text{and} \quad \widehat{P}_2 = \frac{\sum_{j=1}^{N_1}\sum_{k=1}^{n} I_k^{(j)} \sum_{i=1}^{N_2} J_k^{(i)}}{N_2 \sum_{j=1}^{N_1}\sum_{k=1}^{n} I_k^{(j)}},$$

and the *Splitting* estimator is:

$$\widehat{\gamma}_{spl} = \widehat{P}_1 \times \widehat{P}_2 = \frac{1}{N_1 N_2}\sum_{j=1}^{N_1}\sum_{k=1}^{n} I_k^{(j)} \sum_{i=1}^{N_2} J_k^{(i)} = \widehat{\gamma}_{cie}. \quad (14)$$

This leads to the conclusion that, if the set $\widetilde{C} = \{1, 2, \ldots, n\}$ is a cut in the graph of the Markov chain, *CMIE* and *Splitting* (based on a single level set) produce the same estimation. In other words, *Splitting* with a single level set $\widetilde{C}$ is the particular

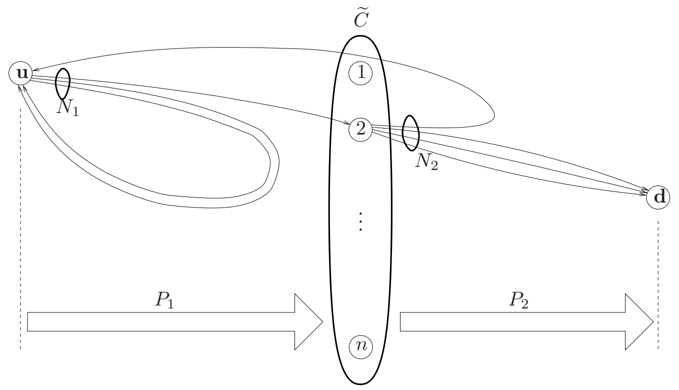


Fig. 4: Some trajectories in a *CMIE* vs. *Splitting* comparison

case of *CMIE* in which the set $\widetilde{C}$ is a cut in the graph of the Markov chain.

In a basic *Splitting* model there are *bounds* or *thresholds* between the initial and the final state, just like the set $\widetilde{C}$ in Figure 4. The consecutive probabilities P_1, $P_2, \ldots$, need to be estimated somehow. One of them is the probability of reaching the final state from the *threshold* that is immediately before. In systems like the ones used in Section VII, there are usually more than one final state scattered through all the Markov chain, some of which may be located between *thresholds*. This requires a particular effort to design a *Splitting* function of importance, while the application of *CMIE* is straightforward.

Another feature that may cause complications in a basic *Splitting* model is failure propagation. Sometimes a particular failure may cause the simultaneous occurrence of a set of other failures, with a given probability. In a basic *Splitting* model this translates into crossing more than one *threshold* simultaneously, what makes necessary to modify the basic approach according to system under analysis. *CMIE* is not affected by failure propagation.

IX. Conclusions and Open Research Lines

This article proposes a *Monte Carlo* method, referred to as Conditional Monte Carlo with Intermediate Estimations (*CMIE*), designed to reduce the variance of the estimator in a context of large and highly reliable Markovian systems.

CMIE was conceived to estimate the probability of visiting the failure state before coming back to the initial state (accepted as the state in which the system is *up*). The application of ordinary *Conditional Monte Carlo* to this type of model requires the knowledge of the exact value of some probabilities in the model. To overcome the fact that this probabilities are unknown, the proposal of *CMIE* is to estimate them, for which it is necessary to launch the method, recursively, from some selected states called intermediate states.

Splitting can be seen as the particular case of *CMIE* in which the events are implicitly defined by means of *thresholds* (cuts) in the state space of the Markov chain. However, the way in which the target probability is recursively computed in *CMIE* is simpler than the *Splitting* algorithm, which needs to determine the probabilities of crossing each *threshold* conditioned to the previous cross, keeping track of the number of times each *threshold* is crossed. Another advantage of *CMIE* over *Splitting*, comes up in systems in which there are more than one target state and/or fault propagation. The presence of more than one target state is a drawback in the determination of *thresholds* (cuts in the graph). Due to the presence of fault propagation, multiple *thresholds* crosses may occur. A particular effort then is required to adapt *Splitting* to these particular settings, whereas the *CMIE* implementations are straightforward and do not differ with respect to ones in which there is only one target state and there is no fault propagation.

CMIE was empirically tested against some other methods taken from the literature. In all cases, the results show that *CMIE* is in the same range of efficiency as the methods to which it was compared, not only in the variance, but also in the precision gain comparison. Some properties of *CMIE* were demonstrated and, besides, its variance was given a closed form. *CMIE* can be easily extended to other types of rare event problems like, for instance, network reliability estimation, either under a static (classic) or a dynamic approach [8].

One possible line of future work is to refine the asymptotic analysis of the behaviour of *CMIE* and to see, for instance, how close or how far it is to have Bounded Relative Error and/or Bounded Normal Approximation. Besides, two important issues to work on are: methods and criteria for the selection of the intermediate states sets and the trade–off between accuracy and execution time.

References

[1] H. Cancela, G. Rubino, and B. Tuffin, "MTTF estimation using Importance Sampling on Markov models." *Monte Carlo Methods and Applications*, vol. 8, no. 4, pp. 321–341, 2002.

[2] A. Goyal, P. Shahabuddin, P. Heidelberger, V. F. Nicola, and P. W. Glynn, "A unified framework for simulating markovian models of highly dependable systems," *IEEE Transactions on Computers*, vol. 41, no. 1, pp. 36–51, 1992.

[3] S. Juneja and P. Shahabuddin, "Fast Simulation of Markov Chains with Small Transition Probabilities," *Management Science*, vol. 47, no. 4, pp. 547–562, Apr. 2001. [Online]. Available: http://dx.doi.org/10.1287/mnsc.47.4.547.9827

[4] G. Rubino and B. Tuffin, *Rare Event Simulation Using Monte Carlo Methods*. Wiley, 2009. [Online]. Available: http://books.google.co.uk/books?id=KoRdR-pSfKsC

[5] M. J. J. Garvels, "The splitting method in rare event simulation," Ph.D. dissertation, Faculty of mathematical Science, University of Twente, The Netherlands, 2000.

[6] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic, "Splitting for rare event simulation: Analysis of simple cases," in *Proceedings of the 1996 Winter Simulation Conference*. San Diego, California: IEEE Computer Society Press, 1996, pp. 302–308.

[7] P. L'Ecuyer, V. Demers, and B. Tuffin, "Rare events, splitting, and quasi-Monte Carlo," *ACM Trans. Model. Comput. Simul.*, vol. 17, no. 2, Apr. 2007. [Online]. Available: http://doi.acm.org/10.1145/1225275.1225280

[8] L. Murray, H. Cancela, and G. Rubino, "A Splitting algorithm for network reliability estimation," *IIE Transactions*, vol. 45, no. 2, pp. 177–189, 2013. [Online]. Available: http://hal.inria.fr/hal-00773319

[9] M. Villén-Altamirano and J. Villén-Altamirano, "RESTART: A method for accelerating rare events simulations," in *Proceedings of the 13th International Teletraffic Congress*. North–Holland, 1991, pp. 71–76.

[10] Z. I. Botev, P. L'Ecuyer, and B. Tuffin, "Markov chain importance sampling with applications to rare event probability estimation," *Statistics and Computing*, vol. 23, no. 2, pp. 271–285, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11222-011-9308-2

[11] P. W. Glynn and D. L. Iglehart, “Importance sampling for stochastic simulations,” *Management Science*, vol. 35, no. 11, pp. 1367–1392, November 1989.

[12] J. Villén-Altamirano, “Importance functions for RESTART simulation of highly–dependable systems,” *Simulation*, vol. 83, pp. 821–828, December 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1363142.1363148

[13] ——, “RESTART simulation of non–Markov consecutive–k–out–of–n: F repairable systems,” *Reliability Engineering & System Safety*, vol. 95, no. 3, pp. 247–254, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6V4T-4XHVH09-2/2/976438347aa6e9fd178b753fe1646bc7

[14] Z. I. Botev and D. P. Kroese, “Efficient Monte Carlo simulation via the generalized splitting method,” *Statistics and Computing*, pp. 1–16, 2010, dOI: 10.1007/s11222-010-9201-4. [Online]. Available: http://dx.doi.org/10.1007/s11222-010-9201-4

[15] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, ser. Wiley Series in Probability and Statistics. Wiley, 2013. [Online]. Available: http://books.google.fr/books?id=Trj9HQ7G8TUC

[16] P. L’Ecuyer, G. Rubino, S. Saggadi, and B. Tuffin, “Approximate zero-variance importance sampling for static network reliability estimation,” *IEEE Transactions on Reliability*, vol. 8, no. 4, pp. 590–604, 2011.

[17] P. L’Ecuyer and B. Tuffin, “Effective approximation of zero-variance simulation in a reliability setting,” in *Proceedings of the 2007 European Simulation and Modeling Conference*. Ghent, Belgium: EUROSIS, 2007, pp. 48–54.

[18] ——, “Approximating Zero-Variance Importance Sampling in a Reliability Setting,” *Annals of Operations Research*, vol. 189, no. 1, pp. 277–297, 2011.

[19] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*, ser. Information Science and Statistics. Springer, 2004. [Online]. Available: http://books.google.fr/books?id=9KZCZIogU4YC

[20] A. Ridder, “Importance sampling simulations of Markovian reliability systems using cross-entropy,” *Annals of Operations Research*, vol. 134, pp. 119–136, 2005.

[21] S. M. Ross, *Simulation, Fourth Edition*. Orlando, FL, USA: Academic Press, Inc., 2006.

[22] J. Zuk and L. Rosenberg, “Rare-event simulation for radar threshold estimation in heavy-tailed sea clutter,” in *IEEE Workshop on Statistical Signal Processing, SSP 2014, Gold Coast, Australia, June 29 - July 2, 2014*, 2014, pp. 436–439. [Online]. Available: http://dx.doi.org/10.1109/SSP.2014.6884669

[23] S. Asmussen and D. Kortschak, “Error rates and improved algorithms for rare event simulation with heavy Weibull tails,” *Methodology and Computing in Applied Probability*, pp. 1–21, Jan. 2013.

[24] J. Blanchet, J. Li, and M. K. Nakayama, “A conditional monte carlo method for estimating the failure probability of a distribution network with random demands,” in *Proceedings of the Winter Simulation Conference*, ser. WSC ’11. Winter Simulation Conference, 2011, pp. 3837–3848. [Online]. Available: http://dl.acm.org/citation.cfm?id=2431518.2431974

[25] J. Chan and D. Kroese, “Rare-event probability estimation with conditional monte carlo,” *Annals of Operations Research*, vol. 189, no. 1, pp. 43–61, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10479-009-0539-y

[26] L. Murray, “New variance reduction methods in Monte Carlo rare event simulation,” Ph.D. dissertation, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 2014.

[27] P. L’Ecuyer, F. Le Gland, P. Lezaud, and B. Tuffin, “Splitting techniques,” in *Rare Event Simulation using Monte Carlo Methods*, G. Rubino and B. Tuffin, Eds. John Wiley & Sons, 2009, ch. 3, pp. 39–61.

Discrete Choquet integral based method for criteria synergy determination

Rubén Bernal – Marcelo Karanik
National Technological University
Chaco, Argentine
rbernal@frre.utn.edu.ar, marcelo@frre.utn.edu.ar

José Ignacio Peláez Sánchez
Department of Languages and Computer Sciences
University of Malaga
Málaga, Spain
jignacio@lcc.uma.es

Estela del Rocio Yanez Benavides
University of Guayaquil
Ecuador
estela.yanezb@ug.edu.ec

Abstract—**In multi-criteria decision-making processes, a set of criteria that can act independently or have some kind of relation between them are considered. When this latter appears, the process cannot be a simple problem due to complexity to model such synergy relations. To deal with these issues, a Choquet integral based method has been developed to determine the most appropriate ranking of alternatives. Therefore, a fuzzy measure that models considerations of an expert in the problem domain must be identified. The proposed model is a novel and efficient algorithm for determining a non-additive fuzzy measure guided by linguistic attributes. Finally, an illustrative example and a comparison with other aggregation operators is presented.**

Keywords—Fuzzy Measure; Choquet Integral; Multiple Criteria Decision Analysis.

I. Introduction

Decision-making (DM) is a very important activity related to choosing the most adequate option among the ones available. Multi-criteria decision-making (MCDM) process uses several techniques to select the most appropriate alternative in order to solve complex problems. This alternative is selected from the alternatives set evaluated with respect to several criteria or attributes. It can become extremely complex, especially when uncertainty environments, such as risk or imprecise data situations, are involved.

In many situations, adequateness ranking of alternatives can be wrong if dependence between criteria is not considered.

Clearly, this type of decision process involves evaluating several alternatives based on multiple criteria with dependence relations between them. This dependence generates groups that have more or less weight (importance) than the sum of individual ones (positive or negative interaction or synergy) [1]. To model this phenomenon, the weight vector of the weighted arithmetic mean can be replaced by a function of non-additive sets (on evaluation criteria set) called fuzzy measure. Thereby, it is possible to define a weight for each criterion and each subset of criteria. Many examples in the literature deal with this trouble; e.g., consumers preference analysis [2], estimations in project management [3], hotel selection preferences of Hong Kong inbound travelers [4], valuation of residential properties [5].

In problems with independent criteria, i.e. when there is no relation between them, the importance of the groups of criteria is additive. However, for many practical applications, the individual criterion has some interaction with other/s and traditional method/s (such as weighted arithmetic mean or OWA operators) are not adequate to deal with this situation [6].

There are different models to determine fuzzy measure with a lot of techniques such as genetic algorithms, gradient descent algorithms, neural networks, particle swarm algorithm and even, by requiring some type of information from experts. Grabisch introduced a gradient descendent algorithm with constraints for identifying fuzzy measure applied to pattern recognition [7]. In [1] two approaches were presented (based on minimization of squared error and constraint satisfaction). Wang proposed a speculative algorithm to extract a fuzzy measure from sample data [8]. Other researchers determine fuzzy measures from sample data with genetic algorithms [9] and use random generation algorithms as a special case of generating points in an order polytope [10]. In an effort to generate methods or techniques to learn distinct fuzzy measures, a particle swarm algorithm to determine a type of general fuzzy measures from data is proposed [11].

All the methods mentioned above can determine a fuzzy measure based on sample data. Complementary to this class of methods, other techniques that require some information from different experts are proposed. For example, in [12], the decision maker must identify the weights and interaction degrees among criteria using a labeled diamond.

The main purpose of this paper is to propose a method (based on discrete Choquet integral) to establish the importance of individual criterion and the possible groups, when dependence exists. A computing with words method is used to obtain that fuzzy measure based on knowledge of different experts. Linguistic labels [13] [14] [15] and preference relations to establish the importance (weight) of each criterion and groups of them are used. In this proposed method, the experts are not required to boast a high level of accuracy to prioritize criteria, but a more flexible way to express their assessment is allowed. The discrete Choquet integral regarding non-additive (or fuzzy) measures, is an aggregation operator suitable for modeling the phenomenon above exposed [16].

The structure of this paper is as follows: In Section II some basic definitions such as fuzzy measure, Choquet integral and general considerations are presented. In Section III, a new model for constructing a fuzzy measure is described, including an illustrative example. In Section IV the results of the model, WAM and OWA operators are compared. Finally, concluding remarks are exposed in Sections V.

II. Preliminary Definitions

A. *Aggregation functions*

Aggregation functions are special real functions with inputs from a subdomain $\mathbb{I}$ of the extended real line. The basic feature of all aggregation functions is their non-decreasing, monotonicity and boundary conditions. Increasing of input values cannot decrease the output values and they are aggregated in the same scale of input values, respectively.

Formally [17], an *aggregation function* is a function $A^{(n)}: \mathbb{I}^n \to \mathbb{I}$ that:

i. is non-decreasing (on each variable);
ii. fulfills the boundary conditions;

$$\inf_{x\in\mathbb{I}^n} A^{(n)}(\boldsymbol{x}) = \inf \mathbb{I} \quad \text{and} \quad \sup_{x\in\mathbb{I}^n} A^{(n)}(\boldsymbol{x}) = \sup \mathbb{I} \qquad (1)$$

$$\text{If } \mathbb{I} = [a,b],\ A^{(n)}(\mathbf{a}) = a \text{ and } A^{(n)}(\mathbf{b}) = b \qquad (2)$$
$$\text{where } \mathbf{a} = (a, \dots, a) \text{ and } \mathbf{b} = (b, \dots, b)$$

iii. for all $x \in \mathbb{I}$

$$A^{(1)}(x) = x \qquad (3)$$

The integer n represents the cardinality of the aggregation function, i.e. the number of its variables. Examples of basic aggregation functions are the arithmetic mean, geometric mean, minimum function, maximum function, OWA [18], MA-OWA [19], etc.

The integral concept (defined with respect to a measure) is another generalization of aggregation function. Fuzzy measures and fuzzy integrals have been applied to many fields, such as MCDM because they can be considered as an aggregation operator that currently constitutes a key research topic.

B. *Fuzzy measure*

Let N a classical set and let $\mathcal{A}$ a σ-algebra of subsets of N; a fuzzy measure [1] on N is a monotonic set function $\mu: \mathcal{A} \to [0,1]$ satisfying:

$$\mu(\emptyset) = 0; \qquad \mu(N) = 1 \qquad (4)$$
$$\mu(A) \le \mu(B)\ \forall\, A, B \in \mathcal{A} \text{ such that } A \subseteq B \qquad (5)$$

A fuzzy measure assigns weight of importance to all possible subsets of criteria from a given criteria set. Thus, in addition to the assignment of weights for individuals, weights assigned to any combination of criteria are also defined. A fuzzy measure μ is *additive* if $\mu(A \cup B) = \mu(A) + \mu(B)$ whenever $A \cap B = \emptyset$.

Generally, in problems of MCDM, criteria used to evaluate a set of alternatives have some kind of interaction such as correlation, substitutiveness/complementarity and preferential dependence. These relations are usually very complex and identifying them is difficult [20]. Often, they are modeled with sub-additive or super-additive fuzzy measures. A positive (direct) correlation between criteria i and j must be modeled by $\mu(ij) < \mu(i) + \mu(j)$ which expresses negative interaction or negative synergy; that is, "the marginal contribution of j to every combination of criteria that contains i is strictly lesser than the marginal contribution of j to the same combination when i is excluded" [20].

A negative (inverse) correlation between criteria i and j must be modeled by $\mu(ij) > \mu(i) + \mu(j)$ which expresses positive interaction or positive synergy; that is, "high partial scores along i usually imply low partial scores along j and vice versa. Simultaneous satisfaction of both criteria is rather uncommon; therefore, the alternatives that present such a satisfaction profile should be favored" [20].

C. *λ-Fuzzy measure*

A λ-fuzzy measure [21] is a fuzzy measure μ that for all $A, B \in N, A \cap B = \emptyset$ satisfies:

$$\mu(A \cup B) = \mu(A) + \mu(B) + \lambda\mu(A)\mu(B) \qquad (6)$$

where $\lambda \in (-1, \infty)$.

Varying λ conveniently according to the information provided by the expert, it is possible to determine a fuzzy measure to model such preferences. It measures the criteria interaction intensity and varies (proportionally) with the number of linguistic labels that the expert can distinguish. When the value of λ is selected, boundary and monotonicity conditions must be taken into account. Specifically:

i. Negative interaction:
$$\text{If } \lambda \in (-1, 0),\ \mu(A \cup B) < \mu(A) + \mu(B) \qquad (7)$$
ii. Positive interaction:
$$\text{If } \lambda \in (0, \infty),\ \mu(A \cup B) > \mu(A) + \mu(B) \qquad (8)$$
iii. No interaction:
$$\text{If } \lambda = 0,\ \mu(A \cup B) = \mu(A) + \mu(B) \qquad (9)$$

where $A, B \in N$.

In (9) the fuzzy measure behaves like the weight arithmetic mean, while in (7) and (8) negative and positive synergy is, respectively, observed.

Consider two criteria c_1 and c_2, used to evaluate and to rank several alternatives. If one of these alternatives satisfies only one single criterion, the decision-maker may consider that it is equally worthless as if it satisfies none. In this case c_1 and c_2 act conjunctively, so that both can be satisfied (positive interaction). This implies that the importance of a single criterion for the decision is almost zero while that of the pair is highly important. The criteria can be said to be complementary ($\lambda > 0$).

On the contrary, if one alternative satisfies only one single criterion, e.g., c_1, the decision-maker may consider that it is equally good as another that satisfies both. In this case, c_1 and c_2 act disjunctively, and it is sufficient that one of them is satisfied (negative interaction). The union of criteria has no additional benefit, and the importance of the pair is almost the same as the importance of the single criteria. In such case, they are said to be redundant ($\lambda < 0$).

If the decision-maker thinks that the importance of the pair is approximately the sum of the individual importance of criteria; they act independently and there is no interaction between them [16].

Fig. 1 shows the way to determine the value of $\mu(A \cup B)$ considering the variation of λ and the level of satisfaction (score) for each criterion.

As mentioned above, to maintain the monotonicity of the fuzzy measure, the value of λ must be in the range $[\lambda_{min}, \lambda_{max}]$, where

$$\lambda_{min} = \frac{max[\mu(A), \mu(B)] - \left(\mu(A) + \mu(B)\right)}{\mu(A)\mu(B)} \qquad (10)$$
$$A, B \in \mathcal{P}(N)$$

and[1]

$$\lambda_{max} = abs(\lambda_{min}) \qquad (11)$$

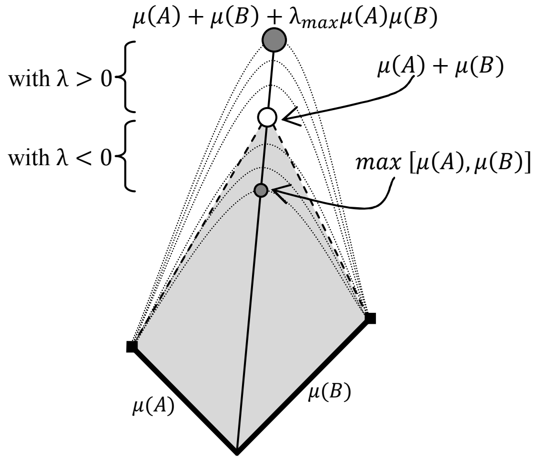


Fig. 1. $\mu(A \cup B)$ Depending on the variation of λ

[1] To keep equidistance to 0.

D. Choquet integral

The Choquet integral was introduced in 1954 by Gustave Choquet [22]. It can be considered as an aggregation operator, which generalizes the weighted arithmetic mean and is able to represent dependence between criteria in many situations.

Let μ be a fuzzy measure on $N = \{c_1, \dots, c_n\}$ and $\mathbf{x} = (x_1, x_2, \dots, x_n) \in [0, \infty)^n$, the (discrete) Choquet integral of $\mathbf{x}$ with respect of μ is defined by:

$$\mathbb{C}_\mu(\mathbf{x}) = \sum_{i=1}^{n} x_{(i)}\left[\mu\left(A_{(i)}\right) - \mu\left(A_{(i+1)}\right)\right] \qquad (12)$$

where $x_{(\cdot)}$ indicates a permutation such that $x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$; $A_{(i)} = \{i, \dots, n\}$; $A_{(n+1)} = \varnothing$; $\mu\left(A_{(n+1)}\right) = 0$.

The Choquet integral has very good properties of aggregation [20]. For instance, *increasing monotonicity* (non-negative response to any increase of the arguments); *idempotence* (comprised between min and max); *stability* with respect to the same interval scales and coincides with the weighted arithmetic mean when the fuzzy measure is additive (see e.g. [20], [17]).

E. Linguistic Labels

Information can be expressed in a quantitative or qualitative way. Normally, it is assumed that dealing with quantities is easier than dealing with qualities. Sometimes this is not true, specifically when perceptions over quantities are required. Humans have an intrinsic capability to perform a wide variety of physical and mental tasks without any measurements and any computations. Indeed, this capability allows decisions under uncertainty. In order to choose the best alternative available, decision-makers manipulate different kinds of perceptions. This process involves human recognition, decision and execution processes [23].

In order to express perceptions, humans use words representing estimates. These words play the role of labels of perceptions. More generally, perceptions are expressed using several linguistic computational models [24].

Linguistic labels are mathematical functions used for assigning values to concepts. These values belong to a numerical scale previously defined. The number of elements in the term set establishes the granularity of the uncertainty (see [25]). The decision-maker uses these labels without knowledge of what values they represent. The use of several linguistic labels sets gives more flexibility, i.e., one expert may choose values from 1 to 5 as his evaluation [{none, low, medium, high, perfect}], while another expert may use 7 labels [{none, very low, low, medium, high, very high, perfect}] to make a more detailed opinion. Clearly, using less linguistic labels simplifies the information representation and by using more linguistic labels, the correctness of the decision is improved, thus increasing the computational complexity.

In this work, linguistic labels are used for two purposes: (1) to assign the importance level for each criterion and (2) to assign synergy degree of criteria groups. For both, the expertise *level* defines the label granularity.

III. Proposed Model

An interesting challenge arising in the practical use of fuzzy measures is to determine a fuzzy measure model that deals with concrete situations. A problem involving n criteria, requires $2^n - 2$ coefficients[2] in order to define a fuzzy measure μ. Clearly, to provide this amount of information is very hard and usually the meaning of the values $\mu(A)$ is not sufficiently clear for the experts.

In the proposed method it is not necessary for the experts to provide the synergy degree of all combination of criteria. Only the obvious relationships are necessary; the rest is considered additive. With these considerations, the proposed model computes an adequate fuzzy measure and the expert can regulate the interaction degree by specifying different linguistic labels.

Let´s consider $A = \{a_1, \dots, a_k\}$ a set of alternatives to be evaluated with respect to a set of n criteria $N = \{c_1, \dots, c_n\}$. Each alternative $a_i \in A$, has a profile $x^a = (x_1^a, x_2^a, \dots, x_n^a) \in \mathbb{R}^n$, where x_i^a is a partial valuation of a w.r.t. the criterion c_i. From x^a it is possible to calculate an overall measure $M(x^a)$ for each alternative by an aggregation operator $M: \mathbb{R}^n \to \mathbb{R}$. Also consider A, B, C ... the subsets $\in \mathcal{P}(N)$ and $\mu(A), \mu(B)$, ... their weights.

To identify the weight of each subset of criteria, two steps are considered: (a) *Determine the importance of criteria:* each criterion will be evaluated individually to determine its relative importance with respect to the objective. Each $\mu(C_i)$, with $C_i \in N$ and $1 \le i \le n$ will be determined. (b) *Determine the weight of interacting criteria:* the experts will be asked for the sign and degree of interacting criteria in order to determine the weight of each coalition. Linguistic labels are used to establish the importance (weight) of each group of criteria. Each $\mu(C_i), C_i \in \mathcal{P}(N) - N$ will be determined, where $\mathcal{P}(N)$ is the power set of N.

It aims to build a fuzzy measure $\mu: \mathcal{P}(N) \to [0,1]$ such that:
$\mu(\emptyset) = 0$;
$\mu(N) = 1$;
$\mu(A) \le \mu(B)$ if $A \subseteq B, \forall A, B \in \mathcal{P}(N)$

Let´s also consider:

- a set of labels $V = \{v_1, \dots, v_j\}$; each v_i is used to provide the relative importance of each criteria w.r.t. the objective being evaluated;[3]
- a set of labels $W = \{w_1, \dots, w_j\}$; each w_i is used to provide the synergy degree between criteria including the null label;[4]
- $el \in [1..3]$ that characterizes each expert according to its expertise level;
- a pair (el, lq) indicating the number of different linguistic labels of V that the expert can distinguish; the values of lq will be set depending on the expertise level el.

  if $(el, lq) = (1, 3)$:
  $V = \{$*low, medium, high*$\}$ and
  $W = \{$*null, moderate, extreme*$\}$;
  if $(el, lq) = (2, 5)$:
  $V = \left\{\begin{matrix}\textit{very low, low ,medium,} \\ \textit{high, very high}\end{matrix}\right\}$ and
  $W = \left\{\begin{matrix}\textit{null, weak, moderate,} \\ \textit{strong, extreme}\end{matrix}\right\}$;
  if $(el, lq) = (3, 7)$:
  $V = \left\{\begin{matrix}\textit{lowest, very low, low, medium, high,} \\ \textit{very high, highest}\end{matrix}\right\}$ and
  $W = \left\{\begin{matrix}\textit{null, very weak, weak, moderate,} \\ \textit{strong, very strong, extreme}\end{matrix}\right\}$.
- a pair (e, lq) with the characterization profile of the expert e.

Algorithm

```
Determine a fuzzy measure $\mu$.
1. Determine the number of labels $lq$ that the expert $e$ can distinguish.
2. Determine the weight of each criterion individually:
    a. Associate a label to each criterion on $N$.
    b. Compute the corresponding weight for each criterion.
3. For each $C \in \mathcal{P}(N)$ (if coalitions between criteria appear) do:
    Begin
    a. Determine the weight of coalition: for $C \in \mathcal{P}(N)$ with cardinality $1 < i \le n$:
       $\mu(C) = \mu(A) + \mu(B) + \lambda\mu(A)\mu(B)$
       where
       $\mu(A) = max\left(\mu(A_j)\right);\ A_j \subseteq C;\ |A_j| = i - 1$
       $\mu(B) = \mu(C - A)$
       $\lambda \in [\lambda_{min}, \lambda_{max}]$
       with
       $\lambda_{min} = \frac{max[\mu(A), \mu(B)] - (\mu(A) + \mu(B))}{\mu(A)\mu(B)}$;
       $\lambda_{max} = abs(\lambda_{min})$
       $A, B \in \mathcal{P}(N)$
    b. Establish a mapping between each linguistic label (of $W$) and $\mathbb{N}_0$, determine the synergy sign (positive or negative) and compute the $\lambda$ value:
       for (-) synergy: $\lambda = -n\,\Delta_\lambda;\ 0 \le n \le (lq - 1)$
       for (+) synergy: $\lambda = n\,\Delta_\lambda;\ 0 \le n \le (lq - 1)$
       with: $\Delta_\lambda = \frac{\lambda_{max}}{lq-1}$
    End
```

[2] The coefficients of $\emptyset$ and N are determined by definition: $\mu(\emptyset) = 0$; $\mu(N) = 1$
[3] V does not include the Null label because a criterion with weight equal to 0 should not be taken into account for the decision process.
[4] W includes the Null label because there may exist criteria without interaction.

4. Normalize the values to $[0..1]$ to obtain μ:
$$\mu(A_i) = \frac{\mu(A_i)}{max\left(\mu(B)\right)}; \quad \forall A_i, B \in \mathcal{P}(N)$$
5. For each alternative $a_i \in A$, determine the profile:
$$x^a = (x_1^a, x_2^a, \dots, x_n^a) \in \mathbb{R}^n .$$
6. Aggregate results:
    a. For each alternative $a_i \in A$, compute the discrete CI $\mathbb{C}_\mu(a_i)$ w.r.t. fuzzy measure μ.

IV. Illustrative Example

A. *Definition*

In this section, a problem of evaluation of people in their job performance according to four criteria is presented.

The evaluation criteria are defined accordingly: if the person is extrovert, interpersonal relationships with peers, technical skills, and readiness to transmit their knowledge.

c_1 (E) *Extroversion*: it refers to a person who is friendly and outgoing. Each candidate is rated on a scale of 1 to 10, evaluating characteristics such as friendliness, activities, social influence, positive emotions, etc.

c_2 (IR) *Interpersonal Relationships*: it is a close association between two or more people. Love, solidarity, regular business interactions or some other activity may influence in this type of relations.

c_3 (TS) *Technical Skills*: it is the level of knowledge and skill relating to a specific field. Each candidate is rated on a scale of 1 to 10 according to several theoretical evaluations.

c_4 (KT) *Knowledge Transfer*: it is the practical problem of transferring knowledge from one person to another; seeks to organize, create, capture or distribute knowledge to ensure its availability for future users.

Considerations and point of view of the expert are summarized in Sections B and C.

B. *Importance of criteria*

For each criterion mentioned in section A, the relative importance with respect to the objective to be tested was determined. An expert with $el = 2$ is considered, thus five linguistic labels are used. With $(el, lq) = (2,5)$:

$$V \to \mathbb{N} = \left\{ \begin{matrix} \textit{very low} \to 1, \textit{low} \to 2, \textit{medium} \to 3, \\ \textit{high} \to 4, \textit{very high} \to 5 \end{matrix} \right\}$$

Technical skills and how knowledge is transferred are the most important criteria with respect to personality. However, a candidate with a balanced profile between technical knowledge and readiness to transmit his knowledge will be taken into account. The following assignment of weights was made to the proposed criteria:

c_1: Extroversion (E)	*Low*	0.143
c_2: Interpersonal Relationships (IR)	*Medium*	0.214
c_3: Technical Skills (TS)	*Very High*	0.357
c_4: Knowledge Transfer (KT)	*High*	0.286

which allows to obtain the weight vector:

$$w = (0.143, 0.214, 0.357, 0.286).$$

Three candidates, whose performance on each criterion is given in Table I, are considered.

TABLE I. Performance of the Different Candidates

Candidate	E	IR	TS	KT
Cand1	6	7	8	2
Cand2	7	8	8	1
Cand3	7	8	5	4

C. *Weight of interacting criteria*

If a person is outgoing, they usually have good interpersonal relationships and vice versa. These two criteria have some degree of redundancy (overlap); therefore, the global evaluation will be overestimated (respectively underestimated) by extroverts (respectively introverts).

This phenomenon between Extroversion (c_1) and Interpersonal Relationship (c_2), can be easily overcome by using a fuzzy measure μ and the Choquet Integral $\mathbb{C}_\mu$ such that: $\mu(E, IR) < \mu(E) + \mu(IR)$ that expresses a negative synergy.

In the same way, due to the fact that most people have deeply ingrained the win/lose mentality from birth and think in terms of either/or in competitive situations, Technical Skills (c_3) and Knowledge Transfer (c_4) are negatively correlated. That is, high partial scores along c_3 usually imply low partial values along c_4 and vice versa. The simultaneous satisfaction of both criteria is rather uncommon (it involves mutual learning, mutual benefits and a win/win mentality); therefore, the alternatives that present such a satisfaction profile should be favored, i.e.: $\mu(TS, KT) > \mu(TS) + \mu(KT)$ that expresses a positive synergy.

Mapping of synergy scale is defined as follows:

$$W \to \mathbb{N}_0 = \left\{ \begin{matrix} \textit{null} \to 0, \textit{weak} \to 1, \textit{moderate} \to 2, \\ \textit{strong} \to 3, \textit{extreme} \to 4 \end{matrix} \right\}$$

The synergy degree assigned by the expert is "*strong*$(-)$" for the coalition $\{E, IR\}$ and "*extreme*$(+)$" for the coalition $\{TS, KT\}$.

With these considerations and by using the algorithm described above (steps 3 and 4), a fuzzy measure μ was automatically constructed (Table II).

D. *Results*

Using the Choquet Integral as aggregation operator (step 6 of algorithm) with respect to μ, global scores was obtained.

The analysis of the results is performed by comparing the results obtained by using the WAM, OWA and Choquet Integral operators. Table III summarizes the obtained results.

TABLE II. FUZZY MEASURE μ

Coalitions	Weights
{E}	0.120
{IR}	0.180
{TS}	0.300
{KT}	0.240
{E, IR}	0.240
{E, TS}	0.420
{E, KT}	0.360
{IR, TS}	0.480
{IR, KT}	0.420
{TS, KT}	0.700
{E, IR, TS}	0.600
{E, IR, KT}	0.540
{E, TS, KT}	0.820
{IR, TS, KT}	0.880
{E, IR, TS, KT}	1.000

TABLE III. RANKING OF ALTERNATIVES OBTAINED WITH WAM, OWA AND CHOQUET INTEGRAL

Candidate	E	IR	TS	KT
Cand1	6	7	8	2
Cand2	7	8	8	1
Cand3	7	8	5	4

$\mathbf{WAM_w}$	Rk	$\mathbf{OWA_w}$	Rk	$\mathbb{C}_\mu$	Rk
5.7857	2	6.3571	3	5.1800	2
5.8571	1	6.7857	1	5.0800	3
5.6429	3	6.4286	2	5.2000	1

Using the Weight Arithmetic Mean and OWA operator, the best candidate is candidate 2, because, on average, he/she has the best partial values. The optimistic OWA operator used in the example, improves results and allows the candidate 3 goes up in the ranking. However, results are not satisfactory because there is no operator that takes into account the interacting criteria.

On the contrary, by using the CI, candidates 1 and 2 lose their positions because they are the most unbalanced for the criteria c_3 and c_4 . Candidate 3 goes up two positions (comparing with WAM) and one position (comparing with OWA operator), who appears to be the best ranked when correlated criteria were considered (due to balanced scoring in all criteria). In addition, there was a decrease in the global score of all alternatives because criteria c_1 and c_2 have negative synergy. This avoids overestimates in high scores.

V. CONCLUSIONS

A new model to identify non-additive fuzzy measures based on linguistic labels was presented in this article. The proposed model allows the decision-maker to be focused on individual criteria and their relations, reducing the judgment process notably. Decision-maker categorizes criteria groups and the interaction type between them using linguistic labels. Then, the level of positive (or negative) synergy for every criteria group is calculated using λ-fuzzy measures. The proposed linguistic labels are very simple to use and they provide an adequate abstraction level.

λ-variability provides accurate values of fuzzy measure to express the strength of criteria coalition. This point is relevant because the decision-maker's opinion is properly represented.

By using the Choquet integral it is possible to replace the weighed arithmetic mean to aggregate dependent decision criteria obtaining more accurate results. By comparing the performance of the OWA operator and the Choquet integral (using the proposed method) as aggregation operator, better results can be observed. Clearly, WAM and OWA operators cannot represent this type of situations.

FUTURE WORK

This work is part of a research project related to the study of Decision Support Systems (DSS) and it contains the initial results of the proposed criteria coalition model. Currently, some proposed model improvements are under development (as the use of other mechanisms to obtain fuzzy measures).

In addition, the research team is working on future application of this proposed model to the assessment and evaluation of investment projects for financial organizations.

Furthermore, some interesting topics have been analyzed in order to extend the use of this model with opinion of multiple experts.

ACKNOWLEDGMENT

This work has been supported by the Project TIN 2011-26046 of the Science and Innovation Minister (Spain), the Project UTN2106 of National Technological University (Argentine) and the PROMETEO Project (Ecuador).

REFERENCES

[1] M. Grabisch and M. Roubens, "Application of the Choquet integral in multicriteria decision making," in *Fuzzy Measures and Integrals: Theory and Applications*, M. Grabisch, T. Murofushi, and M. Sugeno, Eds. Heidelberg: Physica Verlag, 2000, pp. 348–374.

[2] H. Q. Vu, G. Li, and G. Beliakov, "A fuzzy decision support method for customer preferences analysis based on Choquet Integral," in *FUZZ-IEEE 2012, IEEE International Conference on Fuzzy Systems, Brisbane, Australia, June 10-15, 2012, Proceedings.*, 2012, pp. 1–8.

[3] A. Bonetti, S. Bortot, M. Fedrizzi, R. A. M. Pereira, and A. Molinari, "Modelling group processes and effort estimation in project management using the Choquet integral: an MCDM approach," *Expert Syst Appl*, vol. 39, no. 18, pp. 13366–13375, 2012.

[4] G. Li, R. Law, H. Q. Vu, and J. Rong, "Discovering the hotel selection preferences of Hong Kong inbound travelers using the Choquet Integral," *Tour. Manag.*, vol. 36, no. 0, pp. 321 – 330, 2013.

[5] L. F. A. M. Gomes, M. A. S. Machado, F. F. da Costa, and L. A. D. Rangel, "Criteria interactions in multiple criteria decision aiding: A Choquet formulation for the TODIM method," in *Proceedings of the*

First International Conference on Information Technology and Quantitative Management, ITQM 2013, Dushu Lake Hotel, Sushou, China, 16-18 May, 2013, 2013, pp. 324–331.

[6] J.-L. Marichal and M. Roubens, “Determination of weights of interacting criteria from a reference set,” *Eur. J. Oper. Res.*, vol. 124, no. 3, pp. 641 – 650, 2000.

[7] M. Grabisch, “A new algorithm for identifying fuzzy measures and its application to pattern recognition,” in *Int. Joint Conf. of the 4th IEEE Int. Conf. on Fuzzy Systems and the 2nd Int. Fuzzy Engineering Symposium*, 1995, pp. 145–150.

[8] X. Wang, A. F. G. Contreras, M. Ceberio, C. D. Hoyo, and L. C. Gutierrez, “A speculative algorithm to extract fuzzy measures from sample data.,” in *FUZZ-IEEE*, 2012, pp. 1–8.

[9] E. F. Combarro and P. Miranda, “Identification of fuzzy measures from sample data with genetic algorithms,” *Comput. Oper. Res.*, vol. 33, no. 10, pp. 3046 – 3066, 2006.

[10] E. F. Combarro, I. Díaz, and P. Miranda, “On random generation of fuzzy measures,” *Fuzzy Sets Syst.*, vol. 228, pp. 64–77, 2013.

[11] X.-Z. Wang, Y.-L. He, L.-C. Dong, and H.-Y. Zhao, “Particle swarm optimization for determining fuzzy measures from data,” *Inf. Sci.*, vol. 181, no. 19, pp. 4230 – 4252, 2011.

[12] E. Takahagi, “A Fuzzy Measure Identification Method by Diamond Pairwise Comparisons: AHP Scales and Grabish’s Graphical Interpretation,” in *Knowledge-Based Intelligent Information and Engineering Systems*, vol. 4694, B. Apolloni, R. Howlett, and L. Jain, Eds. Springer Berlin Heidelberg, 2007, pp. 316–324.

[13] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning-III,” *Inf. Sci.*, vol. 9, no. 1, pp. 43 – 80, 1975.

[14] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning—I,” *Inf. Sci.*, vol. 8, no. 3, pp. 199 – 249, 1975.

[15] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning—II,” *Inf. Sci.*, vol. 8, no. 4, pp. 301 – 357, 1975.

[16] M. Grabisch and C. Labreuche, “Fuzzy measures and integrals in MCDA,” in *Multiple Criteria Decision Analysis: State of the Art Surveys*, J. Figueira, S. Greco, and M. Ehrgott, Eds. Boston, Dordrecht, London: Springer Verlag, 2005, pp. 563–608.

[17] M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap, “Aggregation functions: means,” *Inf. Sci.*, vol. 181, no. 1, pp. 1 – 22, 2011.

[18] R. R. Yager, “On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking,” *IEEE Trans Syst Man Cybern*, vol. 18, no. 1, pp. 183–190, Jan. 1988.

[19] J. I. Peláez and J. M. Doña, “Majority additive-ordered weighting averaging: A new neat ordered weighting averaging operator based on the majority process.,” *Int J Intell Syst*, vol. 18, no. 4, pp. 469–481, 2003.

[20] J.-L. Marichal, “An axiomatic approach of the discrete Choquet integral as a tool to aggregate interacting criteria.,” *IEEE T Fuzzy Syst.*, vol. 8, no. 6, pp. 800–807, 2000.

[21] M. Sugeno, “Theory of fuzzy integrals and its applications,” Tokyo Institute of Technology, 1974.

[22] G. Choquet, “Theory of capacities,” *Ann. Inst. Fourier*, vol. 5, pp. 131–295, 1954.

[23] L. A. Zadeh, “From computing with numbers to computing with words — from manipulation of measurements to manipulation of perceptions,” in *Logic, Thought and Action*, vol. 2, D. Vanderveken, Ed. Springer Netherlands, 2005, pp. 507–544.

[24] F. Herrera, S. Alonso, F. Chiclana, and E. Herrera-Viedma, “Computing with Words in Decision Making: Foundations, Trends and Prospects,” *Fuzzy Optim. Decis. Mak.*, vol. 8, no. 4, pp. 337–364, Dec. 2009.

[25] E. Herrera-Viedma, F. Herrera, L. Martínez, and F. Herrera, “A fusion approach for managing multi-granularity linguistic term sets in decision making,” in *Sets in Decision Making, Fuzzy Sets and Systems*, 2000, pp. 43–58.

Case-based Reasoning for Web Service Discovery and Selection

Alan De Renzis*[1], Martin Garriga*,
Andres Flores* and Alejandra Cechich

GIISCo Research Group, Faculty of Informatics
UNComa University, Neuquen, Argentina
[1]*corresponding author:* alanderenzis@fi.uncoma.edu.ar

Alejandro Zunino*

ISISTAN Research Institute, UNICEN University
Tandil, Buenos Aires, Argentina
**Also Consejo Nacional de Investigaciones Científicas*
y Técnicas (CONICET)

***Abstract*—Web Service discovery and selection deal with the retrieval of the most suitable Web Service, given a required functionality. Addressing an effective solution remains difficult when only functional descriptions of services are available. In this paper, we propose a solution by applying Case-based Reasoning, in which the resemblance between a pair of cases is quantified through a similarity function. We show the feasibility of applying Case-based Reasoning for Web Service discovery and selection, by introducing a novel case representation, learning heuristics and three different similarity functions. We also experimentally validate our proposal with a dataset of 62 real-life Web Services, achieving competitive values in terms of well-known Information Retrieval metrics.**

***Index Terms*—Web services, Service Selection, Service Discovery, Case-based Reasoning, Service Oriented Application**

I. Introduction

Service-Oriented Computing (SOC) has seen an ever increasing adoption by providing support for building distributed, inter-organizational applications in heterogeneous environments [1]. Mostly, the software industry has adopted SOC by using Web Service technologies. A Web Service is a program with a well-defined interface that can be located, published, and invoked by using standard Web protocols [2].

However, a broadly use of the SOC paradigm requires efficient approaches to allow service discovery, selection, integration and consumption from within applications [3]. Currently, developers are required to manually search for suitable services to then provide the adequate "glue-code" for their assembly into a service-oriented application [4]. Even with a wieldy candidates list, a skillful developer must determine the most appropriate service for the consumer application. This implies a prohibitive effort into discovering services, analyzing the suitability of retrieved candidates (i.e., service selection) and identifying the set of adjustments for the final integration of a selected candidate service.

In this work we make use of Case-based Reasoning (CBR) [5]– from the Artificial Intelligence (AI) field – to overcome the aforementioned problems in Web Service Discovery and Selection. A Case-based Reasoner solves problems by using or adapting solutions from old recurrent problems [6]. Sometimes called similarity searching systems, the most important characteristic of CBR systems is the effectiveness of the similarity function used to quantify the degree of resemblance between a pair of cases [7].

Our proposal models a Case-based Reasoner for Service Selection, where the main contribution is threefold. We define a case representation capturing information in Web Services functional descriptions (typically WSDL). Moreover, we draw a parallel among the key steps in CBR and the problem of Web Service Discovery and Selection. Finally, we provide three implementations for the similarity function, concerning structural and semantic aspects from functional service descriptions.

The rest of the paper is organized as follows. Section II details the service selection process. Section III presents the application of CBR in the context of service selection. Section IV details the alternatives for the similarity function. Section V presents the experimental validation of the approach. Section VI discusses related work. Conclusions and future work are presented afterwards.

II. Service Selection

During development of a Service-oriented Application, some of the comprising software pieces could be fulfilled by the connection to Web Services. In this case, a list of candidate Web Services could be obtained by making use of any service discovery registry. Nevertheless, even with a wieldy candidates' list, a developer must be skillful enough both to determine the most appropriate service and to shape the adaptation artifacts for seamless integration of the selected service. Therefore, a reliable and practical support is required to make those decisions. For this, in previous work [8], [9] we defined an approach for service selection.

The service selection method is based in an Interface Compatibility assessment of the candidate Web Services and the (potentially partial) specification of the required functionality – depicted in Figure 1. The procedure matches the required interface I_R and the interface (I_S) provided by a candidate service S (previously discovered). All available information from the two interfaces is gathered to be assessed at semantic and structural levels. The semantic assessment makes use of the WordNet lexical database [10] for identifiers evaluation, by means of terms separation, stop words removing, stemming and terms similarity (synonymy and hypo/hyperonymy). The

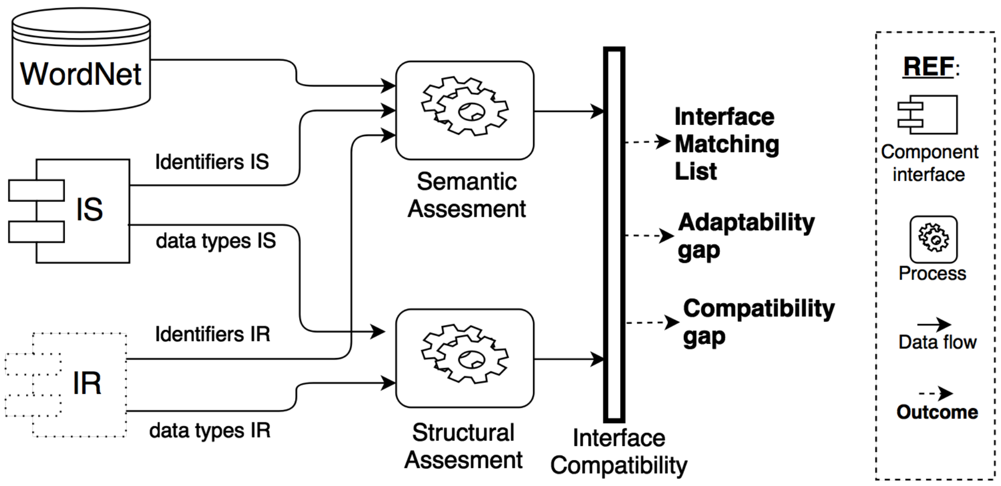


Fig. 1: Interface Compatibility Scheme

structural evaluation considers data types equivalence and subtyping.

The outcome of these evaluations is an Interface Matching list where each operation from I_R may have a correspondence with one or more operations from I_S. In addition, two appraisal values are calculated: *compatibility gap* (concerning functional aspects), and *adaptability gap* – which reflects the required effort for integrability of the selected service.

III. Case-based Reasoning for Service Selection

This work extends the Interface Compatibility Scheme by means of a CBR methodology [5]. The main goal is to capture the knowledge obtained from successive service selections as a set of cases in the form of problem-solution pairs. Figure 2 shows the CBR approach (adapted from [5]).

Let be a *knowledge base KB* containing an initial set of cases. Each case consists of a pair (problem, solution): the *problem* is a description of certain functionality, and the associated *solution* is the candidate service that fulfills such functionality. A *new case C* is a problem part (required functionality) that has to be paired with its corresponding solution (candidate service). For this, the first step compares *C* with all the problem parts in *KB*, according to a similarity function. The outcome of this step is the most similar case to *C* (*retrieved case*) – i.e., the pair $(functionality, service)$ with the most similar functionality w.r.t. *C*.

Then, the retrieved case is reused to generate the *solved case*, by combining its solution part with the new case *C* as the problem part. The solved case is then a pair $(required\_functionality, service)$. At this point, the solved case is returned as the *suggested solution*, which can be revised by expert users. If the suggested solution succeed in the revision, it then becomes a *confirmed solution* (*tested case*). If it fails, the case is discarded.

Finally, the last step decides whether or not to include the confirmed solution (tested case) in *KB*. The *learning case* decision can rely upon different criteria. In this approach, we use a threshold value (*th*) over the similarity function: if the similarity function returns a value higher than the threshold, then the case is added to the *KB*.

In the following sections, we describe the application of CBR concepts to Web Service discovery and selection.

A. Case Representation

First, it is essential to define an adequate case representation in the context of service selection. We have used an object-oriented (OO) case representation, where the cases are represented as object collections described by a set of attribute-value pairs. Object-oriented representations are appropriate for complex domains where different case structures may occur [11]. Figure 3 shows the OO case representation structure for service selection. As stated earlier, the Case class is divided into two parts – namely Problem and Solution.

The Problem part captures the required functionality to be fulfilled by a candidate service. The required functionality is composed of three simple attributes and a collection of a complex attribute. The simple attributes are *Service name* (`String`), *Category* (`String`), and *Operations number* (a positive integer including zero). The complex attribute is *Operations*, which represents the required operations. Each *Operation* contains two attributes – *Operation name* and *Return type* (Simple or Complex type) – and two collections of complex attributes – *Parameter*s and *Exception*s. *Parameter* contains two attributes: *parameter name* and *parameter type* (Simple or Complex type). *Exceptions* contains a simple attribute: *Type* (string).

The Solution part is a simple string attribute *Candidate Service* that represents the name of the service associated to the problem description.

B. Case Retrieval

New cases are given as input to the Case-based Reasoner (Figure 2) in the form of a required functionality – i.e., a problem part (P_n) . To find a solution to a *new case*, the first step calculates the similarity function (*DIST*) as a distance between the new case and each case in the knowledge base

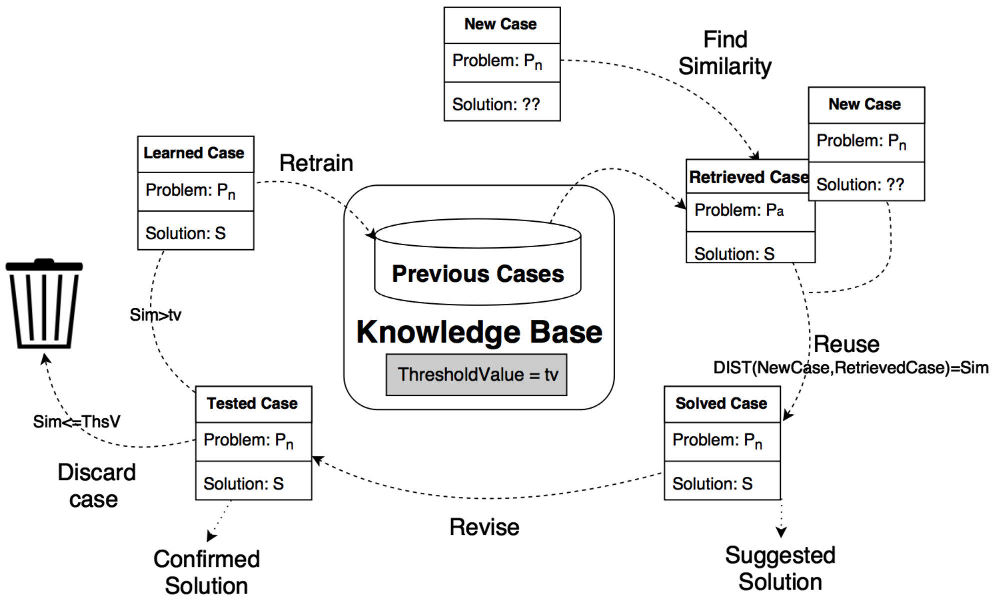


Fig. 2: Case-based Reasoning for Service Selection

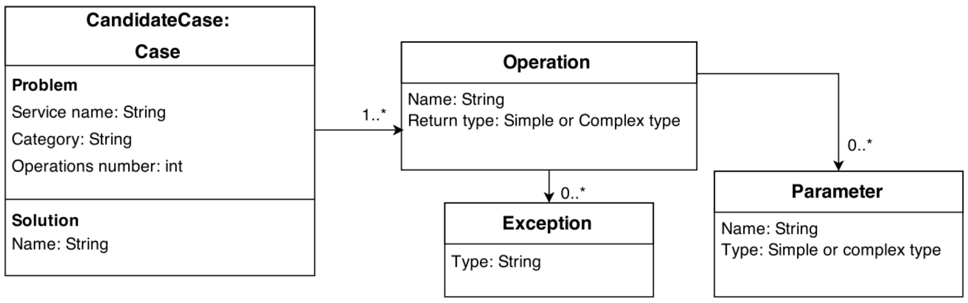


Fig. 3: Case representation

KB – according to Formula 1, extracted from [7]. For each attribute in the case representation, we have defined specific similarity functions *sim* and weights w_i (where the sum of all weights w_i is 1), – that are presented in the following sections.

$$DIST(C^N, C^C) = \sum_{1}^{n} (w_i * sim_i(C_i^N, C_i^C)) \qquad (1)$$

where C^N is a new case to evaluate and C^C is the candidate case in *KB*.

Service name and Category: The *Service name* and *category* evaluation consists in comparing the `String` values of these attributes between the new case under analysis and the problem part of each case in the *KB*. Similarity is calculated using an algorithm for Identifiers evaluation which considers semantics of terms in identifiers – discussed in detail in Section IV-B. The given weight for service name and category attributes is low ($w = 0.1$) as they do not directly express functional aspects.

Operations number: The *operations number* evaluation consists in comparing the numerical value for such attribute between the new case under analysis and the problem part of each case in the *KB*. Similarity is calculated according to Formula 2. Candidate services (solutions) in the *KB* with fewer operations than the required functionality (problem) are considered as incompatible, being discarded as potential solutions. The given weight for this attribute is higher ($w = 0.3$) as it directly expresses a functional aspect (i.e., the number of required operations).

$$Sim(\#op^N, \#op^C) = \begin{cases} \#op^N <= \#op^C & 1 \\ otherwise & 0 \end{cases} \qquad (2)$$

where $\#op^N$ and $\#op^C$ are the values of the operation number attribute in the new case and a candidate case of the *KB respectively*

a) Operations: The *operations* evaluation calculates similarity between this complex attribute in the new case, and the analogous attribute in the problem part of each case in *KB*. Since the main criterion of our service selection approach is functional similarity, this attribute presents the highest weight ($w = 0.6$) and the most complex similarity function in this Case-based Reasoner. Details of the similarity function for the *operations* attribute are presented in Section IV.

C. Case Reuse and Revision

We use the first nearest neighbor (1-NN) strategy for case adaptation, which implies that the most similar case is chosen as the best solution [12]. Therefore, the solution part into the new case will be the solution part (*S*) – i.e., the candidate service – of the most similar *retrieved case* according to Formula 1.

The *solved case* is then returned as the *suggested solution* (Figure 2), and can be revised by expert users. Expert users are people with high knowledge about the domain and the service-oriented application under construction. Thus, an expert user decides if the solution is suitable for the target application or relevant for the underlying domain. Otherwise, the *solved case* is rejected. Relevant cases can then be part of the *KB*'s initial state. Experts feedback is not mandatory but necessary to improve the reasoner performance and to determine the *threshold value* according to the state (i.e., the number of cases) of the *KB* at a given time.

D. Case Retraining

At this point, the Case-based Reasoner has compared the new case against the problem part of all the cases in the *KB* and (expectedly) has found a solution in terms of the attribute similarity presented in the previous sections. Also, the solution had been revised and acknowledged as valid by expert users – i.e., it is a confirmed solution (*tested case*). The next step consists in deciding if the *tested case* will be added to the *KB*.

On the one hand, too many retrained cases can generate noise in the evaluation, decreasing the performance of the reasoner in the long term. On the other hand, if no new cases are added, no learning occurs, so the reasoner will not be capable to deal with new cases. In order to prevent these problems, we have defined a threshold value (*th*) over the *similarity function* (*DIST*). The threshold value determines whether or not a new case is retrained (*learned case*) in the *KB*. If the *similarity function* returns a value higher than the threshold, then the case is added to the *KB* as a new meaningful case, otherwise it is discarded.

The goal is to prevent the uncontrollable growth of the *KB* while improving the performance of the reasoner. The threshold value is a configurable constant that depends on the reasoner implementation and the initial number of cases. If the initial number of available cases is low (e.g., with regard to the total number of services in a given domain), the threshold value will also be settled low, allowing the reasoner to add new cases and to enrich the *KB*. If the number of available cases grows in a certain moment, the threshold value can be increased to add only new cases with a significant similarity.

IV. Operations similarity

As we stated earlier, operation similarity is the key attribute for assessing new cases against the potential candidate services, included as cases in the *KB*. The operations similarity evaluation accounts semantic and structural aspects extracted from operation definitions. Structural aspects involve data types equivalence (subtyping), while semantic aspects involve concepts from terms and identifiers. We have defined three implementations for operations similarity that mainly differ in their semantic basis. This resulted in three different *similarity functions* for operations evaluation. In the following sections we describe the similarity evaluation for each element in operations: identifiers, operation name (evaluated as a special case of identifiers), parameters, return type, and exceptions.

A. Case Study

A simple case study has been outlined to illustrate key steps of our proposal. We considered the Car Rental domain, where the required features are portrayed according to the OO case representation (Figure 3). Thus Figure 4a shows a new case (*newCase*), which contains the description of a proposed service named `RentaCar`. Such interface defines three operations and three complex data types. Note that the solution part of the *newCase* is not instantiated since the case has not been evaluated.

The functionality of the required interface will be fulfilled by engaging a third-party Web Service. The case in the *KB candidateCase* contains the representation of the service `CarRentalBrokerService`. The *candidateCase* defines four operations and three complex data types – as shown in Figure 4b.

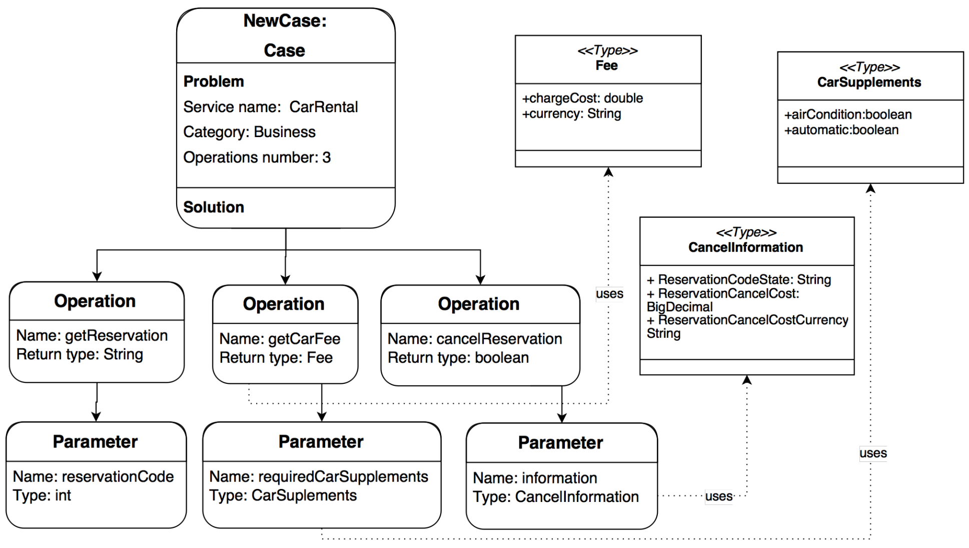
Both cases were built by adapting real Web Services[1,2] from the Car Rental domain to illustrate our proposal.

B. Identifiers Evaluation

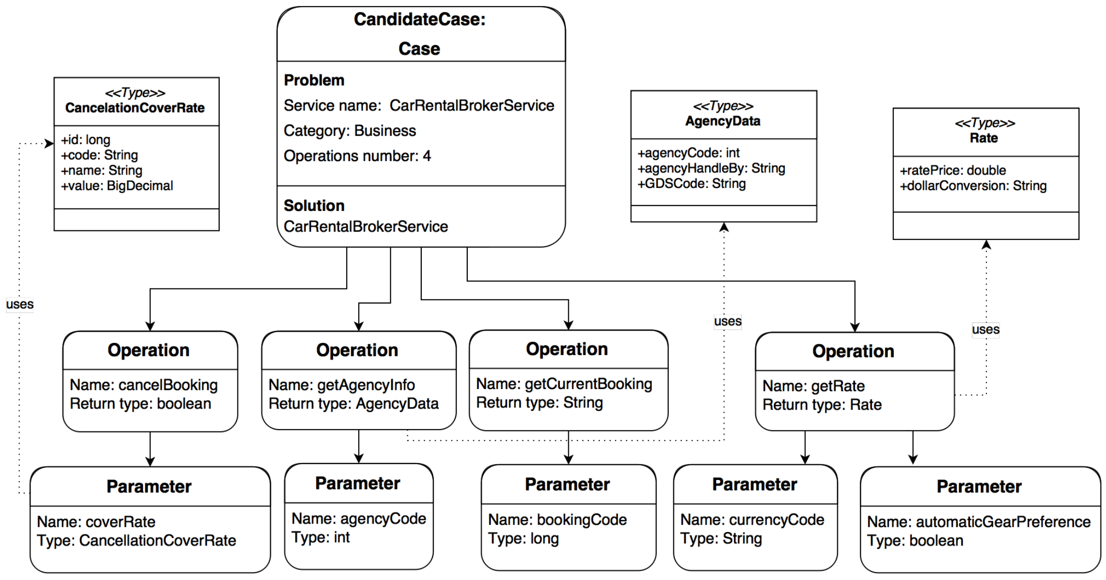
To evaluate semantic aspects the similarity functions compare terms and identifiers from operations. We implemented three alternatives for these functions. The first two make use of WordNet [10]. WordNet is a domain-independant lexical database of the english language that is structured as a lexical tree. WordNet groups terms in *synsets* (synonym sets) that represent the same lexical concept. Several relationships connect different synsets, such as hypo/hyperonyms, holonyms/meronyms and antonyms. All hierarchies ultimately go up the root node $\{entity\}$. The WordNet structure can be accessed through different Java libraries, each one implementing different metrics and features [13]. Particularly, in this work we used JWI[3] (in the first similarity function) and JWNL[4] (in the second similarity function). These libraries are among the most complete and easy to use for WordNet lexical tree manipulation [13].

The third alternative for the similarity function is based upon DISCO [14], a pre-computed database of collocations and distributionally similar words. DISCO's Java library[5] allows to retrieving the semantic similarity between arbitrary words. The similarities are based on the statistical analysis of very large text collections (e.g., Wikipedia), through co-occurrence functions. For each word, DISCO stores the first and second order vectors of related words using a Lucene index [15]. To

[1]http://goo.gl/MC7uXh
[2]http://goo.gl/LL0k0w
[3]http://projects.csail.mit.edu/jwi/
[4]http://web.stanford.edu/class/cs276a/projects/docs/jwnl/overview
[5]http://www.linguatools.de/disco/disco-api-1.4/



(a) New case instantiation



(b) Candidate case instantiation

Fig. 4: Object cases representation for the Car Rental example

determine the similarity between two words, DISCO retrieves the corresponding word vectors from the index and computes the similarity based in co-occurrences.

Following we describe the similarity functions implemented using JWI, JWNL and DISCO. To determine the similarity between two identifiers, these implementations share two preliminary common steps. Thus, two identifiers are initially preprocessed through term separation and stop words removal [8]:

Term Separation: Identifiers are normally restricted to a sequence of one or more letters in ASCII code, numeric characters and underscores ("_") or hyphens ("-"). The algorithm supports the rules in Table I – i.e., the usual programming naming conventions – plus a semantic level to consider identifiers that do not follow those conventions. The Term Separation step analyzes the identifiers, recognizing potential terms (uppercase sequences and lowercase sequences). Then, WordNet is used to analyze all the potential terms and determines the most adequate term separation. The term separation step is crucial to consider the correct terms as input to the semantic analysis.

Example. Let be the identifier `GDSCode` from the Car Rental domain. This identifier does not strictly follow the Java Beans notation. An initial analysis identifies an uppercase sequence (`GDSC`), and a lowercase sequence (`ode`). Then, the sequence `C + ode = Code` is given as input to WordNet. As it is an existing word in the WordNet dictionary, `Code` is considered as a term and `GDS` as an acronym (an abbreviation of Global Distribution System) that is also considered as a term.

Stop Words Removal: Stop words are meaningless words that are filtered out prior to, or after, processing natural language data (text) [16]. We defined a stop words list containing

TABLE I: Rules for Decomposing Identifiers

Notation	**Rule**	**Source**	**Result**
Java Beans	Splits when changing text case	getZipCode	get Zip Code
Special symbols	Splits when either "_" or "-" occurs	get_Quote	get Quote

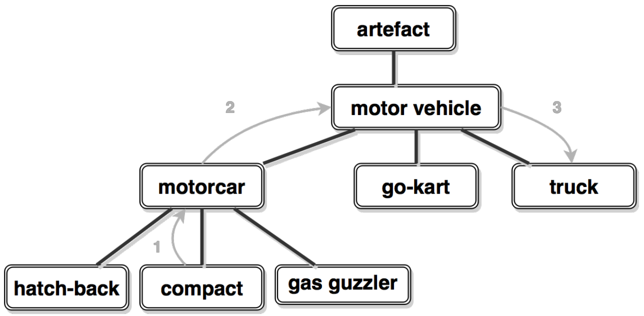


Fig. 5: Length between "compact" and "truck" in the WordNet hierarchy

articles, pronouns, prepositions, words from other stop words lists and each letter of the alphabet. The terms lists obtained from the previous step are analyzed to remove any occurrence of a word belonging to the stop words list.

Example. Let consider the identifier `AgencyHandledBy` which corresponds to a field in the Data Type `AgencyData` of the Car Rental example. According to the Java Beans notation, the identifier is decomposed in three terms: [`Agency`, `Handled`, `By`]. As `'By'` belongs to the stop words list, it is removed from the terms list.

1) JWI-based Identifiers Evaluation: The JWI implementation comprises three main additional steps: stemming, terms lists semantic comparison and identifiers compatibility calculation.

Stemming is the process for reducing words to their stem, base or root form. Due to common problems of standard syntactical stemmers [17], we adapted the semantic stemmer provided by WordNet. The Stemming step receives as input a terms list. For each term in the list is verified that it belongs to the WordNet dictionary. If it does so, the corresponding stems are added to the result list. Otherwise, the original term is added to the result list, considered as an abbreviation or acronym.

After generating both lists of stems, their compatibility is calculated considering semantic information. This information is expressed as a vector of integers $v = \{t, e, s, h_1, h_2\}$ including: the total terms between both lists (t), the identical (exact) terms (e), synonyms (s), hyperonyms (h_1) and hyponyms (h_2). For example, let be the identifiers `GetReservation` and `GetCurrentBooking` extracted from the cases of the Car Rental example in Section IV-A. According to the term lists semantic comparison, these identifiers present:

- Four distinct terms: (`Get`, `Reservation`, `Current`, `Booking`)
- One exact (identical) term: `Get`
- One synonym: (`Reservation`, `Booking`)
- No hypo/hyperonyms

Using these values in the vector v as input, the *Identifiers Compatibility Value* is calculated according to Formula 3.

$$ICValue = \frac{e + s + 0.5 * (h_1 + h_2)}{t - s} \qquad (3)$$

Example. Let be the identifiers `GetReservation` and `GetCurrentBooking` extracted from the interfaces of the Car Rental example, by replacing the values in Formula 3 we obtain:

$$ICValue = \frac{1 + 1 + 0.5 * (0 + 0)}{4 - 1} = \frac{2}{3} = 0.66$$

Which indicates a compatibility value of 0.66 between the identifiers `GetReservation` and `GetCurrentBooking` – considering that the maximum value for *ICValue* is 1, the obtained value indicates a moderate to strong compatibility between the identifiers.

2) JWNL-based Identifiers Evaluation: The JWNL implementation calculates the compatibility value according to two main additional steps: generation of the normalized depth matrix and term matching maximization.

First, the *Normalized Depth* matrix (*ND*) is generated. The depth is defined as the shortest path between two terms in the WordNet hierarchy. These values are normalized by the maximum depth of the WordNet hierarchy (16). Formally, the normalized depth is calculated according to Formula 4.

$$NormalizedDepth(t_i, t_j) = \frac{2D - length(t_i, t_j)}{2D} \qquad (4)$$

where $length(t_i, t_j)$ = shortest path between t_i, t_j in the WordNet hierarchy, D is the maximum tree depth (16)

Example. Figure 5 shows an excerpt of the WordNet hierarchy, showing different types of vehicles. It shows that the *length* between the concepts `compact` and `truck` is 3, and the length between `compact` and `motor vehicle` is 2. These values indicate that `compact` and `motor vehicle` are more similar than `compact` and `truck` – according to JWNL.

Accounting this notion of length, lets consider the two identifiers `GetReservation` and `GetCurrentBooking` (analyzed in Section IV-B1). The *ND* matrix will be a 2x3 matrix containing the length between each pair of terms in the identifiers, as shown in Table II. Notice that $ND($`Reservation`,`Booking`$) = 1$ since these terms are synonyms in the WordNet hierarchy (their path length is zero).

TABLE II: Normalized Depth matrix for the identifiers `GetReservation` and `GetCurrentBooking`

	Get	Current	Booking
Get	1.00	0.56	0.72
Reservation	0.72	0.72	1.00

Higher is better. Maximum and minimum values are 1 and 0 respectively.

After calculating the *ND* matrix, the best term matching (among all possible pair-wise combinations) must be selected – i.e., the combination of terms from both terms lists that maximizes their compatibility. For each possible pair-wise term assignment (t_i, t_j) between both lists, the similarity value is obtained from the corresponding matrix cell ND_{ij}. The value of each possible term matching is the sum of all pair-wise assignments that compose it (*assignSum*). The matching with the highest value is obtained through the Hungarian method [18], as an instance of the allocation problem.

Finally, the identifiers compatibility value (*ICValue*) using JWNL is calculated according to Formula 5, which weights the sum of the pair-wise assignments of terms according to the maximum number of terms in the identifiers under analysis.

$$ICValue = \frac{assignSum}{max(n,m)} \tag{5}$$

where n and m are the number of terms in both terms lists.

Example. Considering the *ND* matrix shown in Table II, the term matching that maximizes the compatibility between the identifiers consists of the following pair-wise assignments:

- [`Get`,`Get`], stored in the cell $ND_{1,1} = 1.00$
- [`Reservation`, `Booking`], stored in the cell $ND_{2,3} = 1.00$

Then, replacing the corresponding values in Formula 5, the compatibility value between identifiers `GetReservation` and `GetCurrentBooking` is calculated as follows:

$$ICValue = \frac{1+1}{max(2,3)} = \frac{2}{3} = 0.66$$

3) DISCO-based Identifiers Evaluation: The DISCO-based implementation calculates the compatibility value according to two main steps: generation of the co-occurrences matrix and term matching maximization.

First, the Co-occurrences matrix (*Co*) is generated. This matrix contains the similarity values between each term from both terms lists. These values are the result of applying the co-occurrences similarity notion of DISCO, explained earlier. After calculating the *Co* matrix, the best term matching (among all possible pair-wise combinations) must be selected. Similarly to the JWNL-based implementation, this step uses the *Co* matrix as input for the Hungarian algorithm. The matching with the highest value will be the most compatible. Such matching is also obtained through the Hungarian method – introduced in Section IV-B2. Finally, the identifiers compatibility value using DISCO is calculated according to Formula 5.

Example. Lets consider the pair of identifiers of the previous section – namely `GetReservation` and `GetCurrentBooking`. The *Co* matrix will be a $2x3$ matrix containing the co-occurrence values between each pair of terms in the identifiers, as shown in Table III. Notice that, when using DISCO rather than WordNet, synonyms do not present a co-occurrence value of 1 – as can be seen for the pair (`Reservation`, `Booking`).

TABLE III: Co-occurrences matrix for the identifiers `GetReservation` and `GetCurrentBooking`

	Get	Current	Booking
Get	1.00	0.006	0.02
Reservation	0.01	0.01	0.1

Higher is better. Maximum and minimum values are 1 and 0 respectively.

Considering the *Co* matrix shown in Table III, the term matching that maximizes the compatibility between the identifiers consists of the following pair-wise assignments:

- [`Get`,`Get`], stored in the cell $Co_{1,1} = 1.00$
- [`Reservation`, `Booking`], stored in the cell $Co_{2,3} = 0.1$

Then, replacing the corresponding values in Formula 5, the compatibility value between identifiers `GetReservation` and `GetCurrentBooking` is calculated as follows:

$$ICValue = \frac{1+0.1}{max(2,3)} = \frac{1.1}{3} = 0.36$$

C. Return type

Data Type Equivalence: Conditions for data type equivalence involves the subsumes relationship or subtyping, which implies a direct subtyping in case of built-in types in the Java language [19], as shown in Table IV. It is expected that types on operations from a *new case* have at least as much precision as types on operations from a candidate service (case in the *KB*). For example, if $op^N \in newCase$ includes an `int` type, a corresponding operation $op^C \in candidateService$ should not have a smaller type (among numerical types) such as `short` or `byte`. However, the `String` type is a special case, which is considered as a wildcard type since it is generally used in practice by programmers to allocate different kinds of data [20]. Thus, we consider `String` as a supertype of any other built-in type.

TABLE IV: Subtype Equivalence

op^N **type**	op^C **type**
char	string
byte	short, int, long, float, double, string
short	int, long, float, double, string
int	long, float, double, string
long	float, double, string
float	double, string
double	string

Complex Data Types: Complex data types imply a special treatment in which the comprising fields must be equivalent one-to-one with fields from a counterpart complex type. This means, each field of a complex type from an operation $op^N \in$ *newCase* must match a field from the complex type in $op^C \in$ *candidateService* – though extra fields from *newCase* may be initially left out of any correspondence.

The return type similarity value is calculated according to the following cases:

- $Ret = 3$: Equal Return Type.
- $Ret = 2$: Equivalent Return Type (Subtyping, String or Complex types).
- $Ret = 1$: Non-equivalent complex types or precision loss.
- $Ret = 0$: Not compatible .

Example. Figure 6 shows the field-to-field equivalence (considering only data types) for two complex types of the Car Rental example, which contains information about booking cancellation rates. The three fields of the `CancelInformation` type have a one-to-one correspondence with three fields of the `CancellationCoverRate`. The dotted arrows indicate a likely correspondence between the `String` types. For this example the return type similarity value is $Ret = 2$.

D. Parameters evaluation

The algorithm for Parameters Evaluation consists of calculating three matrices: Type (T), Name (N) and Compatibility ($Comp$). For the three matrices, the cell M_{ij} represents the compatibility value between the *i-th* parameter of op^N and the *j-th* parameter of op^C – where op^N is an operation of the required functionality (*new case*) and op^C is an operation of a candidate service (case in the *KB*).

In the T matrix, the notions of structural data type equivalence and subtyping are used to assess parameter types. The goal of the T matrix is to store the relationship between all pairs of parameter types from both operations. A cell T_{ij} contains the compatibility value between the *i-th* parameter's type of op^N and the *j-th* parameter's type of op^C, according to Formula 6.

$$\begin{cases} Type(P_i) = Type(P_j) & T_{ij} = 2 \\ Type(P_i) <: Type(P_j) & T_{ij} = 1.5 \\ otherwise & T_{ij} = 1 \end{cases} \tag{6}$$

where <: represents the sutyping relationship

The N matrix contains the compatibility values between the name of each parameter from op^N and the name of each parameter from op^C. The underlying rationale is similar to the T matrix. The cell N_{ij} contains the compatibility value between the *i-th* parameter's name of op^N and the *j-th* parameter's name of op^C. This value is the result of applying the Identifiers Evaluation Algorithm presented in Section IV-B. Therefore, these values depend on the chosen similarity function implementation – from the three alternatives.

Then, the $Comp$ matrix is generated from the T and N matrices. The goal of the $Comp$ matrix is to store the compatibility value between all parameter pairs from operations op^N and op^C, considering structural and semantic aspects – collected in the T matrix and the N matrix respectively. Each cell $Comp_{ij}$ stores the product between T_{ij} and N_{ij}. Thus: $Comp_{ij} = T_{ij} * N_{ij}$.

After calculating the $Comp$ matrix, the best parameter matching (among all possible pair-wise combinations) must be selected – i.e., the combination of parameters from op^N and op^C that maximizes their compatibility. This step applies the Hungarian algorithm to calculate the best pair-wise parameters assignments – similarly to the term matching maximization in JWNL-based Identifiers Evaluation (Section IV-B),.

E. Exceptions

Structural conditions for exceptions are evaluated as follows. First, any operation op^N may define *default* exceptions – i.e., using the `Exception` type – or ad-hoc exceptions. Likewise, an operation op^C from a candidate case may define a *fault* (the WSDL name for non-standard outputs of operations) as a message including an specific attribute. The exceptions similarity value is calculated according to the following cases:

- $Exc = 1$: op^N and op^C have equal amount, type and order for exception.
- $Exc = 2$: op^N and op^C have equal amount and type for exception into the list.
- $Exc = 3$: if nonempty op^N exception list then nonempty op^C exceptions list.
- $Exc = 0$: op^N and op^C exceptions are not compatible.

In fact, in the context of Web Services, faults definitions have not become a common practice [21]. However, the Case-based Reasoner considers this simple schema to analyze exceptions.

Example. Lets consider the following operations for obtaining rates for Car Rental, according to different vehicles and conditions (from the cases presented in Section IV-A):

- `getCarFee(requiredCarSupplements: CarSupplements): Fee throws unavailableSupplementsException;`
- `getRate(currencyCode: String, vehicleTypeId: long, AutomaticGearPreference: boolean): Rate throws vehicleNotFoundException, rateNotFoundException;`

If we consider `getCarFee` as $op^N \in$ *newCase* and `getRate` as $op^C \in$ *candidateCase* respectively, the exceptions analysis shows that the required operation throws an exception that may have two likely exceptions (form different types) in the candidate service's operation – as defined in case (2).

F. Similarity value

The similarity value between two cases (C^N, C^C) is calculated according to Formula 7.

$$sim(C^N, C^C) = \frac{\sum_{i=1}^{N}(Max(simOp(op_i^N, C^C))}{N} \tag{7}$$

where N is the number of operations in C^N, and *simOp* is the best equivalence value $simOpValue(op_i^N, op_j^C)$ for all op_j^C in C^C

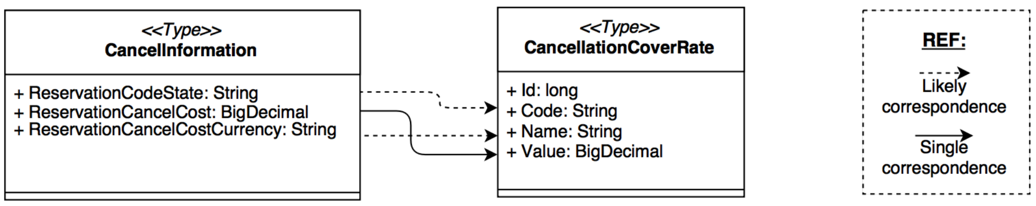


Fig. 6: Equivalence of Complex Data types for the CarRental example

The value for operation similarity (*simOpValue*) between an operation op^N and a potentially compatible operation op^C is calculated according to Formula 8.

$$simOpValue(op_i^N, op_j^C) = Ret + Exc + Name + Par \tag{8}$$

a) Example.: Lets consider the full cases of the Car Rental case study, presented in Section IV-A. Table V shows, for each required operation $op^N \in newCase$, the operation $op^C \in candidateCase$ with higher compatibility (according to their *adapOpValue*) in the interface of the candidate Web Service. Calculations were done using the WordNet semantic basis accessed through the JWI library.

As the higher (better) *sim* value is 8, the obtained *sim* value (5.4) can be considered as moderate to high.

After obtaining the similarity value for operations, we can calculate the distance between the cases presented in 4 according to the Formula 1. Table VI shows a summary of the required calculations to obtain the distance value.

Let be new case $C^N = CarRental$ and candidate case $C^C = CarRentalBrokerService$ the distance between the cases is:

$$DIST(C^N, C^C) = 0,1 * s_N + 0,1 * cat + 0,3 * \#op + 0,6 * ops$$

$$DIST(C^N, C^C) = 0,1 * 0,5 + 0,1 * 1 + 0,3 * 1 + 0,6 * 5,4$$

$$DIST(C^N, C^C) = 3,69$$

V. Experimental evaluation

This section describes the experimental evaluation of the CBR for service selection presented in the previous sections. The goal of the experiments is to measure the overall performance of the three alternative implementations of the CBR for service selection in comparative terms. We adopted an empirical, automatized and widely used methodology [9], [22]–[24]. Our hypothesis is that CBR for service selection could increase visibility of the most relevant services for certain required functionality.

A. Experiment configuration

The considered data-set consisted in 62 services extracted from the data-set of [25]. We have generated (through a tool developed in our group) one case for each service to settle the initial *KB*, according to the Object-oriented case representation presented in Section III-A.

First, we extracted operation signatures from the 62 relevant services. Each new case consisted of three operations representing required functionality. Then, we applied interface mutation techniques [26] to generate 506 new cases (only problem part). We applied three mutation operators[6] to each operation signature:

- *Encapsulation* – where a random number of parameters are encapsulated as fields of a new complex data type.
- *Flatten* – where a random number of complex parameters are flattened generating as many parameters as fields in the complex type.
- *Upcasting* – where the return type and/or a random number of parameters are upcasted either to a direct supertype or to the wildcard `String` type.

B. Case-based Reasoning Execution

To execute the CBR for service selection, we have defined one scenario considering three implementations according to the similarity functions presented in Section IV, and the 506 new cases generated by mutating operation signatures. Considering traditional techniques of service retrieval and selection, we also populated the EasySOC service registry [27] with the relevant services, and then queried such registry with the operation signatures. EasySOC leverages Vector Space Model (VSM) and Web Service query-by-example (WSQBE) to represent Web Service descriptions and queries.

The three versions of the CBR for service selection were executed to rank the retrieved cases. Finally, the results of each new case are measured in terms of two well-known Information Retrieval metrics: *recall* and *precision-at-n.*

C. Results

Considering the results list as the first 10 retrieved cases for each query, we compared the results according to *precision-at-n* and *recall.*

Precision-at-n: Indicates in which position are retrieved the relevant services, at different cut-off points. For example, if the top five documents are all relevant to a query and the next five are all non-relevant, precision-at-5 is 100%, while precision-at-10 is 50%. In this case, precision-at-n has been calculated for each query with *n* in [1–10].

[6]https://code.google.com/p/querymutator/

TABLE V: Operations matching for the Car Rental cases

`RentaCar` (*newCase*)	**`CarRentalBrokerService`** (*candidateCase*)	*simOpValue**
GETRESERVATION	GETCURRENTBOOKING	7.3
GETCARFEE	GETRATE	2.4
CANCELRESERVATION	CANCELBOOKING	6.4
	$sim(newCase, candidateCase)$	5.4

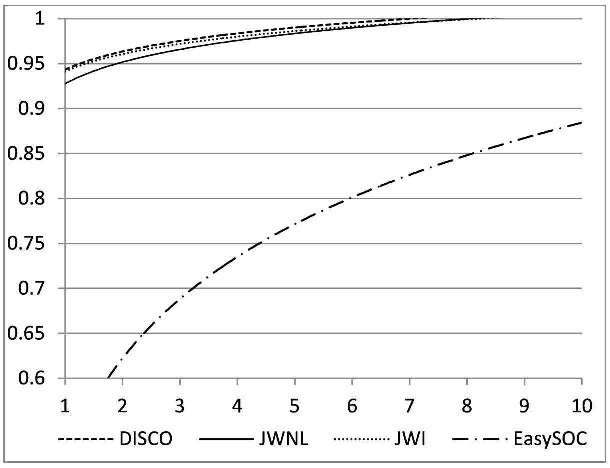
* Higher is better

TABLE VI: Summary of the required calculations

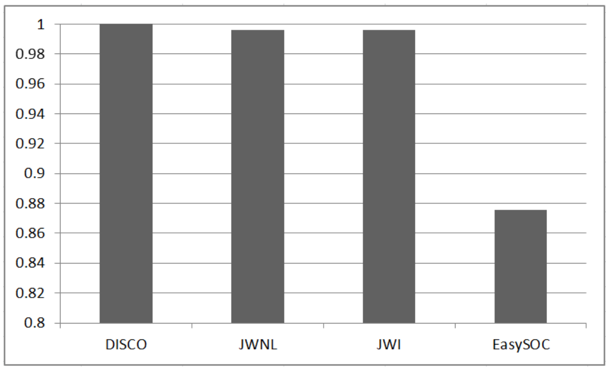
Attribute	**Weight***	**Evaluation**	**Result****
service name (Formula 3): s_N	0.1	$ICValue(CarRental, CarRentalBrokerService)$	0.5
category (Formula 3): *cat*	0.1	$ICValue(Business, Business)$	1
operations number (Formula 2):$\#op$	0.3	$sim(3,4)$	1
operations (Formula 7): *ops*	0.6	$sim(CarRental, CarRentalBrokerService)$	5.4

* The sum of all weights is 1. The higher the weight the more important the attribute.

** Higher is better.



(a) Precision-at-n



(b) Recall

Fig. 7: Results for the CBR implementations and EasySOC

Recall: Formally, Recall is defined as:

$$Recall = \frac{Relevant}{R}$$

Where Relevant is the number of relevant services includes in the results list and R is the number of relevant services for a given query.

In particular, for this experiment the numerator of the Recall formula could be 0 or 1 – when the relevant service is/is not included within the results respectively – and the denominator (Retrieved) is always 10.

Figure 7a depicts the cumulative average precision-at-n (with n=[1,10]) for the three implementations of the CBR approach (JWI-, JWNL- and DISCO-based) and the EasySOC registry. The CBR for service selection obtained precision values over 90% for the first position of the results (with n = 1) with any implementation. Also, the difference among the precision-at-n of the three CBR implementations was not significant. The CBR for service selection outperformed EasySOC registry between 20% and 40% for the first positions of the results lists (with n=[1,4]).

Figure 7b shows Recall results for the three implementations of the CBR-based service selection and the EasySOC registry. The results show that the CBR for service selection presents high recall values – over 98% – independently of the underlying implementation. This means that the relevant case for the given problem is almost always retrieved. The CBR for service selection outperformed recall results for EasySOC by about a 10%, although the EasySOC presented highly competitive values (over 85%) for recall as well.

D. Discussion

The results of the experiments have shown that our CBR for service selection approach achieves a high precision and recall with the three alternative implementations of the similarity function. Comparison with a service discovery registry (EasySOC) presented encouraging results as well. This confirms our hypothesis, as CBR increased visibility of relevant services during service selection. This is significant since users tend to select higher ranked search results, regardless to their actual relevance [28]. The overall performance of the approach with multi-operation queries suggest the suitability for matching complex required functionality with many candidate services. In this direction, the reasoner could be extended to the Web Service Composition (WSC) problem, by means of case adaptation using the K-nearest neighbors (K-NN) strategy [12]. Finally, the threshold value over the similarity function can be used to fine-tune the reasoner, according to the size of the initial *KB* and the domain. In this experiment, an average of 57% of the solved cases was added to the *KB* as new cases – i.e., 288 of the 506 queries.

As limitations, we can mention that the results can be specific for this experiment, and cannot be merely generalized to other experimental configurations. The dataset was relatively small (62 services), and the threshold values were fine-tunned by trial and error in the experimental scenario. In real scenarios, it would be wise to account the expert feedback from the Case Revision step to adjust the threshold value. Finally, the initial set of cases and solutions in a real scenario has to be manually built, which can be time consuming and also need expert feedback.

VI. RELATED WORK

A. *Case-based Reasoning for Web Services*

AI has contributed significantly to the Web Services field, either in the form of planning [29]–[31], abstraction and refinement techniques [32], or case based reasoning [33], [34].

The work in [34] presents an approach for WSC using CBR. This approach combines CBR with semantic specifications of services in the OWL-S language [35] to firstly reduce the search space of Web Services (i.e., improve service discovery), and then build an abstract composite process. Authors assume that Web Service providers are in charge of semantically annotating functional service descriptions according to the OWL-S ontology. However, this hardly occurs in practice, and most domains currently lack a descriptive ontology [22]. Our work exploits the most possible information in the (always available) service functional descriptions, to build the case representation.

The work in [33] also applies CBR for WSC. Similarly to our work, CBR is applied for service discovery, as a crucial step in the composition process. This approach requires knowing a priori a set of relationships between the services that compose the KB – e.g., dependence, substitutability and independency. However, this approach is strongly dependant of the Universal Description, Discovery and Integraton (UDDI) registry, that lacks a broad adoption in the industry [36]. Our approach is not tied to any particular discovery registry.

B. *Structural-semantic service selection*

Web Service similarity is addressed in [37] as a key solution to find relevant substitutes for failing Web Services. The approach calculates lexical and semantic similarity between identifiers comprising service names, operations, input/output messages, parameters, and documentation. To compare message structures and complex XML schema types, authors make use of schema matching. However, a straightforward comparison of complex types can be performed without dealing with the complexity of an XML schema [9].

The Woogle search engine for Web Services is presented in [38]. Based on similarity search, Woogle returns similar Web Services for a given query based on operation parameters as well as operations and services descriptions. Authors introduced a clustering algorithm for grouping descriptions in a reduced set of terms. After that, similarity between terms is measured using a classical IR metric such as TF/IDF. The provided solution is limited to evaluating similarity using semantic relations between clustered terms.

The work in [39] extends UDDI with UDDI Registry By Example (URBE), a Web Service retrieval algorithm for substitution purpose. The approach considers the relationships between the main elements composing a service specification (portType, operation, message, and part) and, if available, semantic annotations. The weak point of the approach is, as we stated earlier, that providers do not annotate their services often in practice, even when introducing annotations provides a more accurate description of the service.

VII. CONCLUSIONS AND FUTURE WORK

This paper presented the application of CBR to the problem of service discovery and selection. This approach leverages the semantic and structural information gathered from always-available functional descriptions of services. Also, the approach combines notions of CBR with the use of WordNet and DISCO as lightweight semantic basis. This results in a Case-based Reasoner capable of increasing the visibility of relevant services to fulfill certain required functionality – the relevant service was returned as suggested solution in about a 90% of the cases.

The proposed scheme was tested for three different similarity functions, which shown similar performance by considering the whole semantic and structural information available from services. However, it is mandatory to define and fine-tune adequately the threshold values to circumscribe the growing of the knowledge base. This can be done at runtime, accounting feedback from domain-experts and service-experts. As future work, we plan to extend our reasoner to the WSC field, by combining different cases as a solution for complex required functionality [33].

ACKNOWLEDGMENT

This work is supported by projects: PICT 2012-0045 and UNCo–Reuse(04-F001).

REFERENCES

[1] J. Erickson and K. Siau, “Web Service, Service-Oriented Computing, and Service-Oriented Architecture: Separating hype from reality,” *Journal of BD Management*, vol. 19, no. 3, pp. 42–54, 2008.

[2] M. Bichler and K. Lin, “Service-Oriented Computing,” *Computer*, vol. 39, no. 3, pp. 99–101, 2006.

[3] R. McCool, “Rethinking the Semantic Web,” *IEEE Internet Computing*, vol. 9, no. 6, pp. 86–87, 2005.

[4] M. Crasso, A. Zunino, and M. Campo, "A survey of approaches to web service discovery in service-oriented architectures," *Journal of Database Management (JDM)*, vol. 22, no. 1, pp. 102–132, 2011.

[5] A. Aamodt and E. Plaza, "Case-based reasoning; foundational issues, methodological variations, and system approaches," *AI COMMUNICATIONS*, vol. 7, no. 1, pp. 39–59, 1994.

[6] C. K. Riesbeck and R. C. Schank, *Inside case-based reasoning*. Psychology Press, 2013.

[7] T. W. Liao, Z. Zhang, and C. R. Mount, "Similarity measures for retrieval in case-based reasoning systems," *Applied Artificial Intelligence*, vol. 12, no. 4, pp. 267–288, 1998.

[8] A. De Renzis, M. Garriga, A. Flores, A. Cechich, and A. Zunino, "Semantic-structural assessment scheme for integrability in service-oriented applications," in *Computing Conference (CLEI), 2014 XL Latin American*, pp. 1–11, Sept 2014.

[9] M. Garriga, A. Flores, C. Mateos, A. Zunino, and A. Cechich, "Service selection based on a practical interface assessment scheme," *International Journal of Web and Grid Services*, vol. 9, pp. 369–393, October 2013.

[10] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, "Introduction to Wordnet: An On-line Lexical Database," *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, 1990.

[11] F. Coenen, "Data mining: past, present and future," *The Knowledge Engineering Review*, vol. 26, no. 01, pp. 25–29, 2011.

[12] I. Watson, "Case-based reasoning is a methodology not a technology," *Knowledge-based systems*, vol. 12, no. 5, pp. 303–308, 1999.

[13] M. A. Finlayson, "Java libraries for accessing the princeton wordnet: Comparison and evaluation," in *Proceedings of the 7th Global Wordnet Conference, Tartu, Estonia*, 2014.

[14] P. Kolb, "Experiments on the difference between semantic similarity and relatedness," *Proceedings of the 17th Nordic Conference on Computational Linguistics - NODALIDA'09*, May 2009.

[15] E. Hatcher, O. Gospodnetic, and M. McCandless, *Lucene in Action*. Manning Publications Greenwich, CT, 2004.

[16] M. Armentano, D. Godoy, M. Campo, and A. Amandi, "Nlp-based faceted search: Experience in the development of a science and technology search engine," *Expert Syst. Appl.*, vol. 41, no. 6, pp. 2886–2896, 2014.

[17] P. Willett, "The porter stemming algorithm: then and now," *Program: electronic library and information systems*, vol. 40, no. 3, pp. 219–223, 2006.

[18] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[19] J. Gosling, B. Joy, G. Steele, and G. Bracha, *Java^{TM} Language Specification*. Sun Microsystems, Inc, US: Addison-Wesley, 3rd. ed., 2005. http://java.sun.com/docs/books/ jls/third_edition/html/j3TOC.html.

[20] J. Pasley, "Avoid XML Schema Wildcards For Web Service Interfaces," *IEEE Internet Computing*, vol. 10, no. 3, pp. 72–79, 2006.

[21] M. Crasso, J. M. Rodriguez, A. Zunino, and M. Campo, "Revising WSDL Documents: Why and How," *IEEE Internet Computing*, vol. 14, no. 5, pp. 48–56, 2010.

[22] D. Bouchiha, M. Malki, A. Alghamdi, and K. Alnafjan, "Semantic web service engineering: Annotation based approach," *Computing and Informatics*, vol. 31, no. 6, pp. 1575–1595, 2012.

[23] E. Stroulia and Y. Wang, "Structural and Semantic Matching for Assessing Web-Services Similarity," *International Journal of Cooperative Information Systems*, vol. 14, pp. 407–437, 2005.

[24] C. Mateos, M. Crasso, A. Zunino, and J. L. Ordiales, "Detecting WSDL bad practices in code–first Web Services," *International Journal of Web and Grid Services*, vol. 7, no. 4, pp. 357–387, 2011.

[25] A. Heß, E. Johnston, and N. Kushmerick, "Assam: A tool for semi-automatically annotating semantic web services," in *The Semantic Web–ISWC 2004*, pp. 320–334, Springer, 2004.

[26] S. Gosh and A. P. Mathur, "Interface Mutation," *Software Testing, Verification and Reliability*, vol. 11, pp. 227–247, 2001. http://www.interscience.wiley.com.

[27] M. Crasso, C. Mateos, A. Zunino, and M. Campo, "Easysoc: Making web service outsourcing easier," *Information Sciences*, vol. 259, pp. 452 – 473, 2014.

[28] E. Agichtein, E. Brill, S. Dumais, and R. Ragno, "Learning User Interaction Models for Predicting Web Search Result Preferences," in *29th Annual ACM SIGIR International Conference on Research and Development in Information Retrieval*, pp. 3–10, ACM Press, 2006.

[29] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso, "Automated composition of web services by planning at the knowledge level," in *IJCAI*, pp. 1252–1259, 2005.

[30] P. Bertoli, M. Pistore, and P. Traverso, "Automated composition of web services via planning in asynchronous domains," *Artificial Intelligence*, vol. 174, no. 3, pp. 316–361, 2010.

[31] J. Rao and X. Su, "A survey of automated web service composition methods," in *International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, pp. 43–54, 2004.

[32] H. Kil, W. Nam, and D. Lee, "Efficient abstraction and refinement for behavioral description based web service composition.," in *IJCAI*, pp. 1740–1745, 2009.

[33] B. Limthanmaphon and Y. Zhang, "Web service composition with case-based reasoning," in *Proceedings of the 14th Australasian database conference-Volume 17*, pp. 201–208, Australian Computer Society, Inc., 2003.

[34] S. Lajmi, C. Ghedira, and K. Ghedira, "Cbr method for web service composition," in *Advanced Internet Based Systems and Applications*, pp. 314–326, Springer, 2009.

[35] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. McGuiness, E. Sirin, and N. Srinivasan, "Bringing semantics to web services with owl-s," *World Wide Web*, vol. 10, pp. 243–277, 2007.

[36] Y. Dai, Y. Feng, Y. Zha^{oo}o, and Y. Huang, "A method of uddi service subscription implementation," in *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on*, pp. 661–666, IEEE, 2014.

[37] O. Tibermacine, C. Tibermacine, and F. Cherif, "Wssim: a tool for the measurement of web service interface similarity," in *Proceedings of the french-speaking Conference on Software Architectures (CAL'13)*, (Toulouse, France), May 2013.

[38] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proceedings of the International Conference on Very Large Data Bases VLDB*, pp. 372–383, VLDB Endowment, 2004.

[39] P. Plebani and B. Pernici, "Urbe: Web service retrieval based on similarity evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 11, pp. 1629–1642, 2009.

Cambiando el foco de una organización: hacia sistemas de información conscientes de los procesos

Andrea Delgado[1]
[1]Universidad de la República. Montevideo Uruguay
email: *adelgado@fing.edu.uy*

Schedule: Thu 22st@16:15, **Room:** B

Charla invitada: Cambiando el foco de una organización: hacia sistemas de información conscientes de los procesos

Evolution of a Model-driven Process Framework

Wilson Pádua
Federal University of Minas Gerais
Rua dos Oitis, 288 – Morro do Chapéu
34000-000 Nova Lima – MG – Brazil
E-mail: wppf@ieee.org

***Abstract*—We discuss the evolution of Praxis, a model-driven process framework, building on feedback from educational and professional applications, along the past fifteen years. We follow the evolution from Praxis first version to the current one, discussing what was introduced in each. For past and current versions, we classify model improvements, discussing their nature and rationale, derived from received feedback.**

***Keywords*—process; model-driven development; framework; reuse; model transformations; CRUD transactions; persistent data**

I. INTRODUCTION

According to the CMMI [1], a defined software development process "*has a maintained process description, and contributes process related experiences to the organizational process assets*". By **process framework** we define a set of artifacts which includes process descriptions and other important kinds of assets, such as reusable libraries and guidance resources.

Such framework is defined as **model-driven** when models are its core artifacts, from which others are partially or completely derived. In this work, we describe how a model-driven framework evolved along fifteen years, through improvements suggested by feedback from both educational and professional applications.

The process framework whose evolution is discussed here is Praxis, whose primary purpose is to support software engineering course projects. As such, it has been used in the last fifteen years to support teaching in software engineering courses.

Moreover, Praxis has been systematically applied and evaluated by the author himself, in industry-oriented, graduate software courses. The results of this kind of application have been discussed elsewhere ([2], [3], [4], [5]). As shown there in more detail, the students in such courses were required to develop small applications using the complete process. Typical courses comprised four software engineering disciplines, with about 30 hours each, and typical student project had a size of about 100 to 150 function points.

Praxis-Synergia, a derived process tailored to real-life projects, has been applied in the development of applications in the range of hundreds to thousands of function points. This application is performed by Synergia, a university-based software engineering laboratory which develops real-life applications under contract, mostly for government organizations ([6], [7]).

The Praxis process has evolved along those years, mostly through feedback from both course and Synergia projects. This paper describes which changes were introduced during those years, as feedback from process use was collected and analyzed.

In section 2, we discuss the goals of process and modeling improvements, proposing a classification for them. In section 3, we present the evolution of a model-driven development process, oriented to support course projects, showing the improvements performed in each version, how such improvements were suggested by feedback from its application, and which kinds of change they caused. In section 4, we discuss current work. Conclusions are drawn in section 5.

II. PROCESS IMPROVEMENTS

A. *Improvement goals*

A major process improvement goal is to make it more **effective**, that is, help projects to accomplish their mission within specified constraints. For real-life projects, this usually means delivering a product with a satisfactory quality level, meeting the product requirements in a provable way. However, even a fully effective process will not allow competitive development if it is not also **efficient**, accomplishing such mission within its market budget and schedule constraints. In the evolution of processes, feedback from process application leads to actions to improve both the process effectiveness and efficiency.

For educational processes, effectiveness also means exercising the knowledge and skills that their application in course projects intends to impart. Efficiency also means keeping those projects within course budgets for time and effort.

B. *Process artifacts*

A software development process aims to produce **executable code**, such as application code and test scripts, together with their **environmental data**, such as database schemata, test data, configuration files, localized text, graphics and other resource files.

A number of other artifacts help delivering such code and data. Some may be generated by a tool, while others may require at least partially manual derivation. Such artifacts include:

- a set of **models**, which describe both the problem to be solved and the proposed solution, using often a graphic language such as UML;
- technical **documents** and **hyper-documents** for human consumption, such as requirements specifications, manual test scripts, visual prototypes, user manuals, on-line user help and hyper-document representations of the models themselves;
- managerial artifacts, such as **plans** and **reports**;
- **logs** where data are recorded, perhaps in a partially or totally automated way, such as work, appraisal and test logs.

C. Modeling improvements

Several kinds of process improvements actions call for changes in artifacts and practices. For the purposes of this discussion, we classify them in the following kinds: **artifacts reorganization**, **transformations streamlining**, **process simplification**, **reuse enhancements** and **guidance enhancements**.

Artifacts reorganization happens when there are structural changes in the set of process artifacts; in a model-driven framework, the most important are changes in its core models. The models in the set may change, or there may be changes in their internals. A case of special interest is **model layering**, where a model or model section is split in layers, such that a layer uses only elements defined in the same layer or in a layer below. Those layers may be mirrored in derived artifacts.

Transformations streamlining may be applied to the model transformations that generate derived artifacts, other models or other sections of the same models. This includes transformations automation, but also cases where transformations remain partially or totally manual. This may happen either because the derivation requires human design or choices, or because automation may not be cost-effective, at least in a given moment. In such case, streamlining helps to perform them in a more systematic and reliable way.

Process simplification means dropping artifacts or artifact sections that do not prove to be actually useful, thus reducing process overhead and making it more agile and efficient. However, the choice of artifacts to drop requires careful analysis, in order to avoid reducing quality assurance, which would cause an increase in rework, and this might erase the agility gains.

Reuse enhancements are actions that aim to promote reuse of both models and derived artifacts. Reuse is often the best way to improve process efficiency [8]. Compared to other ways, such as reduction in process overhead, reduction in project rework and automation through use of more powerful tools, it usually requires more long term investment, but offers larger returns.

Guidance enhancements apply to supporting process artifacts, in order to improve the way developers use the models. They include enhancements to the process descriptions, but also to reference materials, such as process use guides, process standards, artifact templates, application samples and teaching aids.

III. PROCESS EVOLUTION

A. Version 1.0

Praxis version 1.0 appeared in 2000, being fully described in a textbook published in 2001 (in Portuguese). It evolved from a previous process that had been developed in the preceding years, under contract from an industrial customer. From the beginning, it had as primary goal to support course projects, following the concept of Humphrey's processes ([10], [11], [12]). Moreover, it intended to exercise key concepts present in the UML and in the SW-CMM [13]. Its process artifacts were organized as specified by the IEEE software engineering standards, 1993 edition [14].

The process structure was loosely inspired on MBase [15]; it had similarities with other MBase descendants ([16], [17], [18]), but, overall, its lifecycle model was closer to cascade than to spiral. Several IEEE-style documents were its main artifacts; analysis and design UML models were used as means to organize information that should be present in the IEEE documents for requirements, design and test; several kinds of spreadsheets were used as source for management documents. A single analysis model, written using Rational Rose, was provided as an example.

B. Version 2.0

Version 2.0 appeared in 2003, together with the second edition of the textbook. This was the first truly model-driven version, where the analysis and design models had been improved and reorganized to hold all important technical information. The IEEE documents became model derivatives; experience had shown that they tended to be hard quite hard to use and update, for the kind of small applications developed in the course.

Table 1 summarizes the enhancements introduced in this version, classifying them according to the categories introduced in Section II.C, and stating their description and rationale.

TABLE 1. MODELING IMPROVEMENTS IN VERSION 2.0

Category	Description	Rationale
Guidance enhancement	Supply of sample application	Illustrate framework use
Artifact reorganization	Standardized use of Rational Rose views	Separate modeling concerns
Artifact reorganization	Model and code layering	Separate concerns, improve reuse

The sample application provided a full analysis model and its derived requirements specification, but the design model, its derived documents and the application code were only partially implemented, since course projects specified a whole application, but did not have time to implement more than one or two functions. Support was provided for development of Java stand-alone applications with Swing user interfaces; this remained the standard environment for the following versions.

The models used Rational Rose standard views: a use case view specified functional requirements, in the analysis model, and user interface design, in the design model; a logical view modeled problem concepts and structural requirements as conceptual classes, in the analysis model, and internal design classes, in the design model. The modeling tool allowed forward and reverse engineering, between design model and application code.

A layered architecture was adopted for the logical view in the design model and the application code; it used the boundary, control, and entity layers proposed by Jacobson et al. [16], plus a persistence layer which translated persistent data between object-oriented and relational representations, and a system layer, encapsulating environment services. The process lifecycle model became closer to spiral, although the analysis model and its derived specification were still expected to be complete at the end of the second project phase.

C. Version 2.1

Intermediate minor versions of the process framework have the purpose of testing enhancements which, if successful, are definitively adopted in the following major version. Such intermediate versions are not fully documented in the textbook; supplementary material is supplied when they are tested. Version 2.1 introduced important changes, which were tested, evaluated and later retained in Version 3.0. Part of the course projects whose results were analyzed in the published papers ([4], [5], [6], [7]) used Version 2.1. Table 2 summarizes its enhancements.

TABLE 2. MODELING IMPROVEMENTS IN VERSION 2.1

Category	Description	Rationale
Reuse enhancement	Reuse framework for application code and design model	Ease development of common CRUD functions
Reuse enhancement	Application-independent persistence layer	Allow applications focus on problem-oriented code
Process simplification	Migration of data from documents to model attachments	Avoid duplication between model and documents
Process simplification	Migration of data from documents to spreadsheets	Collect data to provide quantitative feedback
Artifact reorganization	Requirements and design prototypes	Supplement models with visual aids
Reuse enhancement	Reuse framework test code and model	Ease test-driven development
Guidance enhancement	Guidance through a process model	Provide structured supplementary information
Process simplification	Chain of management artifacts	Streamline project management

A major difference from Version 2.0 was the introduction of a reuse framework, for artifacts related to implementation: application code, design model, and code for unit tests. This framework provided a persistence layer that used Java reflection to become completely independent from applications, following the design proposed by Ambler [19]. The reuse framework, also called Praxis, provided (mostly abstract) base classes that supported simple CRUD functions (managing persistent objects which contained primitive fields only) and CRUD with a single strong detail (managing persistent objects whose fields might contain collections of other objects with independent lifetime).This reuse framework was influenced by the experience with a similar framework in one of the first industrial applications projects.

One of the published papers ([7]) discusses in detail the benefits brought by the reuse framework. Thanks to it, it became possible to fully implement applications with at least a hundred function points, corresponding to about five CRUD functions, enough to exercise most of the techniques taught in the supported courses.

Non-UML requirements, design and test information, formerly present in the IEEE documents only, were reshaped as attachments to the analysis and design models; the corresponding documents became mere formatted reports, which might be mechanically extracted from the models. Thereafter, they were gradually dropped from the course projects. The IEEE documents remained in the sample application, however, to illustrate how they might look; the professional variant, Praxis-Synergia, automated the extraction of the IEEE requirements specification, since most clients required it as a contractual reference.

All the management artifacts became spreadsheets; IEEE documents were replaced by spreadsheets with the same content. Their focus shifted from mere fulfillment of IEEE standards, to become means for collection of useful size, work and quality data, providing quantitative feedback to process evolution.

The framework included support for the creation of low-fidelity requirements prototypes and high-fidelity design prototypes. The sample application used a spreadsheet for the requirements prototype, and a technical drawing tool for the design prototype, both generating HTML. Prototypes did not add information to the models, but provided visual feedback, especially to end users.

Test-driven development began to be used in this version, using JUnit [20] scripts to drive and test each application layer. True system tests, acting on actual user interfaces, were not used, because the then available tool used a non-standard script language. However, they were simulated, in a somewhat contrived way, by JUnit tests that exercised fields and commands in the boundary layer. The reuse framework included test script base classes containing most test procedures logic, allowing their specializations to focus on providing application-specific test data and comparing actual against expected application results.

Since the course textbook did not change for this process release, supplementary process information was supplied to the pilot classes, as an UML **process model**. This represented the process itself as use cases and classes, using Rational stereotypes for business process modeling. As with the other models, required non-UML information was supplied by model attachments.

Management artifacts were organized in a chain of derivation that started in a requirements database maintained using the Rational RequisitePro tool. It kept trace relationships from the primary requirements, expressed by analysis model use cases and persistent classes, to derived items in both models, using the integration with Rational Rose provided by that tool. For the primary requirements, this database held also function point counts and their rationale. Extracted functional size reports fed the estimates performed by project planning spreadsheets. The planning artifacts fed project control reports, which compared expected and actual project performance.

D. Version 3.0

The third and current edition of the textbook reflected new or upgraded relevant software standards, such as UML 2.0 [21], CMMI [1], the 2003 collection of IEEE software standards [22], PMBoK [23] and SPEM 2.0 [24]; UML 2.0 and SPEM deeply affected modeling. Most of the practices of the Extreme Programming [25] agile methodology were adopted; however, models remained in the framework core, unlike XP and more like Agile Modeling [26].

The adoption of such standards in the process aimed to give Praxis users the opportunity to use in practice some of the most important standards then available. In fact, all of those standards have had only minor revisions and improvements, to this date. Table 3 summarizes its enhancements.

TABLE 3. MODELING IMPROVEMENTS IN VERSION 3.0

Category	Description	Rationale
Artifact reorganization	Models migration to Eclipse	Support UML 2.0; sunset of former tool
Guidance enhancement	Process models migration to EPF	Support SPEM; richer on-line documentation
Transformation streamlining	Rich stereotype profile	Embed more data in the models
Transformation streamlining	XML attachments	Ease transformations and visualization
Artifact reorganization	Using activities to model scenarios	Improved use case modeling
Artifact reorganization	Partitioning models into views	Matching models to process steps
Reuse enhancement	Partitioning into framework and product levels	Organize reusable elements in a framework
Artifact reorganization	Internal view layering	Separate architecture, structure and behavior
Artifact reorganization	Layering test view and code	Ease use through separation of concerns
Reuse enhancement	Migration of persistence layer to Hibernate	More powerful persistence. using free components
Process simplification	Improve chain of management artifacts	Artifact streamlining; better data quality assurance

To adopt UML 2.0, and because sunset of Rational Rose was expected, models migrated to IBM Rational Architect, embedded in Eclipse. The vendor-provided conversion tool offered limited help, since UML 2.0 brought useful new modeling facilities to, such as richer sequence diagrams to model interactions. Constructs such as selections, iterations and use references could be formally documented, and more formal definition of specialization helped to reuse collaborations and use cases.

SPEM allowed much better documentation of the process itself. Most of the process model migrated to EPF Composer [27], providing better on-line process reference. However, a smaller UML business model was kept, to document structural relationships among process concepts, not well supported by EPF, which focuses on representation of process dynamic.

UML 2.0 and the Rational Architect provided rich support for stereotype profiles. In the previous version, stereotypes had the limited purpose of providing visual representations, to enhance diagrams clarity. Now the Praxis profile was developed, allowing models to use UML tagged values (called stereotype properties, in the modeling tool) to hold much of the requirements and design information, formerly kept in model attachments.

Those properties provided readier access to both human users and tool extensions, especially when a set of scalar data was associated to a single UML element. For instance, details of I/O requirements were kept in stereotypes for boundary classes; business rules, use case preconditions and post conditions, non-functional requirements, design rules and design decisions were kept in stereotyped UML constraints; persistence requirements and design data went to stereotyped persistent classes.

On the other hand, collections of data associated to sets of UML elements were kept as attachments, since in this case the bare modeling tool was not easy to use. The tool provided an API for Java plug-ins, more convenient than the Microsoft OLE model used in the previous generation. This was used by the professional Praxis-Synergia variant, to provide visual support to more complex data extraction facilities, such as function-point counting ([9], [28]), and generation of requirements specifications and requirements and design prototypes [9].

The development of plug-ins was deemed too expensive for the educational version. Instead, much of attachments migrated to XML, using XSL style sheets to provide visual representations. Spreadsheets remained in use for management artifacts were calculations were required. In the standard version, the prototypes in the sample application were created directly in HTML, using an HTML visual editor. In other cases, such as test data and user messages, a simple conversion transformed XML attachments into Java property files, queried by the application at run-time.

Richer support and formalization of use cases allowed the replacement of the text attachments that described use cases scenarios by activity diagrams. Better formalization of use case specialization and the use of UML 2.0 elements improved use case modeling.

Praxis retained a sharp distinction between modeling **what to do** (problem specification, corresponding to the Requirements and Analysis disciplines) and **how to do it** (solution design, corresponding to Design, Test and Implementation). Such distinction is not in other model-driven, transformation-based proposals, such as AndroMDA [29], Jarzabek and Trung [30], and Mashkoor and Fernandes [31]. Indeed, Praxis models names changed to **Problem model** and **Solution model**, to emphasize this distinction. The Problem model should be technology-independent and sole source for problem complexity measures, such as function point counts.

To a given Problem model might correspond several Solution models, if several solution are developed, using different architectures and technologies.

Instead of the fixed major divisions imposed by Rational Rose, the new tool allowed partitioning the models in **views**, sections corresponding to the steps followed in the development of each function. In the Problem model, the **Requirements** view models requirements at a user-oriented, higher level, using use cases for procedural descriptions of the required functions, and constraints for business rules and non-functional requirements. The **Analysis** view uses conceptual-level classes to hold more detailed requirements, such as required I/O fields and commands, and persistence requirements; collaborations of those classes must realize the use cases in a convincing way. In the professional practice, the requirements view is built during JAD-style workshops (as described by McConnell [32]), reconciling perhaps conflicting requirements of different users, while the analysis view reflects detailed interviews conducted with individual users.

The Solution model has a **Use** view, to model the external product design, that is, its user interfaces and interactions of those with the user and among themselves (such as navigation and changes in appearance). This view expresses design decisions which must match the problem requirements, but include consideration of usability, architecture and implementation issues, for a given technology. Few other methodologies provide that kind of view, and still fewer use UML models, such as RUP-UX [33], but the use of the professional variant has proved it to be one of the most useful applications of models. In that variant, a plug-in allows automated generation of visual design prototypes from the use view, allowing prototype and model to keep synchronized.

Other Solution model views are the **Test** view, which derives from the use view a set of test model elements, from which manual and automated test scripts and data may be generated; and a **Logical** view, which represents internal application design. Forward and reverse engineering facilities provided by the modeling tool are used to keep the last two views synchronized, respectively, with test and application code.

Layering was now performed at three levels. In a first level, reuse was supported by dividing the framework into a **framework level**, containing reusable models, mostly composed by abstract classes, use cases and collaborations; reusable code libraries, matched to the test and logical views of the framework-level Solution model; and reusable artifacts such as XSD schemata, XSL style sheets and spreadsheet templates.

In a second level, each view had an **architecture** section, where requirements and design rules and decision were expressed as stereotyped constraints; a **structure** section, where those constraints were attached to classes, attributes, operations and relationships; and a **behavior** section, where those classes participated in **collaborations**, containing interactions derived from use case scenarios, in a continuous chain.

The third level was used to partition some views in layers that matched code layers. The Analysis view has boundary, control and entity class layers; the Logical view contains additional persistence and system layers.

Fully automated functional tests were introduced, using IBM Rational Functional Tester, which provides Java test scripts, tightly integrated with Eclipse. To ease the use of its somewhat complex API, and provide some degree of technology independence, the test model view and code were also layered and employed reuse. A **common** layer, shared by all kinds of tests, holds test data, represented by classes whose instances contains **test case** data, with similar test cases sharing **test entities**.

Above the common layer, test view and code split in a **black-box** layer, containing system tests, and a **gray-box** layer, containing unit tests for the application entity and control layers. Boundary layer unit tests were no longer used, since, by process guideline, this layer must handle presentation only, delegating functionalities such as field validation to the layers below. To further tame complexity and provide separation of concerns and technology independence, the black-box layer uses three layers of classes: **test inspectors** move data in and out of the user interfaces; **test checkers** compare expected and actual results; and **test procedures** implement the interactions in test collaborations.

Persistence handling underwent a significant change. It was decided to adopt Hibernate [34], instead of improving the previous specific object-to-relational translating mechanism, which had very limited capabilities. This decision was based both on the good results of Hibernate adoption in the professional version, and the acceptance of that tool by the market, especially with the JPA API, much easier and more convenient that then previous Java EJB persistence mechanism. However, to allow upper layers to retain a simple view of persistency, based on specialization of a `PersistentObject` class, JPA calls were encapsulated in in a thin façade layer that offered the same API as the layer in the previous version.

Very few changes were needed in the entity and control layers, mostly to allow for a few collateral effects of the way Hibernate handles its persistence cache. If the relational table names adhere to Hibernate defaults, a minimum of JPA persistent annotations has to be present in the application code, just to mark persistent classes and their methods which return persistent collections.

In this version, the chain of management artifacts continues to start in the requirements database, since this might also be integrated with Architect. Most changes aimed to streamline them further, while a keeping some redundancy among artifacts, to provide consistency checks for the collected data. In several cases, this redundancy allowed detection of incorrect and even faked data, which incurred heavy penalties during course projects grading.

A significant improvement was the adoption of COCOMO [35] for the estimation and planning of project work. Although somewhat old and hard to integrate with the remaining process tools, COCOMO has proved itself very useful to this date.

E. *Version 3.5*

TABLE 4. MODELING IMPROVEMENTS IN VERSION 3.5

Category	Description	Rationale
Transformation streamlining	Richer stereotype profile	Ease of data extraction via reports
Transformation streamlining	Match XML files to stereotypes	Systematic extraction of complex data
Reuse enhancement	Framework-level Problem model	Promote fitting requirements to reuse
Reuse enhancement	Richer CRUD patterns	Support richer variation in reuse

Version 3.5 was developed as a stepping stone to Version 4.0. This version was the first international edition of the framework: all artifacts were translated to English, as well as the user interfaces of the sample application, using the Eclipse support of Java externalized strings. Table 4 summarizes its enhancements.

The framework stereotype profile was enhanced to ease the extraction of derived artifacts. These are useful in the professional environment, easing models use by large teams of developers, many of which not highly UML-proficient, as our experience with industrial projects has shown. For the educational version, simpler data sets are extracted using BIRT, a report generator provided by Eclipse.

Generation of problem-level prototypes required more complex data extraction; for this, stereotyped properties structure match an external XML representation, for which XSL style sheets provide just-in-time HTML generation. This allowed a still manual, but very systematic matching between prototype and model.

In the previous versions, there was no framework-level Problem model, since this was much smaller and simpler that the Solution model. Crude reuse might be performed with copy-and-paste from the sample application. However, a framework-level Problem model was introduced in this version, building on the experience from the professional version. Real-life projects have suffered from low reuse that causes loss of productivity. Reuse of Problem model elements might help the requirements engineers to try and fit user-required functions into standardized framework-supported patterns, allowing the provider to charge less for those functions.

In the educational version, it is expected that this would guide the students to fit the requirements proposed for their projects into reuse patterns. In past projects, sometimes students found out that their originally proposed functions were too hard to implement, using the existing patterns. In such cases, they were allowed to change the requirements, but some time and work had already been wasted before they realized this.

In most cases, implementation difficulties were found with functions that did not fit the patterns present in the Solution model and code; that is, CRUD with a single strong detail. For other kinds of detail, the sample application provided some example functions, but adapting them was much more difficult than reusing the framework. The current version provides support for multiple detail collections, including weak ones (data whose lifetime is bound by the master instance lifetime). Handling such collections builds on UML parameterized classes, in the Solution model, and Java generics, in the code.

IV. CURRENT STATUS

A. *Version 4.0*

Currently, Version 4.0 is under development. This version is not very different from Version 3.5, perhaps reflecting stability of the framework. Most improvements were minor; Table 5 shows the few major ones.

TABLE 5. MODELING IMPROVEMENTS IN VERSION 4.0

Category	Description	Rationale
Reuse enhancement	Simplified boundary layer as prototype	Design prototype becomes way to boundary development
Artifacts reorganization	Support for Selenium tests	System test more similar to unit test, in open-source environment
Guidance enhancements	Support for Vaadin boundary layer	Many web-oriented applications

For solution-level prototypes, it is possible to use HTML prototypes, as done for problem-level. However, it was found in former versions that a significant amount of Javascript code is needed to have a prototype with significant behavior. Reuse of prototype common elements reduces the amount of Javascript needed for each new prototype, but a different solution was tried and accepted for this version.

Several difficulties found with the IBM Rational Functional Tester tool prompted us to go back to Junit-based system tests. By now, system-level testing in a JUnit environment had become much more powerful with the appearance of tools such as Selenium [36], supplemented by JUnitParams [37].

With our new approach, a preliminary version of the boundary layer is used as the solution-level prototype. Only minor modifications are then needed to change that into the definitive code. Some of the differences between the two versions correspond to differences between actual and simulated control and entity layers; some stem from validity checks that are too heavy for prototypes; some correspond to features that only the actual version allows being thoroughly tested; and some are code optimizations that should be done in the final code only. Differences usually amount to less than 5% of the code.

Also, an additional version of the boundary layer infrastructure was provided to support web-based applications that use Vaadin [38] components. This adds to the existing support for local, Swing-based interface components.

B. *Some examples*

In a Problem model, Fig. 1 shows the scenarios for a complex use in an application. However, no specific event flow steps are necessary for any flow. These are all described in the CRUD abstract use case, which the Program Management concrete use case specializes, as shown in Fig. 2. Only stereotyped properties need be instanced; an example for a scenario is shown in Fig. 3. In this case, the scenario instantiates the event flow shown in Fig. 4.

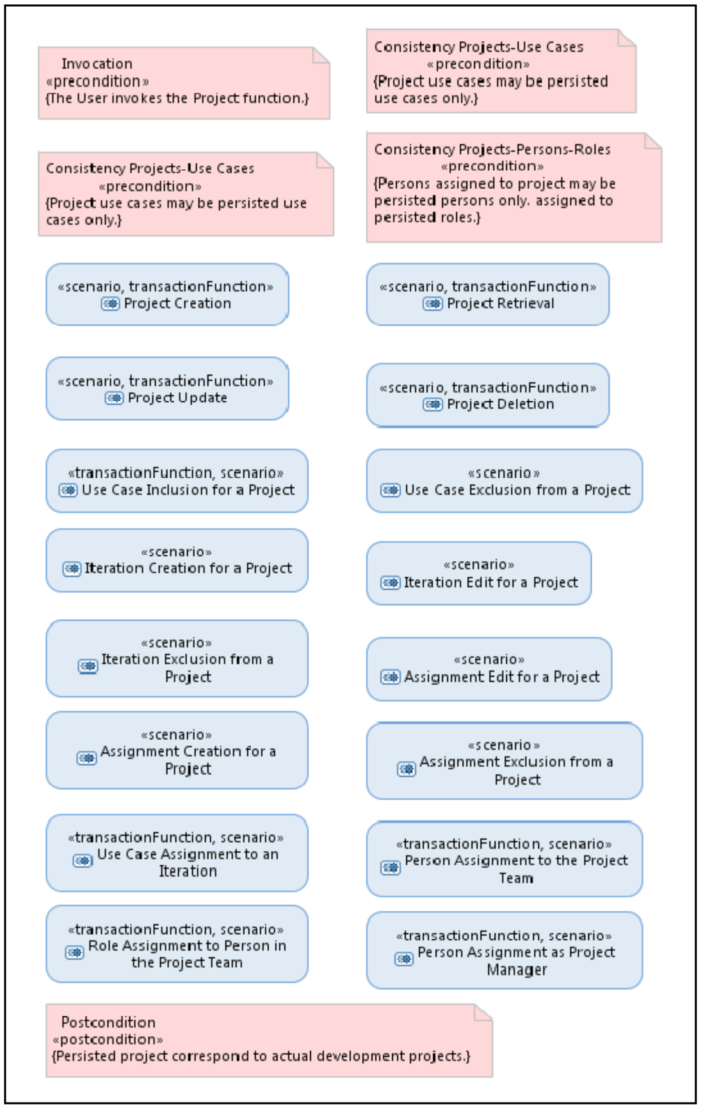

Invocation
«precondition»
{The User invokes the Project function.}
Consistency Projects-Use Cases
«precondition»
{Project use cases may be persisted use cases only.}
Consistency Projects-Use Cases
«precondition»
{Project use cases may be persisted use cases only.}
Consistency Projects-Persons-Roles
«precondition»
{Persons assigned to project may be persisted persons only, assigned to persisted roles.}
«scenario, transactionFunction» Project Creation
«scenario, transactionFunction» Project Retrieval
«scenario, transactionFunction» Project Update
«scenario, transactionFunction» Project Deletion
«transactionFunction, scenario» Use Case Inclusion for a Project
«scenario» Use Case Exclusion from a Project
«scenario» Iteration Creation for a Project
«scenario» Iteration Edit for a Project
«scenario» Iteration Exclusion from a Project
«scenario» Assignment Edit for a Project
«scenario» Assignment Creation for a Project
«scenario» Assignment Exclusion from a Project
«transactionFunction, scenario» Use Case Assignment to an Iteration
«transactionFunction, scenario» Person Assignment to the Project Team
«transactionFunction, scenario» Role Assignment to Person in the Project Team
«transactionFunction, scenario» Person Assignment as Project Manager
Postcondition
«postcondition»
{Persisted project correspond to actual development projects.}

Fig. 1 Requirements for a use case

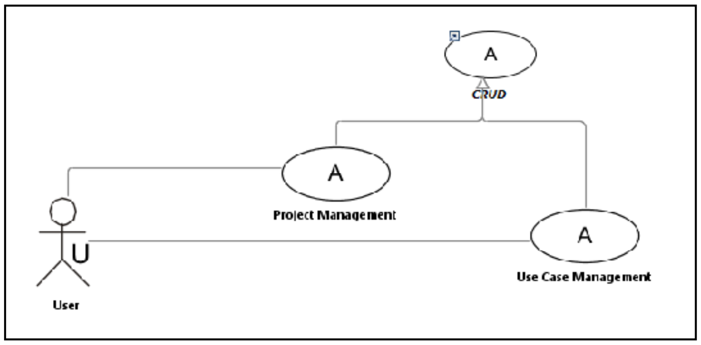
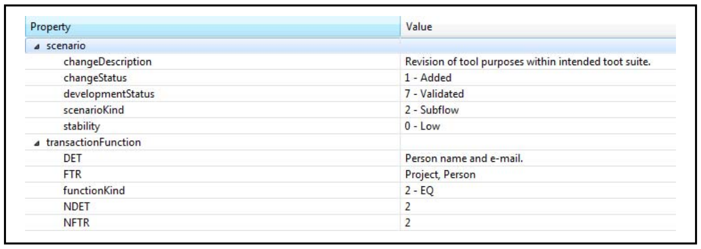


Fig. 2 Sample use cases



Property	Value
scenario	
changeDescription	Revision of tool purposes within intended toot suite.
changeStatus	1 - Added
developmentStatus	7 - Validated
scenarioKind	2 - Subflow
stability	0 - Low
transactionFunction	
DET	Person name and e-mail.
FTR	Project, Person
functionKind	2 - EQ
NDET	2
NFTR	2

Fig. 3 Stereotype properties for a scenario

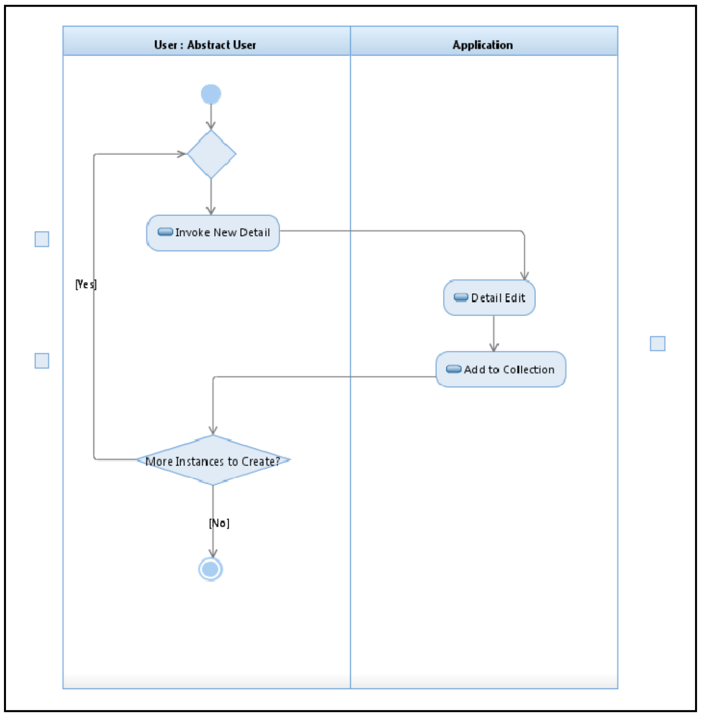


Fig. 4 A reusable scenario

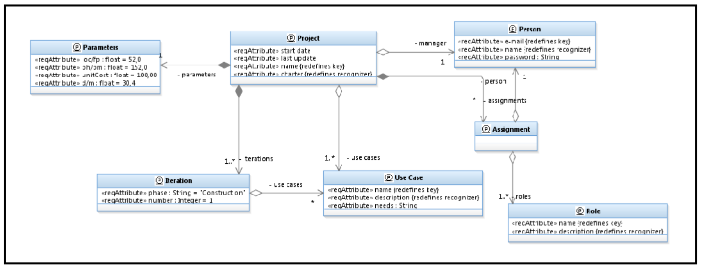

Parameters
«reqAttribute» oc/fp : float = 52,0
«reqAttribute» ph/pm : float = 152,0
«reqAttribute» unitCost : float = 100,00
«reqAttribute» d/m : float = 30,4
- parameters
Project
«reqAttribute» start date
«reqAttribute» last update
«reqAttribute» name {redefines key}
«reqAttribute» charter {redefines recognizer}
- manager
1
Person
«recAttribute» e-mail {redefines key}
«recAttribute» name {redefines recognizer}
«recAttribute» password : String
- person
- assignments
Assignment
1..* - iterations
1..* - use cases
Iteration
«reqAttribute» phase : String = "Construction"
«reqAttribute» number : Integer = 1
- use cases
Use Case
«reqAttribute» name {redefines key}
«reqAttribute» description {redefines recognizer}
«reqAttribute» needs : String
1..* - roles
Role
«recAttribute» name {redefines key}
«recAttribute» description {redefines recognizer}

Fig. 5 Problem model entity layer

The Problem model is completed by an Analysis view; samples for the entity and boundary layer are shown in Fig.5 and Fig. 6, respectively. For the entity layer, all shown classes inherit from a `PersistentEntity` framework class, which means they represent information persisted in the database.

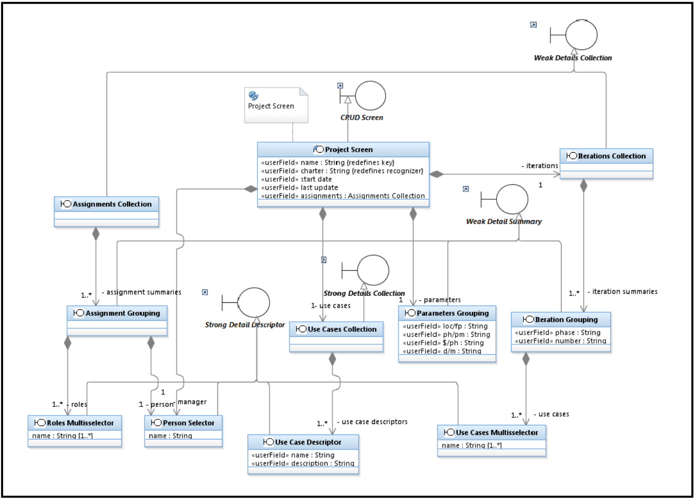
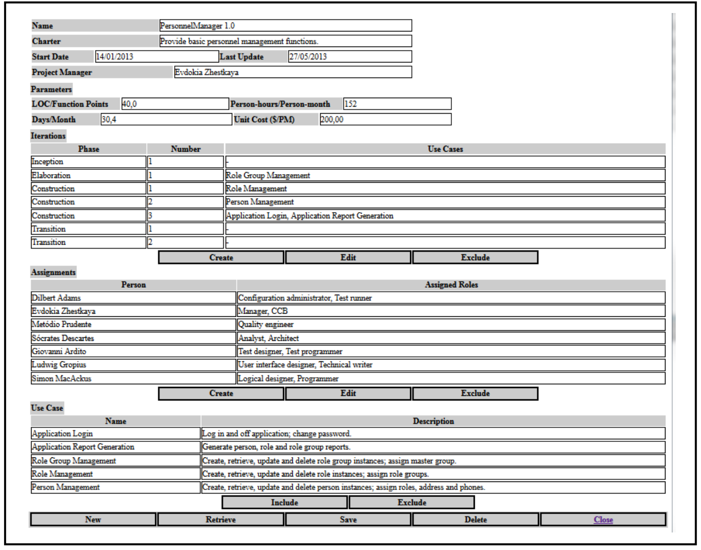


Fig. 6 Problem model boundary layer



Name	PersonnelManager 1.0		
Charter	Provide basic personnel management functions.		
Start Date	14/01/2013	**Last Update**	27/05/2013
Project Manager	Evdokia Zhestkaya		

Parameters

LOC/Function Points	40,0	**Person-hours/Person-month**	152
Days/Month	30,4	**Unit Cost ($/PM)**	200,00

Iterations

Phase	Number	Use Cases
Inception	1	-
Elaboration	1	Role Group Management
Construction	1	Role Management
Construction	2	Person Management
Construction	3	Application Login, Application Report Generation
Transition	1	-
Transition	2	-

Create Edit Exclude

Assignments

Person	Assigned Roles
Dilbert Adams	Configuration administrator, Test runner
Evdokia Zhestkaya	Manager, CCB
Metódio Prudente	Quality engineer
Sócrates Descartes	Analyst, Architect
Giovanni Ardito	Test designer, Test programmer
Ludwig Gropius	User interface designer, Technical writer
Simon MacAckus	Logical designer, Programmer

Create Edit Exclude

Use Case

Name	Description
Application Login	Log in and off application; change password.
Application Report Generation	Generate person, role and role group reports.
Role Group Management	Create, retrieve, update and delete role group instances; assign master group.
Role Management	Create, retrieve, update and delete role instances; assign role groups.
Person Management	Create, retrieve, update and delete person instances; assign roles, address and phones.

Include Exclude

New Retrieve Save Delete Close

Fig. 7 Application interface prototype

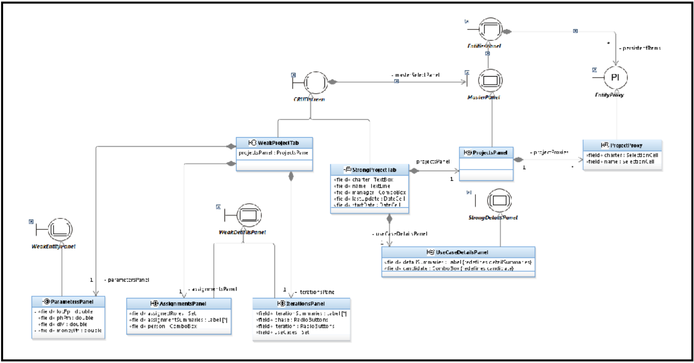
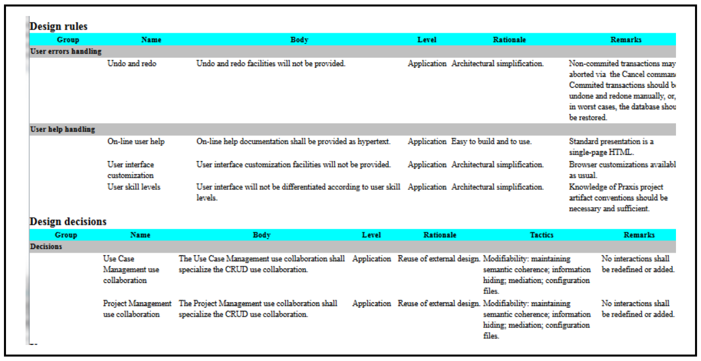

EntitiesPanel
- persistentItems
- masterSelectPanel
CRUDScreen
MasterPanel
PI
EntityProxy
WeakProjectTab
projectsPanel : ProjectsPanel
StrongProjectTab
«field» charter : TextBox
«field» name : TextLine
«field» manager : ComboBox
«field» lastupdate : DateCell
«field» startDate : DateCell
projectsPanel
1
ProjectsPanel
- projectProxies
1
ProjectProxy
«field» charter : SelectionCell
«field» name : SelectionCell
StrongDetailsPanel
WeakDetailsPanel
WeakEntityPanel
- useCaseDetailsPanel
1
UseCaseDetailsPanel
«field» detailSummaries : Label {redefines detailSummaries}
«field» candidate : ComboBox {redefines candidate}
1
- parametersPanel
1
- assignmentsPanel
1
- iterationsPanel
ParametersPanel
«field» locFp : double
«field» phPm : double
«field» dM : double
«field» moneyPh : double
AssignmentsPanel
«field» assignedRoles : Set
«field» assignmentSummaries : Label [*]
«field» person : ComboBox
IterationsPanel
«field» iterationSummaries : Label [*]
«field» phase : RadioButtons
«field» iteration : RadioButtons
«field» useCases : Set

Fig. 8 The Use view of the Solution model

Fig. 7 shows the prototype that corresponds to the model in Fig. 6. In the Solution model, the Use view, shown in Fig. 8. This presents a view of the user interface that is closer to reality than the Problem model view shown in Fig. 6.



Design rules

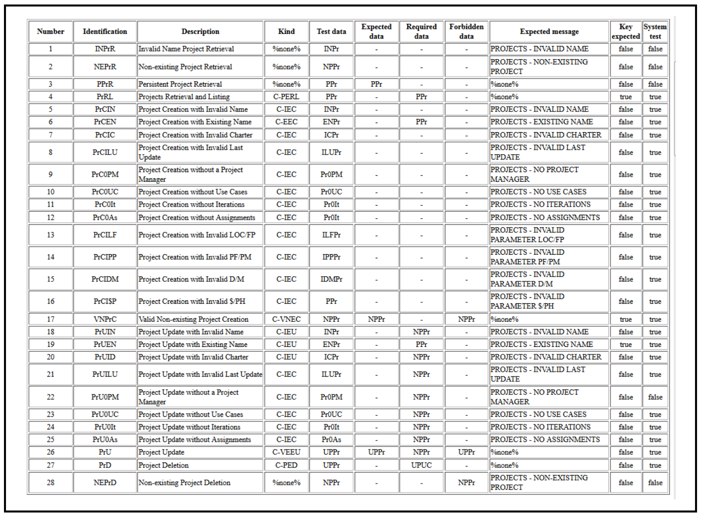
Group	Name	Body	Level	Rationale	Remarks
User errors handling					
	Undo and redo	Undo and redo facilities will not be provided.	Application	Architectural simplification.	Non-commited transactions may aborted via the Cancel comman Commited transactions should b undone and redone manually, or, in worst cases, the database shou be restored.
User help handling					
	On-line user help	On-line help documentation shall be provided as hypertext.	Application	Easy to build and to use.	Standard presentation is a single-page HTML.
	User interface customization	User interface customization facilities will not be provided.	Application	Architectural simplification.	Browser customizations availabl as usual.
	User skill levels	User interface will not be differentiated according to user skill levels.	Application	Architectural simplification.	Knowledge of Praxis project artifact conventions should be necessary and sufficient.

Design decisions

Group	Name	Body	Level	Rationale	Tactics	Remarks
Decisions						
	Use Case Management use collaboration	The Use Case Management use collaboration shall specialize the CRUD use collaboration.	Application	Reuse of external design.	Modifiability: maintaining semantic coherence; information hiding; mediation; configuration files.	No interactions shall be redefined or added.
	Project Management use collaboration	The Project Management use collaboration shall specialize the CRUD use collaboration.	Application	Reuse of external design.	Modifiability: maintaining semantic coherence; information hiding; mediation; configuration files.	No interactions shall be redefined or added.

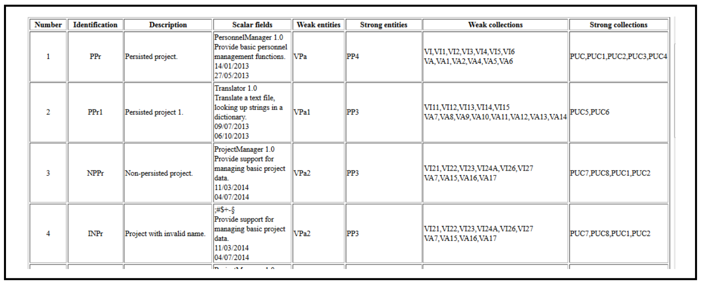
Fig. 9 Design rules and decisions

Fig. 9 shows an example of the documented design rules and decisions that are attached to the Use view.



Number	Identification	Description	Kind	Test data	Expected data	Required data	Forbidden data	Expected message	Key expected	System test
1	INPrR	Invalid Name Project Retrieval	%none%	INPr	-	-	-	PROJECTS - INVALID NAME	false	false
2	NEPrR	Non-existing Project Retrieval	%none%	NPPr	-	-	-	PROJECTS - NON-EXISTING PROJECT	false	false
3	PPrR	Persistent Project Retrieval	%none%	PPr	PPr	-	-	%none%	false	false
4	PrRL	Projects Retrieval and Listing	C-PERL	PPr	-	PPr	-	%none%	true	true
5	PrCIN	Project Creation with Invalid Name	C-IEC	INPr	-	-	-	PROJECTS - INVALID NAME	false	true
6	PrCEN	Project Creation with Existing Name	C-EEC	ENPr	-	PPr	-	PROJECTS - EXISTING NAME	false	true
7	PrCIC	Project Creation with Invalid Charter	C-IEC	ICPr	-	-	-	PROJECTS - INVALID CHARTER	false	true
8	PrCILU	Project Creation with Invalid Last Update	C-IEC	ILUPr	-	-	-	PROJECTS - INVALID LAST UPDATE	false	true
9	PrC0PM	Project Creation without a Project Manager	C-IEC	Pr0PM	-	-	-	PROJECTS - NO PROJECT MANAGER	false	true
10	PrC0UC	Project Creation without Use Cases	C-IEC	Pr0UC	-	-	-	PROJECTS - NO USE CASES	false	true
11	PrC0It	Project Creation without Iterations	C-IEC	Pr0It	-	-	-	PROJECTS - NO ITERATIONS	false	true
12	PrC0As	Project Creation without Assignments	C-IEC	Pr0It	-	-	-	PROJECTS - NO ASSIGNMENTS	false	true
13	PrCILF	Project Creation with Invalid LOC/FP	C-IEC	ILFPr	-	-	-	PROJECTS - INVALID PARAMETER LOC/FP	false	true
14	PrCIPP	Project Creation with Invalid PF/PM	C-IEC	IPPPr	-	-	-	PROJECTS - INVALID PARAMETER PF/PM	false	true
15	PrCIDM	Project Creation with Invalid D/M	C-IEC	IDMPr	-	-	-	PROJECTS - INVALID PARAMETER D/M	false	true
16	PrCI$P	Project Creation with Invalid $/PH	C-IEC	PPr	-	-	-	PROJECTS - INVALID PARAMETER $/PH	false	true
17	VNPrC	Valid Non-existing Project Creation	C-VNEC	NPPr	NPPr	-	NPPr	%none%	true	true
18	PrUIN	Project Update with Invalid Name	C-IEU	INPr	-	NPPr	-	PROJECTS - INVALID NAME	false	true
19	PrUEN	Project Update with Existing Name	C-IEU	ENPr	-	PPr	-	PROJECTS - EXISTING NAME	true	true
20	PrUID	Project Update with Invalid Charter	C-IEU	ICPr	-	NPPr	-	PROJECTS - INVALID CHARTER	false	true
21	PrUILU	Project Update with Invalid Last Update	C-IEC	ILUPr	-	NPPr	-	PROJECTS - INVALID LAST UPDATE	false	true
22	PrU0PM	Project Update without a Project Manager	C-IEC	Pr0PM	-	NPPr	-	PROJECTS - NO PROJECT MANAGER	false	false
23	PrU0UC	Project Update without Use Cases	C-IEC	Pr0UC	-	NPPr	-	PROJECTS - NO USE CASES	false	true
24	PrU0It	Project Update without Iterations	C-IEC	Pr0It	-	NPPr	-	PROJECTS - NO ITERATIONS	false	true
25	PrU0As	Project Update without Assignments	C-IEC	Pr0As	-	NPPr	-	PROJECTS - NO ASSIGNMENTS	false	true
26	PrU	Project Update	C-VEEU	UPPr	UPPr	NPPr	UPPr	%none%	false	true
27	PrD	Project Deletion	C-PED	UPPr	-	UPUC	-	%none%	false	true
28	NEPrD	Non-existing Project Deletion	%none%	NPPr	-	-	NPPr	PROJECTS - NON-EXISTING PROJECT	false	false

Fig. 10 Test cases



Number	Identification	Description	Scalar fields	Weak entities	Strong entities	Weak collections	Strong collections
1	PPr	Persisted project.	PersonnelManager 1.0 Provide basic personnel management functions. 14/01/2013 27/05/2013	VPa	PP4	VI,VI1,VI2,VI3,VI4,VI5,VI6 VA,VA1,VA2,VA4,VA5,VA6	PUC,PUC1,PUC2,PUC3,PUC4
2	PPr1	Persisted project 1.	Translator 1.0 Translate a text file, looking up strings in a dictionary. 09/07/2013 06/10/2013	VPa1	PP3	VI11,VI12,VI13,VI14,VI15 VA7,VA8,VA9,VA10,VA11,VA12,VA13,VA14	PUC5,PUC6
3	NPPr	Non-persisted project.	ProjectManager 1.0 Provide support for managing basic project data. 11/03/2014 04/07/2014	VPa2	PP3	VI21,VI22,VI23,VI24A,VI26,VI27 VA7,VA15,VA16,VA17	PUC7,PUC8,PUC1,PUC2
4	INPr	Project with invalid name.	;#$+-§ Provide support for managing basic project data. 11/03/2014 04/07/2014	VPa2	PP3	VI21,VI22,VI23,VI24A,VI26,VI27 VA7,VA15,VA16,VA17	PUC7,PUC8,PUC1,PUC2

Fig. 11 Test entities

Fig. 10 shows an attachment to the Test view that displays the test cases that a system test must perform for validation. The data used by the tests are referred in tables, a small part of which is shown in Fig. 11. The data shown here describe sample projects, whose details are defined by other described in the same way, which are part of each project collections. Strong collections are those that survive the project existence and are therefore managed in other pages, while weak collections do not and are wholly managed within the project.

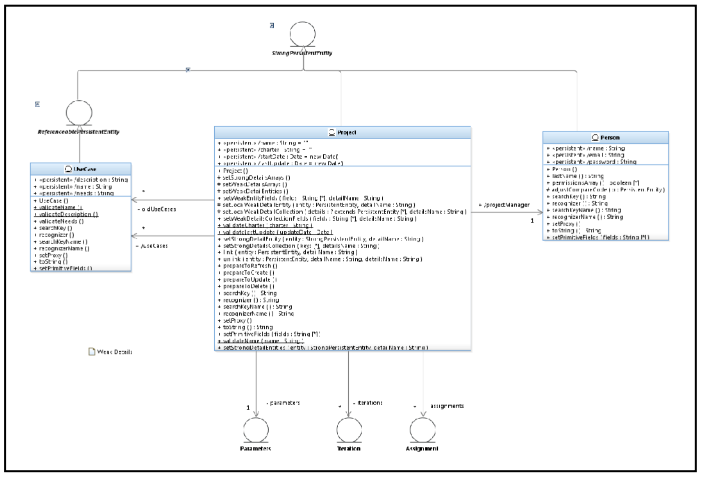
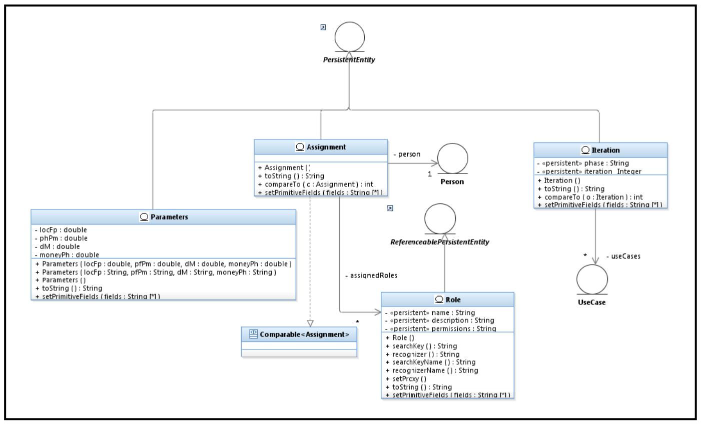
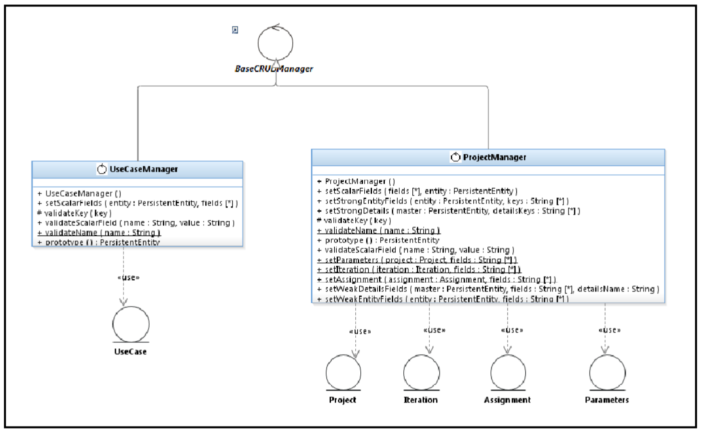


Fig. 12 Logical model – strong entities



PersistentEntity
Assignment
+ Assignment ()
+ toString () : String
+ compareTo (c : Assignment) : int
+ setPrimitiveFields (fields : String [*])
- person
Person
Iteration
- «persistent» phase : String
- «persistent» iteration : Integer
+ Iteration ()
+ toString () : String
+ compareTo (o : Iteration) : int
+ setPrimitiveFields (fields : String [*])
Parameters
- locFp : double
- pfPm : double
- dM : double
- moneyPh : double
+ Parameters (locFp : double, pfPm : double, dM : double, moneyPh : double)
+ Parameters (locFp : String, pfPm : String, dM : String, moneyPh : String)
+ Parameters ()
+ toString () : String
+ setPrimitiveFields (fields : String [*])
ReferenceablePersistentEntity
- assignedRoles
- useCases
UseCase
Comparable<Assignment>
Role
- «persistent» name : String
- «persistent» description : String
- «persistent» permissions : String
+ Role ()
+ searchKey () : String
+ recognizer () : String
+ searchKeyName () : String
+ recognizerName () : String
+ setProxy ()
+ toString () : String
+ setPrimitiveFields (fields : String [*])

Fig. 13 Logical model – weak entities



BaseCRUDManager
UseCaseManager
+ UseCaseManager ()
+ setScalarFields (entity : PersistentEntity, fields [*])
# validateKey (key)
+ validateScalarField (name : String, value : String)
+ validateName (name : String)
+ prototype () : PersistentEntity
ProjectManager
+ ProjectManager ()
+ setScalarFields (fields [*], entity : PersistentEntity)
+ setStrongEntityFields (entity : PersistentEntity, keys : String [*])
+ setStrongDetails (master : PersistentEntity, detailsKeys : String [*])
# validateKey (key)
+ validateName (name : String)
+ prototype () : PersistentEntity
+ validateScalarField (name : String, value : String)
+ setParameters (project : Project, fields : String [*])
+ setIteration (iteration : Iteration, fields : String [*])
+ setAssignment (assignment : Assignment, fields : String [*])
+ setWeakDetailsFields (master : PersistentEntity, fields : String [*], detailsName : String)
+ setWeakEntityFields (entity : PersistentEntity, fields : String [*])
«use»
UseCase
Project
Iteration
Assignment
Parameters

Fig. 14 Logical model – control

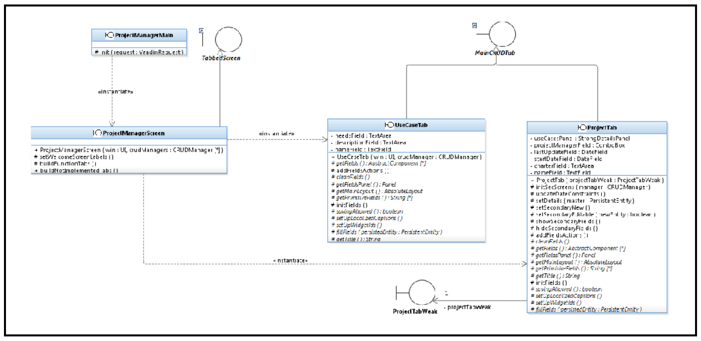


Fig. 15 Logical model – boundary (screens)

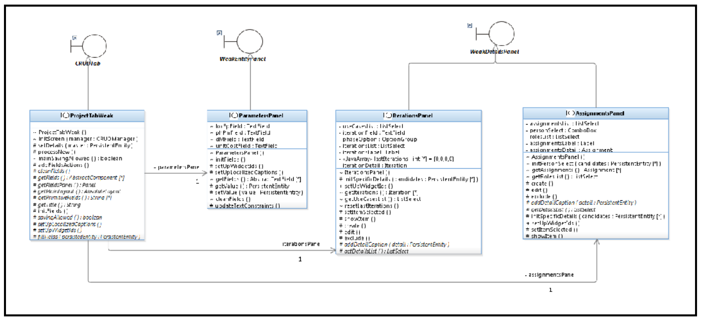
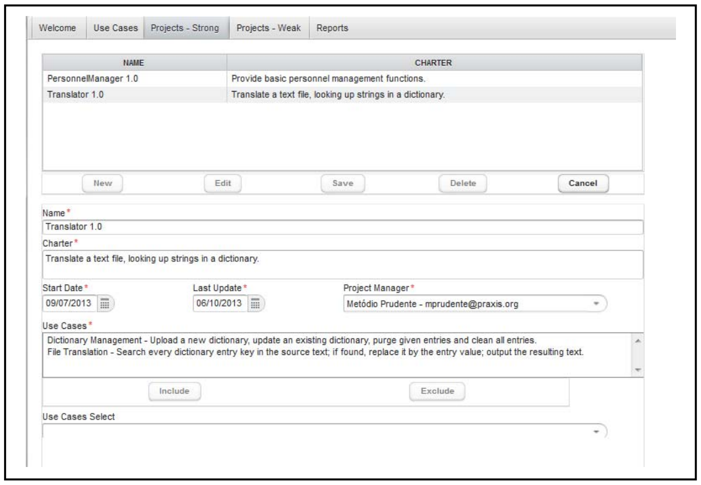


Fig. 16 Logical model – boundary (fields)

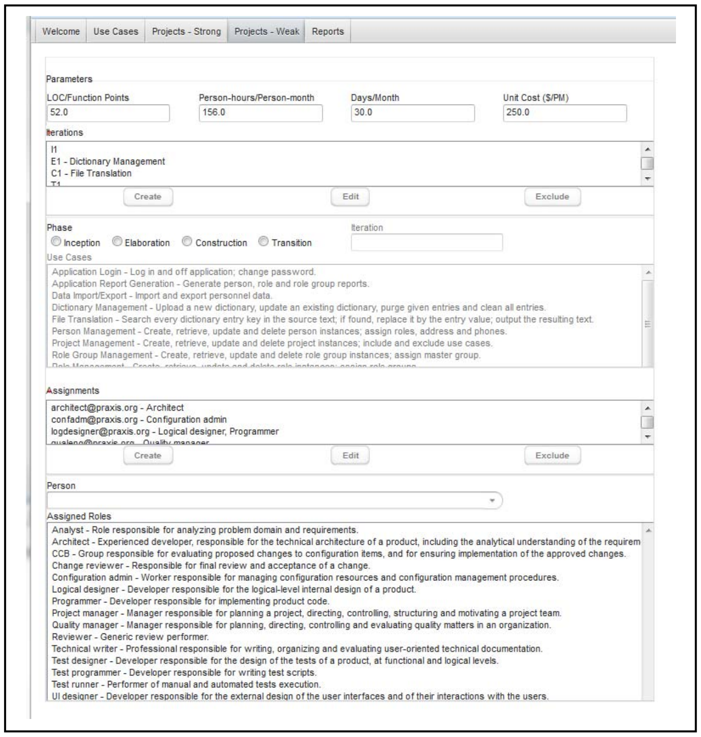
Fig. 12 to Fig. 16 illustrate the resulting application logical model, which is organized into entity, control and boundary layers. Fig. 17 and Fig. 18 show views of one selected project information. Weak details are shown in a separate page, because their information is very extensive.



NAME	CHARTER
PersonnelManager 1.0	Provide basic personnel management functions.
Translator 1.0	Translate a text file, looking up strings in a dictionary.

Name*
Translator 1.0
Charter*
Translate a text file, looking up strings in a dictionary.
Start Date* 09/07/2013
Last Update* 06/10/2013
Project Manager* Metódio Prudente - mprudente@praxis.org
Use Cases*
Dictionary Management - Upload a new dictionary, update an existing dictionary, purge given entries and clean all entries.
File Translation - Search every dictionary entry key in the source text; if found, replace it by the entry value; output the resulting text.
Use Cases Select

Fig. 17 Executed program – main projects page



Parameters
LOC/Function Points 52.0
Person-hours/Person-month 156.0
Days/Month 30.0
Unit Cost ($/PM) 250.0
Iterations
I1
E1 - Dictionary Management
C1 - File Translation
T1
Phase
Inception Elaboration Construction Transition
Iteration
Use Cases
Application Login - Log in and off application; change password.
Application Report Generation - Generate person, role and role group reports.
Data Import/Export - Import and export personnel data.
Dictionary Management - Upload a new dictionary, update an existing dictionary, purge given entries and clean all entries.
File Translation - Search every dictionary entry key in the source text; if found, replace it by the entry value; output the resulting text.
Person Management - Create, retrieve, update and delete person instances; assign roles, address and phones.
Project Management - Create, retrieve, update and delete project instances; include and exclude use cases.
Role Group Management - Create, retrieve, update and delete role group instances; assign master group.
Role Management - Create, retrieve, update and delete role instances; assign role groups.
Assignments
architect@praxis.org - Architect
confadm@praxis.org - Configuration admin
logdesigner@praxis.org - Logical designer, Programmer
qualeng@praxis.org - Quality manager
Person
Assigned Roles
Analyst - Role responsible for analyzing problem domain and requirements.
Architect - Experienced developer, responsible for the technical architecture of a product, including the analytical understanding of the requirem
CCB - Group responsible for evaluating proposed changes to configuration items, and for ensuring implementation of the approved changes.
Change reviewer - Responsible for final review and acceptance of a change.
Configuration admin - Worker responsible for managing configuration resources and configuration management procedures.
Logical designer - Developer responsible for the logical-level internal design of a product.
Programmer - Developer responsible for implementing product code.
Project manager - Manager responsible for planning a project, directing, controlling, structuring and motivating a project team.
Quality manager - Manager responsible for planning, directing, controlling and evaluating quality matters in an organization.
Reviewer - Generic review performer.
Technical writer - Professional responsible for writing, organizing and evaluating user-oriented technical documentation.
Test designer - Developer responsible for the design of the tests of a product, at functional and logical levels.
Test programmer - Developer responsible for writing test scripts.
Test runner - Performer of manual and automated tests execution.
UI designer - Developer responsible for the external design of the user interfaces and of their interactions with the users.

Fig. 18 Executed program – weak details page

V. *FUTURE WORK AND CONCLUSIONS*

During the years where this author lectured practice-oriented courses using Praxis as the process for writing course applications, it was possible to experiment with its use as a process to develop course applications. Currently, as the author has ceased to teach regular courses, a fourth edition of the Brazilian textbook is being written, consolidating in Praxis 4.0 what was learned in its fifteen years of development.

In the last ten years, agile methods became increasingly used, and our professional environment was no exception. UML proficiency has remained a rare asset in the professional market, and Synergia has had to face such reality.

Currently, Synergia demand has switched to a number of smaller projects, together with maintenance of the old projects; the oldest Synergia project is still maintained and updated, after fifteen years of use. Therefore, Synergia has switched to a current process mostly based on the Scrum agile practices [39], together with using Kanban [40] for maintenance projects.

It is intended to extend the Praxis family with an agile variation, which should profit from such experience. In such a version, information contained in the stereotyped properties should be held in spreadsheets, equivalent to those currently extracted from the models by BIRT.

The evolution of the Praxis process and framework was mostly driven by feedback from both course projects and real-life systems. This aligns to a major goal of Synergia: promotion of synergy between academic research, practice-oriented education, and real-life software development, reflected in its very name.

ACKNOWLEDGMENTS

We thank IBM Rational for supporting this work, within the IBM Academic Initiative.

REFERENCES

[1] CMMI Product Team, *CMMI for Development, Version 1.3*, CMU/SEI-2010-TR-033, Software Engineering Institute, Nov. 2010, available at www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm.

[2] W. Pádua, "A Software Process for Time-constrained Course Projects", Proc. 28th International Conference on Software Engineering (ICSE '06), IEEE Press, May 2006, pp. 707-710, Shanghai – China, doi.acm.org/10.1145/1134285.1134397.

[3] W. Pádua, "Using Model-Driven Development in Time-Constrained Course Projects", Proc. 20th Conference on Software Engineering Education and Training (CSEET '07), IEEE Press, pp. 133-140, Dublin, Ireland, Jul. 2007, doi.acm.org/10.1109/CSEET.2007.55.

[4] W. Pádua, "Using Quality Audits to Assess Software Course Projects", 22th Conference on Software Engineering Education and Training, pp. 162-165, Hyderabad, India, Feb. 2009, doi.acm.org/10.1109/CSEET.2009.12.

[5] W. Pádua, "Measuring complexity, effectiveness and efficiency in software course projects", Proc. 28th International Conference on Software Engineering (ICSE '10), IEEE Press, May 2010, vol.1, pp. 545-554, doi.acm.org/10.1145/1806799.1806878 .

[6] B. Pimentel, W. Pádua, C. Pádua, and F. T. Machado, "Synergia: a software engineering laboratory to bridge the gap between university and industry", Proceedings of the 2006 international workshop on Summit on software engineering education (SSEE '06), ACM, New York, NY, USA, pp. 21-24, doi.acm.org/10.1145/1137842.1137850.

[7] V. A. Batista, D. C. C. Peixoto, W. Pádua and C. I. P. S. Pádua, "Using UML Stereotypes to Support the Requirement Engineering: a Case Study", Proceedings of the 2012 International Conference on Computational Science and Its Applications – ICCSA 2012, Salvador, Brazil, 2012.

[8] M. L. Griss, "Software reuse architecture, process, and organization for business success", Proceedings of the Eighth Israeli Conference on Computer Systems and Software Engineering, IEEE Computer Society, Jun 1997, pp. 86 – 89, doi.acm.org/10.1109/ICCSSE.1997.599869.

[9] C. Jones, Assessment and Control of Software Risks, Yourdon Press – Prentice-Hall, 1994.

[10] Watts S. Humphrey, A Discipline for Software Engineering, Addison-Wesley, 1995.

[11] Watts S. Humphrey, Introduction to the Personal Software Process, Addison-Wesley, 1997

[12] Watts S. Humphrey, Introduction to the Team Software Process, Addison-Wesley, 1999.

[13] M. C. Paulk, B. Curtis, M. B.h Chrissis and C. V. Weber, Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-24, Software Engineering Institute, Feb. 1993.

[14] IEEE, IEEE Standards Collection – Software Engineering, IEEE, New York – NY, 1994.

[15] B. Boehm, "Anchoring the Software Process", IEEE Software 13(4), Jul. 1996.

[16] I. Jacobson, J. Rumbaugh and G. Booch, The Unified Software Development Process, Addison-Wesley, 1999.

[17] S. W. Ambler, J. Nalbone and M. J. Vizdos, The Enterprise Unified Process: Extending the Rational Unified Process, Prentice Hall, 2005.

[18] Ph. Kruchten, The Rational Unified Process - An Introduction, 2nd Edition, Addison-Wesley, 2003

[19] S. W. Ambler, The Design of a Robust Persistence Layer for Relational Databases, An AmbySoft White Paper, available at www.ambysoft.com/downloads/persistenceLayer.pdf, Jun. 2005.

[20] P. Tahchiev, F. Leme, V. Massol, and G. Gregory, JUnit in Action, Second Edition, Manning Publications, Jul. 2010.

[21] OMG, Unified Modeling Language: Superstructure, version 2.1.2, Formal/2007-11-02, Nov. 2007, available at www.omg.org/spec/UML/2.1.2/Superstructure/PDF.

[22] IEEE, IEEE Software Engineering Collection on CD-ROM, IEEE, New York – NY, 2003.

[23] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Third Edition, 2004.

[24] OMG, Software & Systems Process Engineering Meta-Model Specification, v2.0, Formal/2008-04-01, Abr. 2008, available at www.omg.org/spec/SPEM/2.0/PDF.

[25] K.Beck, Extreme Programming Explained: Embrace Change, 2nd Edition, Addison-Wesley, 2004.

[26] S. W. Ambler, "Agile Model Driven Development Is Good Enough", IEEE Software 20(5), 70 – 73, Sep. 2003.

[27] P. Haumer, "Increasing Development Knowledge with EPF Composer", Eclipse Review 1(2), Spring 2006, available at www.haumer.net/paper/EPFC-eclipsereview.pdf.

[28] D. Garmus and D. Herron, Function Point Analysis: Measurement Practices for Successful Software Projects, Addison-Wesley, 2000.

[29] AndroMDA, Generate components quickly with AndroMDA, available at www.andromda.org.

[30] S. Jarzabek and H. D. Trung, "Flexible generators for software reuse and evolution", Proceeding of the 33rd International Conference on Software Engineering (ICSE '11), ACM, New York, NY, USA, 920-923, doi.acm.org/10.1145/1985793.1985946.

[31] A. Mashkoor and J. M. Fernandes, "Deriving Software Architectures for CRUD Applications", The FPL Tower Interface Case Study, Proceedings of the International Conference on Software Engineering Advances (ICSEA '07), IEEE Computer Society, Washington, DC, USA, doi.acm.org/10.1109/ICSEA.2007.25.

[32] S. McConnell, Rapid Development: Taming Wild Software Schedules, Microsoft Press, 1996.

[33] J. Heumann, "User experience storyboards: Building better UIs with RUP, UML, and use cases", The Rational Edge, Nov. 2003.

[34] Ch. Bauer, G. King, Java Persistence with Hibernate, Manning Publications, Dec. 2006.

[35] B. Boehm, C. Abts, A. W. Brown and S. Chulani, Software Cost Estimation with Cocomo II, Addison-Wesley, 2000.

[36] Selenium. "Selenium HQ Browser Automation", available at www.seleniumhq.org.

[37] GitHub. "Pragmatists/JUnitParams", available at github.com/Pragmatists/junitparams.

[38] Vaadin, "Thinking of U and I", available at vaadin.com

[39] K. Schwaber and J.Sutherland. "The Scrum Guide", available at hwww.scrumguides.org/scrum-guide.html .

[40] J. Hurtado. "Open Kanban - An Open Source, Ultra Light, Agile & Lean Method", available at agilelion.com/agile-kanban-cafe/open-kanban

Performance and Accuracy conflict in Monitoring Tools for Web Services: a case study

Jael Zela Ruiz
Intitute of Computing
University of Campinas
Campinas, Sao Paulo, Brazil
jael.ruiz@students.ic.unicamp.br

Cecília M. Rubira
Institute of Computing
University of Campinas
Campinas, Sao Paulo, Brazil
Email: cmrubira@ic.unicamp.br

***Abstract*—Web services have become one of the most used technologies in service-oriented systems. Its popularity is due to its property to adapt to any context. As a consequence of the increasing number of Web services on the Internet and its important role in many applications today, Web service quality is a crucial requirement and demanded by service consumers. Terms of quality levels are written between service providers and service consumers to obtain some degree of quality. The use of monitoring tools to control service quality levels is very important. Quality attributes suffer variations in their values during runtime, this is produced by many factors such as memory leak, deadlock, race data, inconsistent data, etc. However, sometimes monitoring tools can impact negatively affecting the quality of service when they are not properly used and configured. This paper aims to show the impact of monitoring tools over service quality, when they are not used properly. The relationship between performance and accuracy is presented and evaluated on web services. Conflict was found between performance and accuracy, where performance was the most affected, because it presented a degradation in its quality level during monitoring.**

Keywords*—*Web Services, SOA, Quality of Service, Quality Attributes, Performance, Accuracy, Monitoring Tools.

I. Introduction

In recent years, Web service has became the most popular and used technology to build SOA applications [1]. Web services are based in a set of protocols and standards as SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description, Discovery, and Integration). Web services are distributed components which are self-contained, discoverable, reusable, composable, and have a transparent location [2]. As a result to its popularity, a increasing number of functionally similar Web services can be found on the internet [3], which entails to the service consumer to ask the question: "what are the better services?" or "which of them fits my needs?" [4]. Service consumers have a difficult task to choose an appropriate service for their requirements. Quality of Service (QoS) has become the most appropriate criterion to distinguish non-functional characteristics between equivalent Web services.

QoS is described as a number of properties, named quality attributes, which take in play the Web service quality. Some of these attributes are: availability, throughput, robustness, and integrity. A set of quality attributes compose a quality model. Several quality models have been proposed both in the research and in the industry [2] [4] [5] [6]. Web services promise quality levels based in quality models. A negociation between the service provider and the service consumer is carried out, in order to assure a specific level of QoS for Web services. A Service Level Agreement (SLA) is the result of this negotiation, where quality is defined, negotiated and tasks to assure quality are established [7]. Nevertheless, afterwards a SLA is arranged for both parties, a new question is asked: How can we be sure that the supposed QoS defined on the SLA is really satisfied?. As a consequence, monitoring tools emerged to control the Web services quality levels. Monitoring tools are based on quality model. They are used to capture, collect, filter, and analyse information from the Web service during runtime [8]. Currently, there are many monitoring tools which come from the research and the industry, such as Dynamo [9], Cremona [10], SALMon [11], WebInject [12], SOAP Monitor [13], Webmetrics Web Services Monitoring [14], etc.

However, because of the dynamic and unpredictable nature of Web services [15], quality attributes can suffer variations in their values during runtime. The relationship among quality attributes can produce conflicts between them when they are monitored at the same time. For example, in a response time and throughput scenario, response time can be a better quality value when it is monitored in isolation, than in parallel with throughput, the reason is because Web service receives a larger number of requests, producing that the Web service takes more time in respond to the user. On the other hand, throughput is also affected, because an smaller number of requests are attended by unit of time, this is due to the required time to respond each request. These conflicts are produced mainly for scalability reasons, Web services can have a lot of service consumers sending many service request at the same time. Monitoring tools become a factor else for quality attributes conflicts.

Monitoring tools can become a double-edged sword, because they are an useful QoS control tool, but they can become the principal reason for conflicts when they are not correctly configured. They can turn out to intrusive agent to the Web service, creating an stressful environment. This is important to know what is being measured, where you are monitoring, how it is being monitored, and how frequently it is monitoring. An active monitoring not correctly configured can overload the Web service and produce a drop in the response time, throughput, or availability to current consumers. The goal of this paper is discover this strife between two important quality attributes, performance and accuracy using FlexMonitorWS

Tool in an active mode over Web services.

This paper is organized as follows. In Section 2, we define monitoring tools and present FlexMonitorWS. In Section 3, we present quality models and define performance and accuracy. We present our conflict scenario to evaluate and results obtained in Section 4. And finally, in Section 5, we provide the conclusions and future works.

II. Monitoring Tools and SOA

In this section we provide a brief description about SOA and monitoring tools. We also present FlexMonitorWS, a monitoring tool for Web services used in this paper.

A. Service-Oriented Architecture

Service-Oriented Architecture (SOA) is an architectural style widely used in distributed applications. Different functional units, services, are connected using standardized and well-defined interfaces [16]. SOA applications are dynamic, heterogeneous, distributed and autonomous.

SOA presents three primary roles: the service provider, the service consumer, and the service broker, as shown in Figure 1. The service provider defines a service in a WSDL file, it is published in the service broker using UDDI, so the service is discoverable to service consumers. Service consumers ask for the service to the services broker, this provides the WSDL file, and the service consumer consumes the service directly. [6].

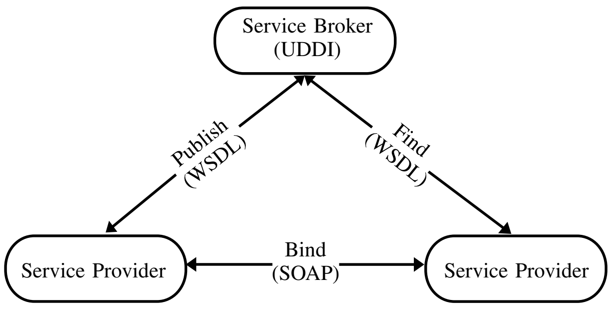


Fig. 1: SOA Architecture

B. Web Service Monitoring Tools

Monitoring tools are systems that capture, collect, filter, and analyze information from a software system in runtime [8]. On Web services, monitoring tools are used to [17]: (1) Improving the process of Web service selection and discovering, this enables QoS-based searches among functionally similar services. (2) Self-healing technique such as dynamic adaptation and dynamic recuperation applied to some quality attributes (availability, scalability, capacity and reliability) when some of then not meet the desired level. (3) To detect violations in SLA, quality metrics based on SLA are used to evaluate and control the Web service.

Monitoring tools can be used according two strategies [4]. *Passive monitoring*, monitor is an sniffer and intercept the exchanged messages between service provider and service consumer, with the aim of obtain the QoS; in this strategy a direct interaction with service provider or consumer is minimized. *Active monitoring*, monitor sent service requests directly to the service provider, acting as a consumer. Monitoring systems can be set up differently according to three components [4], as shown in Figure 2.

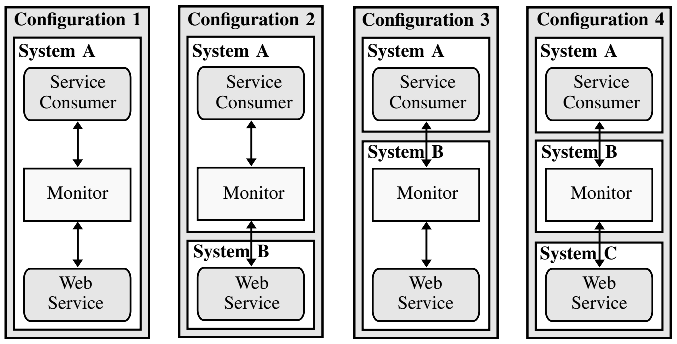


Fig. 2: Monitor configurations.

- Configuration 1: The service consumer, the monitor, and the Web service are in the same system (System A).
- Configuration 2: The service consumer and the monitor are in the same system (System A), and the Web service is in another system (System B).
- Configuration 3: The service consumer is in a system (System A), and the monitor and Web service are together in another system (System B).
- Configuration 4: The service consumer is in the System A, the monitor and the Web service are in the System B and System C, respectively.

C. Monitoring Tools Effects

Depending on how monitoring tools operate over monitored systems, they can produce risk and problems over this last one. Many researchers have reported an intrusion problem of monitoring caused by monitoring tools. This problem is due to two main reasons [18]: (1) monitoring tools consume resource, CPU, memory; (2) potential defects in monitoring tools brings risk to the monitored system, in our case, Web services.

Different methods are used to extract and collect information from Web services. These methods are divided in three types: instrument method, interceptor method, and agent approach [18]. Instrument methods are used by testing techniques. In this method, monitoring code is embedded inside monitored system, Web service, this is inserted manually by the programmers and this can be inserted in any location of the monitored code (e.g. Javassist, AspectJ). Interceptor methods are used in the middleware, they get details about all sent and received messages to the monitored system, Web service. (e.g. Interceptor in CORBA, Handler in AXIS, JVMTI in JVM). This method is more independent but it is executed in the same process with the monitored system. Agent methods are totally independent from the monitored system, Web service, running in its own process [18].

Methods bring intrusive effect to the target system in different degrees. When there are multiple monitors inserted on the target system, this become more complex, instrument mechanisms make the target code difficult to understand and maintain [18]. While interceptor mechanism can lead to performance decrease, because the monitor runs in the same process. On the other hand, the agent approach is the less intrusive method compared with the others, because it runs separately from the target system.

D. FlexMonitorWS Tool

FlexMonitorWS is a Web service monitoring tool based on Software Product Lines (SPL) [19]. This tool is based on the creation of a family of monitors to monitor different points in a Web service application and different quality attributes using different modes of monitoring. It was developed in Java language using FeatureIDE. FlexMonitorWS tool exploits the flexibility property by means of the creation of monitoring profiles which serve to a specific target and user requirements [17].

Monitoring profiles are built according to a feature model (Figure 3) which is divide in (a) monitoring target, (b) quality attributes, (c) operation mode, (d) monitoring frequency, and (e) notification mode [17] [19]. Monitoring target specifies where the monitoring takes place, they can be *Web service*, *server application*, *server*, and/or *network*. Quality attributes indicate what needs to be monitored conforming what is sought, like *availability*, *performance*, *reliability*, *accuracy*, *robustness*, *hardware state*, *failures in log file*, and/or *network QoS*. Operation mode establishes the strategy to be used, a passive monitoring by means of message *interception*, or active monitoring over *invocation* the service directly or by means of *inspection* of log files. Monitoring frequency can be *continuously* or *periodically*. Notification mode setup the method to notify the monitoring generated results, by sending *message* or *log file*. According to the selected features is generated a product, a monitor (jar file), this is executed using a properties file containing the Web service specifications.

Figure 4 shows how FlexMonitorWS works. First the monitoring target is retrieved and its type is recognized, after the monitoring frequency is configured with the help of a *Timer*, the execution is initiated based on the selected operation mode, many samples are captured and recollected continuously or periodically from the Web service, then the information is processed and analyzed according the selected quality attributes. At the end, the results are sent to the interested parties using messages or log file [17] [19].

III. Quality Attributes in SOA

Currently quality models are used in different stages of SOA system lifecycle, such as development, service discovery and selection, and service composition. There are, in the literature, different approaches available to built and select a quality model: *stakeholder approach*, this is classified according the stakeholder concerns (service consumer, service provider and service broker) [2]; *service selection approach* [6] [20]; and *monitoring approach* [4]. Many organizations have also proposed their own quality model [5] such as W3C, OASIS, WS-I, IBM, etc.

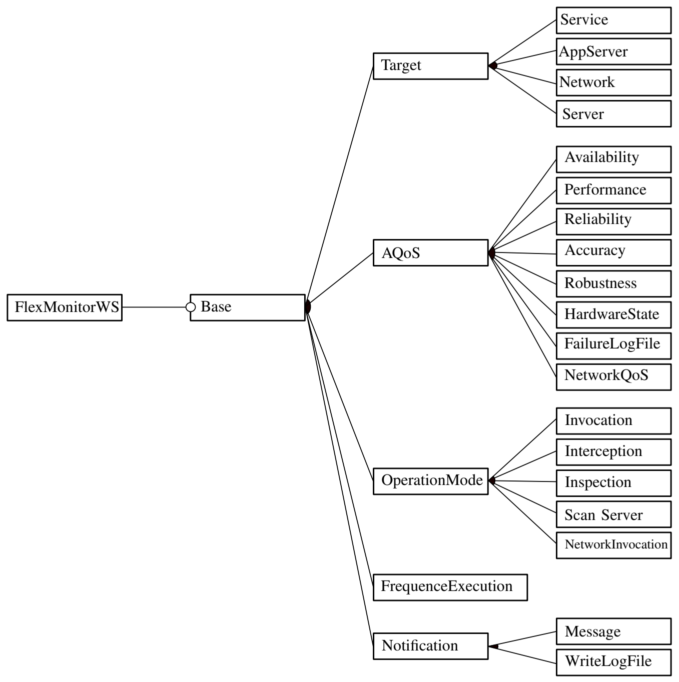


Fig. 3: FlexMonitorWS Feature Model

In this paper, we present two quality attributes, performance and accuracy. While performance is concerned in how quickly a service request can be completed, accuracy is concerned with if the service response is correct. But, correct responses are not always produced quickly. We measure the quality level for each quality attribute to evaluate their relationship and how they are affected for monitoring tools, in particular, using FlexMonitorWS tool.

A. Performance

Performance of a web service represents how fast a service request can be completed [21]. Performance can be measured in terms of throughput, response time, latency, execution time, and transaction time. Response time was selected for this work, because it is the main concern for both service consumers and service providers. It is a critical quality attribute because if a service consumer sees a large delay after send a request to the service, he or she is likely to change to another faster web service [22].

Response Time is the required time to complete a Web service request [5] [21]; the time between sending a request to the Web service and receiving the response. Response time depends primarily on two factors: network delay and server side latency. Response time is measured by the following equation:

$$ResponseTime = T_{response} - T_{request} \qquad (1)$$

Where $T_{request}$ is the time (timestamp) when service request is send to the Web service, and $T_{response}$ is the time (timestamp) when the service response is received from the Web service.

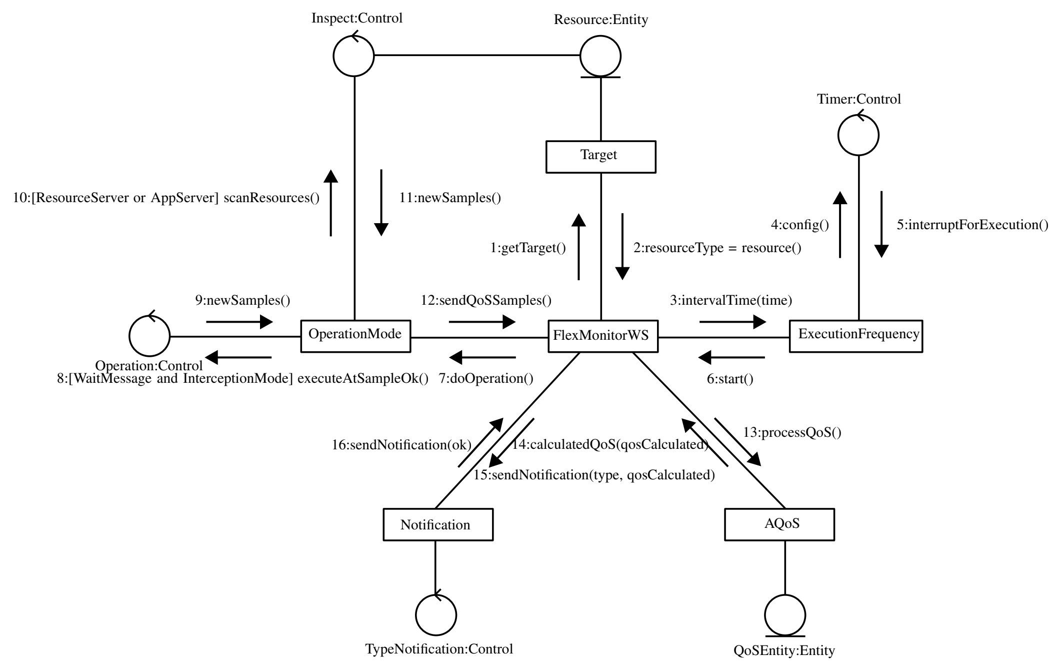


Fig. 4: FlexMonitorWS communication diagram.

B. Accuracy

Accuracy is the level of accurate results that Web service can give to services requests [16]. It is measured by the number of errors (error rate) produced for the Web service over a period of time [17] [21]. Accuracy is concerned about the correctness of the service response, when accuracy value is close to one, it said to be accurate, if it is close to zero, Web service is not accurate; so it loses credibility of its service consumers, when accuracy for a Web service is high.

Accuracy is measured by the following equation:

$$Accuracy = 1 - \frac{nFaults}{totalRequest} \qquad (2)$$

Where $nFaults$ is the number of errors returned for the Web service, and $totalRequest$ is the number of service request sent to the Web service.

IV. Case Study Description

A Web service called *PatientService* to the domain of clinical test delivery, was developed in order to carry out our experiments. *PatientService* presents two operations: *getPatientName()* to get the patient name from a patient, and *getPatients()* to retrieve a list of the recent attended patients.

Given the Web service above, we perform experiments in order to discover possible conflicts between performance and accuracy during monitoring. Our research question have been formulated as follow:

1) What are the performance and accuracy quality levels? each one measured in isolation.
2) Is the accuracy quality level degraded when it is monitored in parallel with performance?
3) Is the performance quality level degraded when it is monitored in parallel with accuracy?

Two products (monitors) were generated using a different set of features. The first monitor called "PerfMonitor" was configured to monitor the performance attribute, where the following features were selected from the feature model in Figure 3:

- Target: *Service* (operation: *getPatients()*)
- Quality attribute: *performance*
- Operation Mode: *invocation*
- Frequency: *30 seconds*
- Notification mode: *WriteLogFile*

Second monitor called "AccMonitor" was configured to monitor the performance attribute, where the following features were selected from the feature model in Figure 3:

- Target: *Web service* (operation: *getPatientName()*)
- Quality attribute: *accuracy*
- Operation Mode: *invocation*
- Frequency: *30 seconds*
- Notification mode: *log file*

The Web service was developed using Java language and it was deployed in Apache Tomcat 7.0, and it was installed in a machine with the following configuration: AMD Phenom(tm) II P920 Quad-Core 1.60 GHz processor, with 6,00 GB main memory, Windows 7 + SP1 as system operating, and JDK 1.8. Monitors were executed in the following environment: Intel(R) Core(TM)2 Duo CPU 2.66 GHz processor, with 4.00 GB main memory, Windows 7 + SP1 system operating, and JDK 1.7.

Monitors were executed in three different cases: (1) "PerfMonitor" monitoring performance quality level over *getPa-*

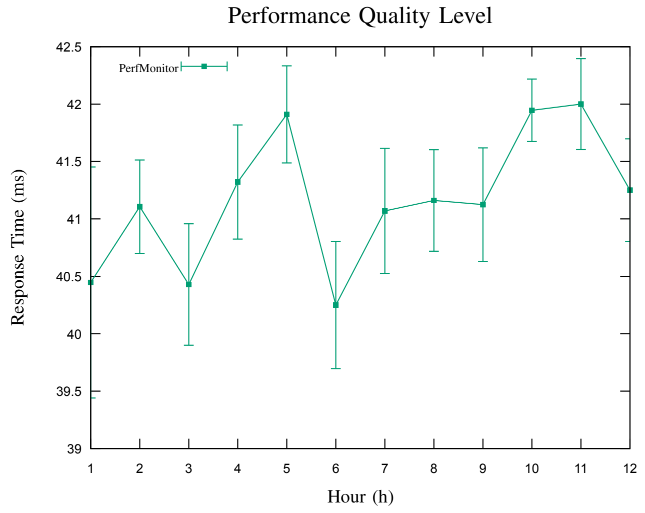


Fig. 5: Performance quality level executing “PerfMonitor”.

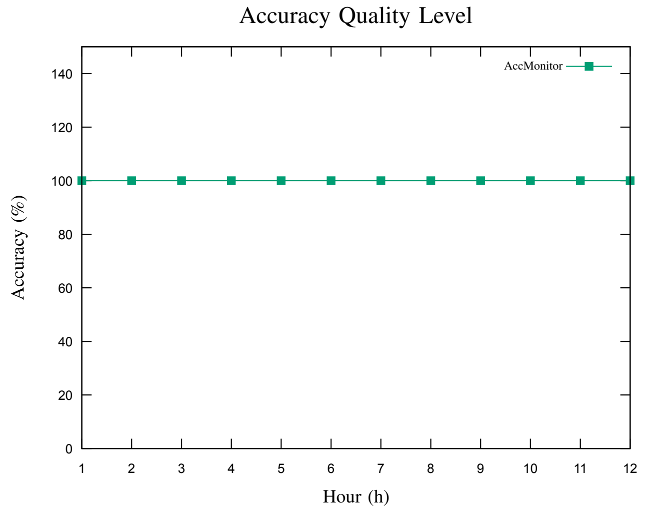


Fig. 6: Accuracy quality level executing just “AccMonitor”.

tients() operation. (2) “AccMonitor” monitoring accuracy quality level over *getPatientName()* operation. (3) “PerfMonitor” monitoring performance and “AccMonitor” monitoring accuracy over *getPatients()* operation and *getPatientName()* respectively at the same time.

A. *PerfMonitor Execution*

“PerfMonitor” was executed over *getPatients()* operation of the *PatientService* Web service, to measure its performance quality level, during 12 hours. Figure 5 shows the average time by monitoring hour, time taken for the Web service to response a request from the service consumer. As an answer to our first research question, the average response time for all was 41.467 milliseconds.

B. *AccMonitor Execution*

“AccMonitor” was executed over *getPatientName()* operation on *PatientService* Web service, to measure its accuracy quality level, during 12 hours. Figure 6 shows the average accuracy percentage calculated by monitoring hour. The accuracy for *PatientService* was 100%, this responds to our first research question.

C. *PerfMonitor and AccMonitor: Parallel Execution*

“PerfMonitor” and “AccMonitor” were executed over *getPatients()* and *getPatientName()* operations respectively on *PatientService* Web service, to measures its performance and accuracy quality levels, during 12 hours. Figure 7 and Figure 8 show the average time to response a request and the average accuracy percentage by hour, when response time and accuracy are monitored in parallel. The accuracy was 100% and the response time for all was 41.208 milliseconds.

Responding to our second research question, it is easy to see that Accuracy quality level remains unchanged during the isolated monitoring and with performance monitoring in parallel, in both cases the accuracy was 100%. It means that the Web service is too strong accurate, and this ensure the correctness about its functionality.

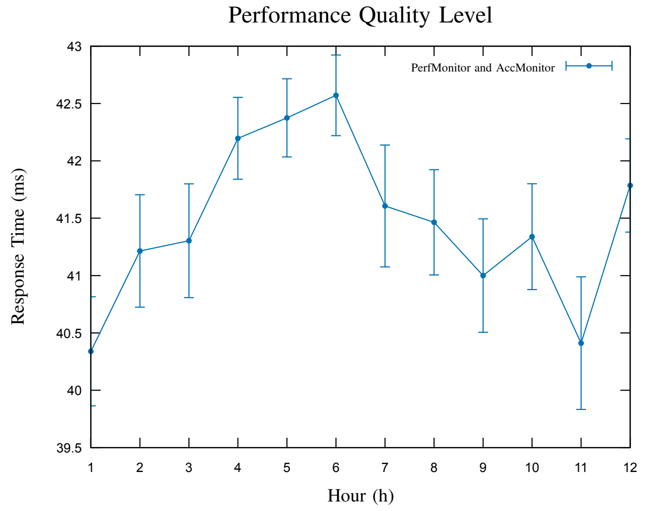


Fig. 7: Performance quality level executing “PerfMonitor” and “AccMonitor” at the same time.

On the other hand, performance quality level is affected when it is executed with accuracy monitor in parallel. Responding our third research question we found a decrease of the quality value in 0.259 milliseconds in the performance. Figure 9 shows the comparison by hour between these two cases. In order to support the difference in the results, we assess the statistical significance between “PerfMonitor” in isolation and “PerfMonitor” with “AccMonitor” results by means of a per-query paired t-test with 95% of confidence. The results of paired t-test confirms that the difference in performance is statistically significant. The Web service presents statistically a better performance when it is monitored just for “PerfMonitor”.

D. *AccMonitor Execution with Fault Injection: PerfMonitor and AccFaultMonitor*

In order to perform some dependability measures and collect evidences our assumption about the strong accuracy of *PatientService*, A new monitor was generated using fault

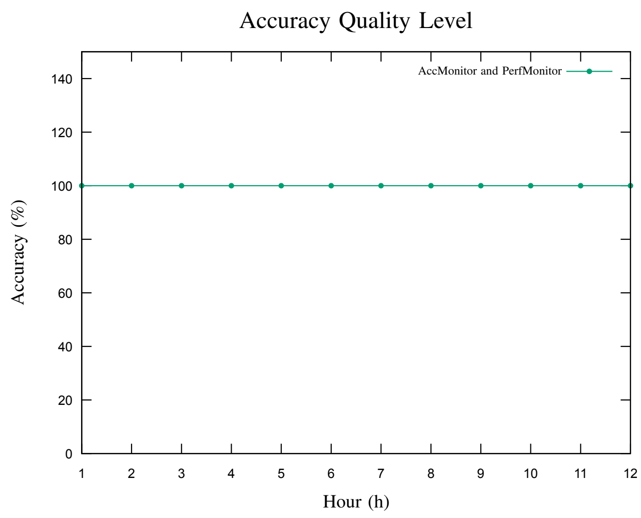


Fig. 8: Accuracy quality level executing “PerfMonitor” and “AccMonitor” at the same time.

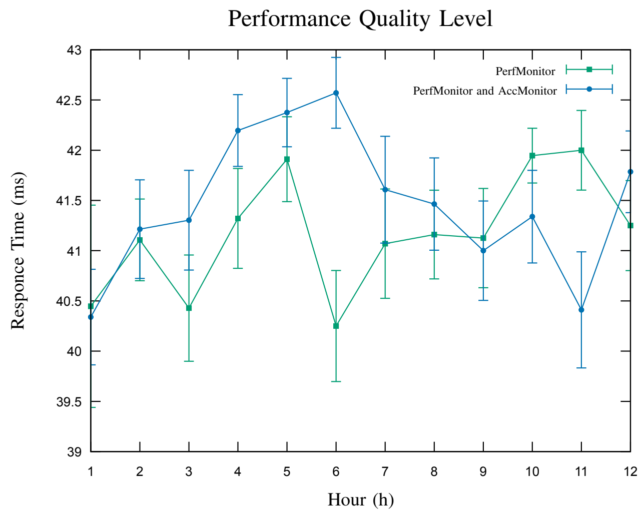


Fig. 9: Performance quality level comparative.

injection, “AccFaultMonitor”. XML injection [23] was used to generate interface faults [24]. We used two kind of injection: Parameters corruption injection and structure corruption injection. For example, we corrupted the patient code sent to *getPatientName()* operation as follows:

Fault 1:

`<arg0>PAT-0239</arg0>` to `<arg0>9320-TAP</arg0>`

we also corrupted the XML structure of the request, inverting opening and closing XML tags as follows:

Fault 2:

`<arg0>PAT-0239</arg0>` to `</arg0>PAT-0239<arg0>`

“AccFaultMonitor” was executed by 12 hours in parallel with “PerfMonitor”. Parameters corruption injection was executed in the first 4 hours, structure corruption injection for the next four hours, and in the last 4 hours, both parameter and structure corruption injections were executed.

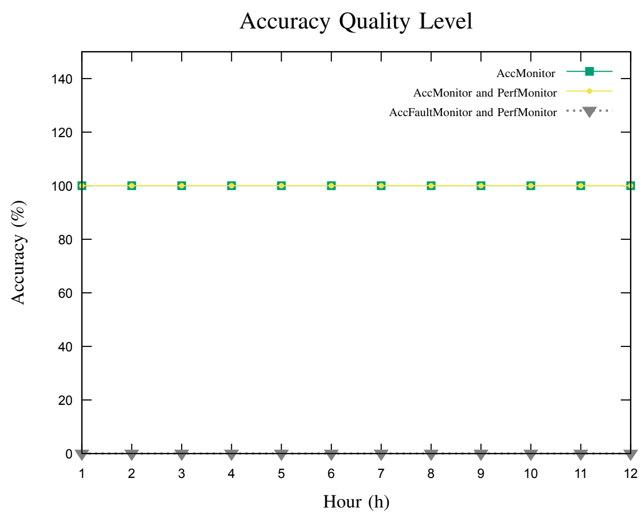


Fig. 10: Accuracy quality level in all scenarios

Figure 10 shows a comparative of the accuracy quality level in all tested scenarios. The calculated percentage for accuracy with fault injection is also displayed by monitoring hour. Corruption of the parameter values produced `java.lang.NullPointerException`. This can be a reasonable behavior because invalid data was sent to the Web service, but on the other hand, it is not a good response, because it is not an adequate response to the service consumer.

Structure corruption faults were all detected by the Web service, and it replied immediately rejecting as malformed SOAP message. Same response was obtained from the last 4 hours, injecting parameters and structure injections.

Figure 11 shows the average time by hour executed in simultaneous with “AccFaultMonitor”. Performance continue suffering a degradation in its quality level as in the previous experiment. Paired t-test between “PerfMonitor” in isolation and “PerfMonitor” with “AccFaultMonitor” in 95% confidence confirms that the Web service present statistically a better performance when performance is monitored in isolation. This is because Web service take more time trying to interpret a corrupted request, holding the processor for more time and also the used memory. This reduces the resources needed for “PerfMonitor”, and therefore taking longer to respond.

V. CONCLUSIONS

Web service monitoring tools are an important issue for the quality of Web service. When we use monitoring tools, we need identify: what is our monitoring target? what do we need monitor?, how monitor it? how often monitor it? and how notify the monitoring results?.

It is necessary to pay attention to the monitor configuration for Web service monitoring, because it can be the main reason for quality level degradation of Web services. Active monitoring (invocation) is a configuration which produce quality level degradation in Web services.

Our study has shown that response time and accuracy have conflict between them, when they are monitored at the same time. Performance is the most affected quality attribute,

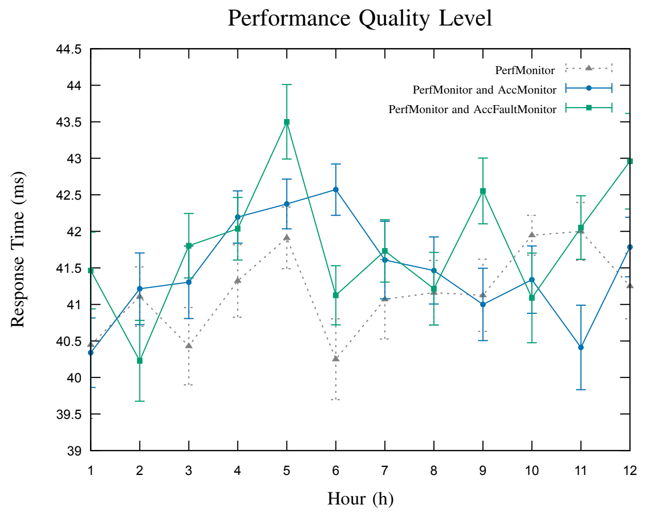


Fig. 11: Performance quality level in all scenarios

because Web service serves a higher number of requests when it is monitored in parallel with accuracy. Statistic tests confirmed a better performance level during monitoring only performance. On the other hand, accuracy was not affected because it remained unchanged in both cases, monitoring in isolation and with performance monitor in parallel.

Injection faults was added to the accuracy monitor to confirm the accuracy of *PatientService*. Parameter and structure corruption injections were not accepted by the Web service, although the exceptional responses were not adequate to the service consumers, it shows that our case study is highly accurate. However, performance needed more time to respond every request, decreasing, even more, its quality level.

Acknowledgment

This work was supported by the Distributed Systems Laboratory, at Institute of Computing, UNICAMP, Campinas, Brazil, and it could not written without support from Laura Cruz and all my friends.

References

[1] Z. Zheng, Y. Zhang, and M. Lyu, “Investigating qos of real-world web services,” *Services Computing, IEEE Transactions on*, vol. 7, no. 1, pp. 32–39, Jan 2014.

[2] Z. Balfagih and M. F. Hassan, “Quality model for web services from multi-stakeholders' perspective,” in *Proceedings of the 2009 International Conference on Information Management and Engineering*, ser. ICIME '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 287–291.

[3] C. R. Choi and H. Y. Jeong, “A broker-based quality evaluation system for service selection according to the qos preferences of users,” *Information Sciences*, vol. 277, no. 0, pp. 553 – 566, 2014.

[4] O. Cabrera and X. Franch, “A quality model for analysing web service monitoring tools,” in *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, May 2012, pp. 1–12.

[5] M. Oriol, J. Marco, and X. Franch, “Quality models for web services: A systematic mapping,” *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1167–1182, Oct. 2014.

[6] M. A. Oskooei and S. M. Daud, “Quality of service (qos) model for web service selection,” in *Computer, Communications, and Control Technology (I4CT), 2014 International Conference on*, Sept 2014, pp. 266–270.

[7] A. Metzger, S. Benbernou, M. Carro, M. Driss, G. Kecskemeti, R. Kazhamiakin, K. Krytikos, A. Mocci, E. Di Nitto, B. Wetzstein, and F. Silvestri, “Analytical quality assurance,” in *Service Research Challenges and Solutions for the Future Internet*, ser. Lecture Notes in Computer Science, M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, Eds. Springer Berlin Heidelberg, 2010, vol. 6500, pp. 209–270.

[8] N. Delgado, A. Gates, and S. Roach, “A taxonomy and catalog of runtime software-fault monitoring tools,” *Software Engineering, IEEE Transactions on*, vol. 30, no. 12, pp. 859–872, Dec 2004.

[9] L. Baresi and S. Guinea, “Towards dynamic monitoring of ws-bpel processes,” in *Proceedings of the Third International Conference on Service-Oriented Computing*, ser. ICSOC'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 269–282.

[10] H. Ludwig, A. Dan, and R. Kearney, “Cremona: An architecture and library for creation and monitoring of ws-agreements,” in *Proceedings of the 2Nd International Conference on Service Oriented Computing*, ser. ICSOC '04. New York, NY, USA: ACM, 2004, pp. 65–74.

[11] D. Ameller and X. Franch, “Service level agreement monitor (salmon),” in *Composition-Based Software Systems, 2008. ICCBSS 2008. Seventh International Conference on*, Feb 2008, pp. 224–227.

[12] W. Goldberg, “Web/http test & monitoring tool,” October 2011. [Online]. Available: http://www.webinject.org

[13] A. S. Fundation, “Web/http test & monitoring tool,” November 2011. [Online]. Available: http://axis.apache.org

[14] N. Webmetrics, “Web/http test & monitoring tool,” November 2011. [Online]. Available: http://www.webmetrics.com

[15] M. I. Ladan, “Web services metrics: A survey and a classification,” in *2011 International Conference on Network and Electronics Engineering IPCSIT, vol.11 (2011) IACSIT Press, Singapore on*, 2011, pp. 93–98.

[16] T. Rajendran and P. Balasubramanie, “Analysis on the study of qos-aware web services discovery,” *CoRR*, vol. abs/0912.3965, 2009. [Online]. Available: http://arxiv.org/abs/0912.3965

[17] R. J. Franco, “FlexMonitorWS: uma solução para monitoração de serviços Web com foco em atributos de QoS,” Master's thesis, Institute of Computing, University of Campinas, Campinas, Sao Paulo, Brazil, 2014.

[18] Q. Wang, Y. Liu, M. Li, and H. Mei, “An online monitoring approach for web services,” in *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, vol. 1, July 2007, pp. 335–342.

[19] R. J. Franco, C. M. Rubira, and A. S. Nascimento, “FlexMonitorWS: uma solução para monitoração de serviços Web com foco em atributos de QoS,” in *Congresso Brasileiro de Software: Teoria e Prática, 21th Sessão de Ferramentas*, vol. 2, 2014, pp. 101–108.

[20] M. A. Oskooei, S. B. M. Daud, and F. F. Chua, “Modeling quality attributes and metrics for web service selection,” *AIP Conference Proceedings*, vol. 1602, pp. 945–952, 2014.

[21] K. Lee, J. Jeon, W. Lee, S.-H. Jeong, and S.-W. Park, “QoS for Web Services: Requirements and Possible Approaches,” in *W3C Working Group Note 25 November 2003*, 2003. [Online]. Available: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

[22] R. Rajamony and M. Elnozahy, “Measuring client-perceived response times on the www,” in *Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, ser. USITS'01. Berkeley, CA, USA: USENIX Association, 2001, pp. 16–16. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251440.1251456

[23] M. I. P. Salas and E. Martins, “Security testing methodology for vulnerabilities detection of xss in web services and ws-security,” *Electron. Notes Theor. Comput. Sci.*, vol. 302, pp. 133–154, Feb. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2014.01.024

[24] F. Bessayah, A. Cavalli, W. Maja, E. Martins, and A. Valenti, “A fault injection tool for testing web services composition,” in *Testing Practice and Research Techniques*, ser. Lecture Notes in Computer Science, L. Bottaci and G. Fraser, Eds. Springer Berlin Heidelberg, 2010, vol. 6303, pp. 137–146.

On the real-state processing of regular operations and The Sakoda-Sipser problem

J. Andres Montoya
Departamento de Matemáticas
Universidad Nacional de Colombia, Bogotá
Email: jamontoyaa@unal.edu.co

David Casas
Departamento de Matemáticas
Universidad Nacional de Colombia, Bogotá
Email: dfcasast@unal.edu.co

***Abstract*—In this work we study some aspects of state-complexity related to the very famous Sakoda-Sipser problem. We study the state-complexity of the regular operations, we survey the known facts and, by the way, we find some new and simpler proofs of some well known results. The analysis of the state of art allowed us to find a new and meaningful notion: Real-state processing. We investigate this notion, looking for a model of deterministic finite automata holding such an interesting property. We establish some preliminary results, which seem to indicate that there does not exists a model of deterministic finite automata having real-state processing of regular expressions, but, on the other hand, we are able of exhibiting a deterministic model of finite automata having real-state processing of star free regular expressions.**

I. Introduction

It is known that nondeterministic finite state automata (1NFAs) are as powerful as deterministic finite state automata (1DFAs), in the sense that 1NFAs can only recognize regular languages. It is also known that 1NFAs are more powerful than 1DFAs, because 1NFAs cannot be simulated by 1DFAs with a polynomial overhead in the number of states. Sakoda and Sipser [1] asked if 1NFAs can be simulated by two-way deterministic finite state automata (2DFAs) with a polynomial overhead in the number of states. It is one of the questions included in the, so called, *Sakoda-Sipser Problem.* The Sakoda-Sipser question is a question about: how, when and to which extent can two-wayness replace nondeterminism? It would be great news if such a question would have an affirmative answer. It is the case, given that 1NFAs (and 2NFAs) are unreliable automata which cannot be used in practice. But, in despite of their *purely theoretical value*, 1NFAs have some remarkable features, which we would like to have in some model of reliable and implementable finite state automata. Thus, we think that the Sakoda-Sipser question is a special case of the following more general question: how, when and to which extent can deterministic finite state automata with added abilities be as powerful and efficient as their nondeterministic counterparts?

Before attacking the later question we will have to consider the following one: which are those remarkable features of nondeterministic finite automata? We wont provide an exhaustive list of remarkable characteristics, but we would like to point out, and to discuss, below, one of those features which has captured our attention.

If one is asked to prove that the set of regular languages is closed under the regular operations, it is very easy to figure out such a proof, if one is allowed to use 1NFAs. Things become harder if one is obligated to employ, in the proof, the weaker model of 1DFAs. Moreover, such an easy proof using 1NFAs yields a linear time algorithm, called Thompson's algorithm, which, on input α (where α is a regular expression), computes an $O(|\alpha|)$-state 1NFA recognizing the language $L(\alpha)$.

We say that a model of deterministic finite automata has *Thompson property*, if and only if, there exists a polynomial time algorithm, which, on input α, computes an $O(|\alpha|^c)$-state automaton within the model, and which recognizes the language $L(\alpha)$, (where c is some fixed constant).

Thus, we have that Thompson property holds for 1NFA, while it is very easy to prove it does not hold for 1DFA. We consider that Thompson property is a remarkable feature of 1NFAs, given that it allows those automata to efficiently process regular expressions. Take into account that the processing of regular expressions is (one of) the main task(s) assigned to finite automata. Unfortunately, the nondeterministic nature of 1NFAs makes them become a nonimplementable solution to the aforementioned problem. Thus, it would be great news if we could exhibit a deterministic model of automata for which Thompson property holds.

In this work we investigate the following question: how, when, and to which extent is it possible to define a model of deterministic finite automata for which Thompson property holds?

Remark 1: We understand The Sakoda-Sipser problem as the question: does there exist a deterministic model of finite state automata which can efficiently simulate nondeterministic automata? It is clear that a positive answer to Sakoda-Sipser implies that our problem can be positively solved. On the other hand, if we could give a positive answer to our question, we could not immediately conclude that Sakoda-Sipser also has a positive answer. It is the case because nondeterministic automata are exponentially more succinct than regular expressions [2].

Remark 2: We assume that the reader knows the definition of the basic models of finite state automata such as DFAs, NFAs, 2DFAs an so on. The interested reader can consult the excellent reference [3].

Organization of the work and contributions. This work is organized into six sections. In section one we introduce the problem that we study in this paper, introducing the notion of real-state processing. In section two we consider the model of DFAs, and we present simpler proofs of some well known results. In section three we study 2DFAs, and we prove that those automata do not have real-state processing of regular operations. Moreover, we prove a strict superpolynomial separation with respect to the model of 1NFAs. In section four we consider a model of two-way pebble automata, and we prove that it is not able of real-state processing concatenations. In section five we introduce a new model of multiplebble automata, and we prove that it has real-state processing of star-free regular expressions. We conclude, in section six, with some concluding remarks.

II. Nondeterministic Finite State Automata and Thompson Property

A model of finite automata is an infinite set $\mathcal{C}$ of state-based recognition devices (automata), that accepts the regular languages, it means that given a regular language there must exist an automaton in $\mathcal{C}$ recognizing the language, and given $\mathcal{M} \in \mathcal{C}$, the language recognized by $\mathcal{M}$ is regular.

We are interested in standard models of finite automata, whose members are constituted by a finite set of internal states, a transition function and, perhaps, some other (finite) resources. For all those models the following definition makes sense.

Definition 3: We say that a model of finite automata, say $\mathcal{C}$, solves the problem of efficiently processing the regular expressions if and only if there exists a polynomial time algorithm $\mathcal{T}$, which, on input α (where α is a regular expression), computes $\mathcal{M}_\alpha \in \mathcal{C}$ such that:

- $L(\mathcal{M}_\alpha) = L(\alpha)$.
- $|Q(\mathcal{M}_\alpha)| \in O(|\alpha|^c)$, where $Q(\mathcal{M}_\alpha)$ denotes the set of internal states of automaton $\mathcal{M}_\alpha$, the symbol $|\alpha|$ denotes the length of α, and c is some positive constant.

Does there exist a model of finite automata that solves the problem of efficiently processing the regular expressions? Yes, there is at least one such model, it is the model of 1NFAs. There exists a linear time algorithm $\mathcal{T}$ which, on input α, outputs $\mathcal{M}_\alpha$, a 1NFA such that $L(\alpha) = L(\mathcal{M}_\alpha)$, and such that $|Q(\mathcal{M}_\alpha)| \in O(|\alpha|)$. Algorithm $\mathcal{T}$ is known as *Thompson's Algorithm* [4], and it is a naive algorithm that exploits the recursive definition of regular expressions plus the following crucial fact: there exists three constants C_U, $C_\cdot$ and C_*, such that given two 1NFAs, say $\mathcal{M}$ and $\mathcal{N}$, one can compute in linear time three 1NFAs $\mathcal{S}^\cup, \mathcal{S}^\cdot$ and $\mathcal{S}^*$ such that:

1) $\mathcal{S}^\cup$ accepts the language $L(\mathcal{M}) \cup L(\mathcal{N})$, and the number of its states is equal to $n + m + C_\cup$.
2) $\mathcal{S}^\cdot$ accepts the language $L(\mathcal{M}) \cdot L(\mathcal{N})$, and the number of its states is equal to $n + m + C_\cdot$.
3) $\mathcal{S}^*$ accepts the language $L(\mathcal{M})^*$, and the number of its states is equal to $n + C_*$.

It is easy to check that the above three facts guarantee that the output of Thompson algorithm, which is the automaton $\mathcal{M}_\alpha$, satisfies the condition $|Q(\mathcal{M}_\alpha)| \in O(|\alpha|)$. It is also easy to check that weaker conditions (as for example $|Q(\mathcal{S}^\cup)| \leq 2n + m + C_\cup$) are not enough to guarantee the existence of a *Thompson Algorithm.* We say that *Thompson Property* holds for $\mathcal{C}$, if and only if, a naive algorithm exploiting the recursive definition of regular expressions can be put to work within the *state constraint* $|Q(\mathcal{M}_\alpha)| \in O(|\alpha|^c)$. For which models of automata does Thompson property holds? We wont characterize those models, but, to begin with, we will exhibit a condition that guarantees that this elusive property actually holds.

Definition 4: We say that $\mathcal{C}$ has real-state processing[1] (real-state conversion) of regular operations if and only if there exists three constants C_U, $C_\cdot$ and C_* such that given two $\mathcal{C}$-automata, say $\mathcal{M}$ and $\mathcal{N}$, with m and n states (respectively), one can compute in polynomial time three $\mathcal{C}$-automata $\mathcal{S}^\cup, \mathcal{S}^\cdot$ and $\mathcal{S}^*$ such that:

1) $\mathcal{S}^\cup$ accepts the language $L(\mathcal{M}) \cup L(\mathcal{N})$, and the number of its states is equal to $m + n + C_\cup$.
2) $\mathcal{S}^\cdot$ accepts the language $L(\mathcal{M}) \cdot L(\mathcal{N})$, and the number of its states is equal to $m + n + C_\cdot$.
3) $\mathcal{S}^*$ accepts the language $L(\mathcal{M})^*$, and the number of its states is equal to $m + C_*$.

We have

Proposition 5: If $\mathcal{C}$ has real-state processing of the regular operations, then Thompson property holds for $\mathcal{C}$

From now on, we will be studying the following problem

Problem 6: (**The Real-State Processing Problem**) Does there exists a deterministic model of finite automata which has real-state processing of the regular operations?

III. Deterministic One-way Finite Automata

The first model of deterministic automata that we will consider is the standard model of 1DFAs. We count, in this model, with a powerful tool for lowerbounding the state complexity of a given regular language, it is Myhill-Nerode theorem.

Definition 7: Given an alphabet Σ, and given a language $L \subset \Sigma^*$, we define an equivalence relation $R_L \subset \Sigma^* \times \Sigma^*$ in the following way: given $x, y \in \Sigma^*$, we have that xR_Ly if and only if for all $w \in \Sigma^*$ it happens that $xw \in L \iff yw \in L$.

Theorem 8: (**Myhill-Nerode**)

$L \subset \Sigma^*$ is a regular language if and only if the quotient $\frac{\Sigma^*}{R_L}$ is finite. Moreover, if the language L is regular, then a minimal 1DFA recognizing L has $|\frac{\Sigma^*}{R_L}|$ states.

For a proof see [3].

We can use Myhill-Nerode theorem to lowerbound the state-complexity of unions in this model. Next result is part of the folklore of automata theory, but the proof of the lower bound is neither trivial nor easy to find in the standard references, we include it for the sake of completeness.

Lemma 9: (upper and lower bounds for unions)

1) Let $\mathcal{N}$ be a 1DFA with n states, and let $\mathcal{M}$ be a 1DFA with m states (over the same input alphabet), the

[1]In analogy with the notion of real-time

language $L(\mathcal{N}) \cup L(\mathcal{M})$ can be recognized by an 1DFA with at most nm states.

2) There exists a sequence of regular languages, say $(P_n)_{n\geq 1}$, such that for all $n \geq 1$, the language P_n can be accepted by an 1DFA with n states, but such that for infinitely many pairs (n, m), the language $P_n \cup P_m$ requires nm states.

Proof. Given $\mathcal{N}$ and $\mathcal{M}$, like in the statement of item 1, we define a third automaton $\mathcal{U}$ as follows:

- $Q_\mathcal{U} = Q_\mathcal{N} \times Q_\mathcal{M}$.
- $q_{0_\mathcal{U}} = (q_{0_\mathcal{N}}, q_{0_\mathcal{M}})$.
- $\delta_\mathcal{U} : (Q_\mathcal{N} \times Q_\mathcal{M}) \times \Sigma \to Q_\mathcal{N} \times Q_\mathcal{M}$ is defined in the following way.

$$\delta_\mathcal{U}((q_\mathcal{N}, q_\mathcal{M}), x) = (\delta_\mathcal{N}(q_\mathcal{N}, x), \delta_\mathcal{M}(q_\mathcal{M}, x)).$$

- $A_\mathcal{U} = (A_\mathcal{N} \times Q_\mathcal{M}) \cup (Q_\mathcal{N} \times A_\mathcal{M})$.

Automaton $\mathcal{U}$ has nm states, and it is easy to check that it recognizes the language $L(\mathcal{N}) \cup L(\mathcal{M})$.

Given $n \geq 1$, we set

$$P_n = \{x \in \Sigma^* : |x| \equiv 0 \, (mod\,(n))\}$$

We have that $x \equiv_{P_n} y$ if and only if $|x| \equiv |y| \; (mod\,(n))$, and hence we know that there exists a 1DFA with n states recognizing the language P_n. Now, we consider the sequence $\{P_n^c\}_{n\geq 1}$. Given $n \geq 1$, there exists a 1DFA with n states recognizing the language P_n^c. We can use Chinese remaindering to prove that $P_n^c \cup P_m^c$ requires $\operatorname{lcm}(n, m)$ states. Now, if we suppose that n and m are *coprime* (i.e. $\gcd(n, m) = 1$) we get the lower bound nm. ■

Corollary 10: 1DFAs do not have real-state processing of unions.

We know that real-state processing implies Thompson property, but given that the converse is not true it is still possible that Thompson property holds for 1DFAs. It is easy to prove that it is not the case.

Proposition 11: Thompson property does not hold for 1DFAs.

Proof. Suppose that Thompson property holds for 1DFAs, then given a regular expression α there must exist a 1DFA with $O(|\alpha|)$ states and which recognizes the language $L(\alpha)$. We know that it is not possible because regular expressions are *exponentially more succinct* than 1DFAs. ■

We used in the above proof that regular expressions are exponentially more succinct than 1DFAs, it means that there exists a sequence of regular languages $\{L_k\}_{k\geq 1}$, and there exists a constant $C > 1$ such that:

- For all k, there exists a regular expression α_k which denotes the language L_k and whose length is linear in k.
- Given $k \geq 1$, a minimal 1DFA accepting L_k requires $\Omega(C^k)$ states.

There are many examples of sequences that behave this way, we include a classical example, which will be used again in the next sections. Let $\Sigma = \{0, 1\}$, let $k \geq 1$ and let L_k be the language defined by

$$L_k = \{x \in \Sigma^* : x[|x| - k + 1] = 1\}$$

That is: language L_k is the set of all strings such that the position that is placed at k positions from the right end is filled with a 1.

Let $k \geq 1$, it is easy to check that a regular expression denoting the language L_k is the expression $(0\cup 1)^* 1 (0 \cup 1)^{k-1}$, notice that the length of this expression is equal to $5 + (k-1)3$. Now consider the equivalence relation $\equiv_k$ determined by the language L_k. We have that x, y are in the same equivalence class if and only if the last k characters of both strings are all equal. Then, we can claim that there are at least 2^k equivalence classes, and it implies that a minimal 1DFA recognizing L_k has at least 2^k states.

Thus, we know that the model of 1DFAs is not the right model, Thompson property does not hold for it, given that it is not able of real-state processing unions (which seems to be the more tractable of the regular operations) and, as we will see, it behaves even worse when it comes to the processing of concatenations.

Next result is a well known result [5], but we have found a new proof which seems to be very much simpler.

Theorem 12: The 1DFA-state complexity of concatenation is at least exponential.

Proof. Let $\Sigma = \{0, 1\}$, and let $\{A_n\}_{n\geq 1}$ be the sequence of languages defined by:

$$A_n = \{x \in \Sigma^* : |x| \geq 0\}$$

Notice that $\{A_n\}_{n\geq 1}$ is a constant sequence (all the languages are the same). Finally, we introduce a second sequence $\{B_n\}_{n\geq 1}$, where given $k \geq 1$ the language B_k is equal to

$$\{x \in \Sigma^* : x[1] = 1 \; \& \; |x| = k\}$$

We can check that for each n, and for each k the equality $A_n \cdot B_k = L_k$ holds. It is also easy to check that:

1) For all n, language A_n can be recognized using an automaton with an unique internal state.
2) For all n, language B_n can be recognized using an automaton with $n + 2$ states.
3) For all n, language L_n requires 2^n states.

Altogether, we get an exponential lower bound for the processing of concatenations employing 1DFAs. ■

IV. Deterministic Two-Way Finite State Automata

In this section we study the model of *two-way terminating deterministic finite state automata* (2DFAs, for short).

Let $\mathcal{M} = (Q, \Sigma, q_0, F, \delta)$ be a 1DFA, and let $x = x_1 x_2 \ldots x_n$ be a string of size n. The computation of $\mathcal{M}$, on input x, takes n time units which is the time required by the workhead to reach the right end of the input. Now suppose that $\mathcal{M}$ is a two-way deterministic finite automaton, the computation of $\mathcal{M}$, on input x, could be infinite. We will avoid this possibility restricting ourselves to studying two-way terminating finite state automata, which are the two-way deterministic finite automata that halt on all their inputs. There is not loss of generality if we restrict the investigation to the later type of two-way automata, it is the case given that:

1) There is no real loss of computation power: the restricted model of two-way terminating automata can recognize all the regular languages (any 1DFA is a 2DFA which never moves leftward).
2) There is not a significant blow-up in state-complexity: any two-way deterministic automaton with n states can be simulated by a 2DFA with $4n+1$ states [6].

It is known that 2DFAs are exponentially more powerful than 1DFAs. To check this, it is enough to consider the sequence $\{L_n\}_{n\geq 1}$ introduced in the proof of theorem 12, notice that L_n can be recognized employing a 2DFA that uses $n+3$ states, while any 1DFA recognizing L_n requires 2^n states. Interesting enough, it can be proved that 2DFAs could be exponentially more powerful than 1NFAs. This fact was known by Sakoda and Sipser [1], who proved the result using a sequence of languages which has been instrumental in the study of The Sakoda-Sipser problem (the sequence defining *The Liveness Problem* [1]). We will include a proof of this fact, which is based on a very much simpler sequence of languages.

Theorem 13: There exists a sequence of regular languages, say $\{M_n\}_{n\geq 1}$, such that.

1) Given $n \geq 1$, there exists a 2DFA recognizing M_n which uses at most $4n+3$ states.
2) Given $n \geq 1$, it happens that any 1NFA recognizing M_n requires at least 2^n states.

Proof. Given $n \geq 1$, we set $\Sigma_n = \{1, \ldots, n\}$. Given $P = \{i_1, ..., i_k\} \subseteq [n]$, we define $L_P^n = \{i_1, ..., i_k\}^*$. Finally, given $n \geq 1$, we define M_n as the language

$$\left\{w \in \Sigma_n^* : \exists P \left(w \in L_P^n 0 L_{[n]-P}^n\right)\right\}$$

It is not hard to figure out a terminating 2DFA with $4n+3$ states recognizing the language M_n. Now, we will prove that any 1NFA recognizing M_n requires 2^n states. Each $P \subseteq \Sigma_n$ can be represented by a string w_P which corresponds to write down the elements of P in increasing order. Thus, we have 2^n different pairs of strings, the pairs in the set $\left\{(w_P 0, w_{[n]-P}) : P \subseteq [n]\right\}$, satisfying the following two conditions:

1) $w_P 0 w_{[n]-P} \in M_n$.
2) If $P \neq Q$, the string $w_P 0 w_{[n]-Q}$ does not belong to M_n.

The existence of such a set of pairs, of size 2^n, implies that any 1NFA recognizing the language M_n has at least 2^n states [7]. ■

Problem 14: Our proof, as well as Sakoda-Sipser's proof, employs increasing alphabets. It is natural to ask if the same exponential separation can be achieved over a fixed alphabet. Does a similar separation hold in the unary case?

Thus, the model of 2DFAs seems to be powerful enough as to be able of efficiently simulating nondeterministic finite automata. Do 2DFAs have real-state processing of regular operations?

Theorem 15: Let $\mathcal{M}_1$ be a 2DFA with m states and let $\mathcal{M}_2$ be a 2DFA with n states, there exists a 2DFA that recognizes the language $L(\mathcal{M}_1) \cup L(\mathcal{M}_2)$, using no more than $m+n+1$ states.

Proof. Suppose that Q_i is the set of states of $\mathcal{M}_i$, $i = 1, 2$. We claim that one can construct a 2DFA $\mathcal{N}$ which recognizes the language $L(\mathcal{M}_1) \cup L(\mathcal{M}_2)$, and such that the set of its states is equal to $Q_1 \cup Q_2 \cup \{q\}$ (we can suppose, without loss of generality, that Q_1 and Q_2 are disjoint sets and that $q \notin Q_1 \cup Q_2$). The computation of this automaton begins in the initial state of $\mathcal{M}_1$, and it proceeds by simulating $\mathcal{M}_1$ until a final state is reached (either an accepting or a rejecting state), if the final state reached is an accepting state, automaton $\mathcal{N}$ halts and accepts the input, otherwise it enters the special state q, and begins to look for the left end of the input. Once the left end is reached, automaton $\mathcal{N}$ begins to simulate the computation of $\mathcal{M}_2$. Along this second phase a final state must be eventually reached, if this final state is accepting the automaton accepts, otherwise it rejects the input. ■

Now, we consider the concatenation operation. Next result is taken from [5]:

Theorem 16: Let $m, n \geq 1$, the language $T_m = L\left(a^{m-1}(a^m)^*\right)$ can be recognized by an m-state 1DFA, but if n and m are coprime the concatenation language $T_n \cdot T_m$ requires mn states.

The above lower bound indicates that 2DFA are unable of real-state processing concatenations, even in the unary case. Thus, it seems that real-state processing is something that is very hard to achieve. We will relax our problem a little bit, considering the following weaker question:

Problem 17: (**The real-state processing problem for star free expressions**)

Does there exist a model of deterministic finite automata which has real-state processing of star free expressions?

We will investigate this new problem in the remaining of the paper, but before of this we will discuss the existence of a superpolynomial separation (related to problem 14) that holds in the unary case.

Definition 18: **Landau's function**

Landau's function is the function $g : \mathbb{N} \to \mathbb{N}$ defined by

$$g(n) = max\{lcm(p_1, \ldots, p_k) : k \geq 1 \ \& \ p_1 + \cdots + p_k \leq n\}$$

Landau introduced this function in the study of some number theoretical problems, he proved the asymptotic lower bound $g(n) \geq e^{(1+o(1))\sqrt{nln(n)}}$. Next result is taken from [5].

Theorem 19: The language $R_n = L\left(a^{g(n)-1}(a^{g(n)})^*\right)$ can be recognized employing a n-state 2DFA, but every 2DFA accepting R_n^* has at least $(g(n)-1)^2$ states.

The first corollary that we could get from the above theorem is that 2DFAs cannot real-state process the Kleene star. We can get a second interesting corollary, which gives a definitive answer to problem 14.

Corollary 20: Unary 2DFAs are superpolynomially more succinct than unary 1NFAs.

Proof. Let $h_1(n)$ be the number of states of a minimal 1NFA recognizing the language R_n, and let $h_2(n)$ be the number of states of a minimal 1NFA recognizing the language R_n^*. Notice that $h_1(n) = h_2(n)$. We define $g_1(n)$ as the number of states of a minimal 2DFA recognizing the language R_n, and we define $g_2(n)$ as the number of states of a minimal 2DFA recognizing the language R_n^*. Notice that

$$g_1(n) \leq n < e^{(1+o(1))\sqrt{nln(n)}} \leq g_2(n)$$

Now, we use that any n-state unary 1NFA can be simulated by a 2DFA with a (at most) quadratic overhead in the number of states [8]. Thus, we have that

$$h_1(n) \geq e^{\frac{(1+o(1))\sqrt{g_1(n)\ln(g_1(n))}}{2}}$$

It is clear that a function $e^{\frac{(1+o(1))\sqrt{m\ln(m)}}{2}}$ is superpolynomial in m, and then the corollary is proved. ■

Remark 21: Corollary 20 shows that, in the unary world, 2DFAs behaves better than 1NFAs when it comes to the processing of regular operations. It should not be considered a big surprise: recall that 2DFAs behaves better than 1NFAs when it comes to the processing of the intersection and complementation operations (see [3]), which are nonregular operations that are analogous to the three basic regular operations.

V. Deterministic Two-Way Pebble Automata

Star free regular expressions are the regular expressions that can be constructed using only unions and concatenations, these expressions constitute an important class of expressions: these are the regular expressions that denote the finite languages. One could think that finite languages are boring, but he has take into account that these are the formal languages that are most used in the theory of programming languages (the area where the efficient processing of regular expressions becomes an important task). Does there exist a deterministic model of finite automata which has real-state processing of the star free regular expressions?

The models of automata studied so far have several different features but a common feature: they cannot write on their tapes. If we add those automata the ability of writing on their tapes, we could leave the regular world. There exists some very weak forms of writing, which does not force 2DFAs to leave the regular world, one important example is the writing ability provided by a single pebble. Ibarra et al [9] studied a model of pebble automata that accepts the regular languages, Ibarra automata are two-way deterministic automata provided with a single pebble which is used by those automata to mark cells on their tapes (for definitions see [9]). We use the symbol 1p2DFA to denote the class of Ibarra automata that adhere to the following further restriction:

Let $\mathcal{M}$ be a 1p2DFA. It has two initial states, the *authentic initial* state which is called the L-initial state, and a second special state which is called the R-initial state. Suppose that cell i is the current position of the pebble, and suppose that the workhead is located on cell j, with $j < i$. Then, the workhead is forced to stay within the first $i-1$ cells until the automaton reaches a *transition state*. Transition states are divided into accepting and rejecting states. Once a transition-accepting state is reached, the automaton looks for the pebble, locates it without picking it up, changes its state to the R-initial state and begins to work on the right side of the tape. On the other hand, if a transition-rejecting state is reached, the automaton looks for the pebble, picks it up, advances one step to the right, places the pebble on the next to the right cell, looks for the leftend and changes its initial state to the L-initial state. The pebble can also be picked up from the right, but it can happen only if the automaton has reached a transition-rejecting state while being on this side of the tape. Once the automaton reaches a transition-accepting state, while being working on the right, it halts and accepts the input. On the other hand, if the transition state is rejecting, the automaton looks for the pebble, locates it and picks it up, moves one step to the right, places the pebble on this cell (the next to the right cell), looks for the left end of the tape and changes its internal state to the L-initial state. Thus, our 1pDFAs use the pebble only to cut the tape into two disjoint segments. When the pebble is placed on the tape, those automata work first on the left segment, and then on the right segment. Moreover, their computations are divided into completely independent stages, the transition between two successive stages being given by moving the pebble one step to the right. Notice that those automata are tailor-made to process the concatenation of two regular languages.

Our restricted model of pebble automata accepts the regular languages: this model cannot accept nonregular languages because it is weaker than the Model of Ibarra et al, and, on the other hand, the model can accept all the regular languages because a 2DFA is a 1p2DFA that never uses its pebble.

It is very easy to check that 1p2DFAs are able of real-state processing unions. It could happens that the pebble (this new ability) allows those automata to real-state process concatenations, but first we have to ask: do those very restricted pebbles yield some computation power?

Definition 22: Given $\mathcal{C}$ and $\mathcal{D}$, two different models of automata, we say that $\mathcal{C}$ cannot be linearly-simulated by $\mathcal{D}$ if and only if there exist a sequence of regular languages, say $\{U_n\}_{n\geq 1}$, a function $f:\mathbb{N}\to\mathbb{N}$ and a positive real number ε such that for all n there exists a $\mathcal{C}$-automaton with $f(n)$ states recognizing the language U_n, while any $\mathcal{D}$-automaton recognizing the same language requires $\Omega\left(f(n)^{1+\varepsilon}\right)$ states.

We prove that 1p2DFAs cannot be linearly-simulated by 2DFAs, even in the unary case

Theorem 23: 1p2DFAs cannot be linearly-simulated by 2DFAs, even in the unary case.

Proof. Suppose that we have two regular languages, say L and T, and suppose that we have a n-state 2DFA $\mathcal{N}$ recognizing the language L, and a m-state 2DFA $\mathcal{M}$ recognizing the language T. It is not hard to figure out a 1p2DFA recognizing $L\cdot T$ and employing $O(n+m)$ states. To achieve the upper bound one can use the pebble in the following way: suppose that the input is w, and suppose that the automaton has placed the pebble on cell i, then it checks if $w[1...i-1]$ belongs to

L, and then if it is the case it checks if $w\left[i...\left|w\right|\right]$ belongs to T. If w pass both tests the automaton halts and accepts the input; otherwise it looks for the pebble, picks it up, places it on the next to the right cell and begins once again.

Thus, 1p2DFAs can real-state process the concatenation of two 2DFAs. Now, given $n \geq 1$, we set $T_n = L\left(a^{n-1}(a^n)^*\right)$. We know that T_n can be recognized employing a 2DFA with n states. Hence, given $n, m \geq 1$ the language $T_n \cdot T_m$ can be recognized by a 1p2DFA with $O\left(n+m\right)$ states. On the other hand, we know that if n and m are coprimes ($\gcd\left(n,m\right) = 1$) any 2DFA recognizing the language $\left(T_n \cdot T_m\right)$ requires $\Omega\left(nm\right)$ states. ■

Remark 24: It is important to remark that a stronger separation result is already known. Let $\left\{p_i\right\}_{i\geq 1}$ be the enumeration of the prime numbers in ascending order. Given $m \geq 1$, we set $P_m = \prod\limits_{i \leq m} p_i$, and we set $L_m = \left\{1^l : l < P_m\right\}$. Geffert and Istonova [10] proved that the sequence $\left\{L_m\right\}_{m\geq 1}$ requires $\Omega\left(2^{m\log(m)}\right)$ states over the model of 2DFAs, but that it can be recognized using at most $O\left(m^2\log\left(m\right)\right)$ states over a model of 1p2DFAs studied by them. We have included the above proof because of three reasons: because our proof is a very simple proof which has been obtained thanks to our analysis of real-state conversion, (which seems to be a meaningful notion that will allow us to discover new proofs and new results), because their model of 1p2DFAs is stronger than our model (and then their proof could not hold in our case), and finally because we will get an interesting corollary (corollary 25) from the above proof:

Corollary 25: 1p2DFAs can real-state process the concatenation of two 2DFAs.

The above corollary is not sufficient for our purposes, we have to ask: are 1p2DFAs able to real-state process concatenations of 1p2DFAs? Notice that if we try to use the naive idea used in the above proof, we will promptly realize that we need three pebbles. Are we allowed to use more than one pebble? It is not hard to figure out a two-pebble automaton recognizing the language of palindromes, and it is well known that this language is not regular. Thus, if we want to consider some type of automata using more than one pebble, we will have to impose some further constraints on the way those automata can handle their pebbles. We follow this direction of research in the next and last section, but before of this we would like to conclude with our analysis of concatenations over the model of 1p2DFAs.

Given n, we use the symbol H_n to denote the language $L\left((a^n)^*\right)$.

Lemma 26: Let n, m be two integers, which are coprime, the number of states that are necessary to recognize the language $H_n \cdot H_m$ using 2DFAs is at least $nm - n + m - 1$.

Proof. Given an unary regular language L, and given $\mathcal{M}$, a minimal DFA accepting L, if the tail of $\mathcal{M}$ is equal to l, then a minimal 2DFA recognizing L has at least $l-1$ states. Notice that the language $H_n \cdot H_m$ is cofinite, and notice that the largest string that is not contained in this language is the string $a^{mn-(m+n)}$. Then, we have that the tail of a minimal DFA recognizing $H_n \cdot H_m$ is equal to $mn - (m+n)$, and then we have that any 2DFA recognizing this language has at least $nm - n + m - 1$ states. ■

Theorem 27: Let n, m, s be three different prime numbers, we have that any 1p2DFA recognizing the language $H_n \cdot H_m \cdot H_s$ requires $\Omega\left(\min\left\{nm, ms, ns\right\}\right)$ states.

Proof. Let $\mathcal{M}$ be a minimal 1p2DFA accepting the language $H_n \cdot H_m \cdot H_s$. Given $w \in \left\{a\right\}^*$, string w determines a unique computation of the automaton $\mathcal{M}$, which is divided in at most $|w|$ stages. If $w \in H_n \cdot H_m \cdot H_s$, it becomes accepted only because of the last stage of the computation, which begins when automaton places its pebble on a given cell, say i, dividing in this way the input string into two strings $w\left[1...i-1\right]$ and $w\left[i...\left|w\right|\right]$.

We define

$$L = \left\{w\left[1...i-1\right] : w \in \left\{a\right\}^* \ \& \ \alpha_{\mathcal{M}}\left(w,i\right)\right\}$$

and

$$T = \left\{w\left[i...\left|w\right|\right] : w \in \left\{a\right\}^* \ \& \ \beta_{\ \mathcal{M}}\left(w,i\right)\right\}$$

where $\alpha_{\mathcal{M}}\left(w,i\right)$ is supposed to mean:

If the computation of automaton $\mathcal{M}$, on input w, begins with the pebble placed on cell i, the workhead located on the left-end of the input and the internal state of $\mathcal{M}$ being equal to the L-initial state, then $\mathcal{M}$ will reach a transition-accepting state before picking up the pebble.

And $\beta_{\mathcal{M}}\left(w,i\right)$ is supposed to mean:

If the computation of automaton $\mathcal{M}$, on input w, begins with the pebble placed on cell i, the workhead located on cell i, and the internal state of $\mathcal{M}$ being equal to the R-initial state, then $\mathcal{M}$ reaches a transition-accepting state.

We have that $\mathcal{M}$ accepts w at the ith stage if and only if $w\left[1...i-1\right] \in L$, and $w\left[i...\left|w\right|\right] \in T$. Therefore, we have that $\mathcal{M}$ accepts w if and only if $w \in L \cdot T$, and it is equivalent to claim that $H_n \cdot H_m \cdot H_s = L \cdot T$. Notice that L and T are regular languages. One can use the primality of n, m and s to prove that there exists a regular language Ω, and that there exist $x, y \in \left\{n, m, s\right\}$ (with $x \neq y$) such that either $L = H_x \cdot H_y \cdot \Omega$ or $T = H_x \cdot H_y \cdot \Omega$. We can suppose, without loss of generality, that $L = H_x \cdot H_y \cdot \Omega$. Notice that one can use $\mathcal{M}$, without the pebble, to recognize the language L, it implies that the number of states of $\mathcal{M}$ is bigger than the minimum number of states that are necessary to recognize L using 2DFAs. It is easy to check that the state complexity of unary 2DFAs can only increase with concatenations, and then the number of states of automaton $\mathcal{M}$ is bigger than the number of states that are required to recognize the language $H_x \cdot H_y$ using 2DFAs. Thus, we have that $\mathcal{M}$ has at least $\Omega\left(xy\right)$ states. Therefore, we get the lower bound. ■

Corollary 28: 1p2DFAs do not have real-state processing of concatenations.

VI. Deterministic Two–Way Multipebble Automata

So far, we have surveyed the most popular models of finite state automata, we showed that all those models, but the model of 1NFA, are unable of real-state processing regular expressions. It seems that real-state processing is not achievable within the world of deterministic finite state automata. Therefore we decided to relax our goal: we would be happy if we could find a model that is able of efficiently processing all the star free regular expressions (a model that is able of real-state processing unions and concatenations).

In this section we study the ultimate model of deterministic finite state automata, which is the model of *directed multipebble deterministic two-way finite automata* (dp2DFA, for short) introduced below. We prove that it is more powerful (succinct) than the model 1p2DFA, and we prove that this new model holds real-state processing of concatenations and unions (real-state processing of star free regular expressions).

A directed multipebble deterministic two-way finite automata is a pebble automaton provided with a fixed number of pebbles which can be larger than 1. We mentioned before that one can construct a two pebble automaton recognizing palindromes, hence we have to impose some strong restrictions on the way those automata can handle their pebbles.

Suppose that we have a dp2DFA with k pebbles, say $P_1, ..., P_k$, which is processing the input string w, and suppose that it has placed its first i pebbles on cells $j_1, ..., j_i$. To begin, we demand that $j_1 \leq j_2 \leq ... \leq j_i$ and that the pebbles were placed one by one respecting the order of their labels (which are $1, ..., k-1$ and k). We suppose that the workhead is placed on the interval containing the string $w[j_{i-1}...j_i - 1]$. It must stay on this cell interval till it reaches a transition state. Depending on the transition state it reaches, it makes one out of two things: either the automaton looks for P_i, picks it up, places it on the next to the right cell and begins to work on the substring $w[j_{i-1}+1...j_i]$; or it looks for P_i, locates it, moves one step rightward, places P_{i+1} on cell $j_i + 1$ (thus, $j_i + 1 = j_{i+1}$) and begins to work on the string $w[j_i...j_{i+1} - 1]$.

It is not hard to formalize the definition of our dp2DFAs, nevertheless we will omit writing down this definition because it happens to be a little bit cumbersome. The key idea is that those automata used their k pebbles to partition their tapes into $k+1$ segments, and then they work on each one of those segments in an independent and sequential way. Moreover, they are designed to consider all the possible partitions of the input string into $k+1$ substrings. Thus, one could say that those automata are tailor-made to deal with concatenations. We will see that it is actually the case, we will see that those automata have real-state processing of concatenations, but before of this we have to check that this new class of automata accepts the regular languages.

Theorem 29: dp2DFA accept the regular languages.

Proof. dp2DFAs accept all the regular languages because a 2DFA is a dp2DFA that never uses its provision of pebbles. Now, we check that those two-way automata, provided with multiple pebbles, can only accept regular languages. Noa Globerman and David Harel studied in [11] a different model of multipebble automata which, they proved, accept the regular languages. A Globerman-Harel automaton is a two-way automaton with k pebbles ($k \geq 0$), say $P_1, \ldots, P_k$, that adheres to the following restrictions:

1) P_{i+1} may not be placed unless P_i is already on the tape, and P_i may not be picked up unless P_{i+1} is not on the tape (Thus the pebbles are placed and picked up in a LIFO style).
2) Between the time P_{i+1} is placed and the time either P_i is picked up or P_{i+2} is placed, the automaton can traverse only the substring located between the current location of P_i and the end of the input word that lies in the direction of P_{i+1}. Moreover, in this substring, the automaton can act only as a 1p2DFA using P_{i+1} as its unique pebble. In particular, it is not allowed to lift up, place, or even sense the presence of any other pebble.

We notice that dp2DFAs adhere to the restrictions imposed on Globerman-Harel automata. Then, a dp2DFA automaton cannot recognize a nonregular language ■

Can we exploit the multiple pebbles to get real-state processing of concatenations? Before studying any possible answer to this question a warning is in order: we are using a new computational resource, the pebbles, which must be quantified.

Definition 30: Let $\circ \in \{\cup, \cdot\}$, we say that dp2DFAs real-state process operation $\circ$ if and only if there exist a constant $C_\circ$ and a polynomial $p(X, Y)$ such that given two dp2DFAs, say $\mathcal{M}$ and $\mathcal{N}$, there exists a dp2DFA $\mathcal{K}$ recognizing the language $L(\mathcal{M}) \circ L(\mathcal{N})$ and such that:

1) The number of states of $\mathcal{K}$ is bounded by $|Q_\mathcal{M}| + |Q_\mathcal{N}| + C_\circ$.
2) The number of pebbles of $\mathcal{K}$ is bounded by $p(\#\mathcal{M}, \#\mathcal{N})$, where $\#\mathcal{X}$ denotes the number of pebbles of automaton $\mathcal{X}$ ($\mathcal{X} \in \{\mathcal{M}, \mathcal{N}\}$).

Notice that dp2DFA can real-state process unions. Thus, we have to focus our attention on concatenations. We will prove that dp2DFAs are able of real-state processing concatenations. First a warm up.

Proposition 31: Let m, n, s be three natural numbers, the language $H_m \cdot H_n \cdot H_s$ can be recognized employing a dp2DFA with two pebbles and $n + m + s + C$ states, where C is a constant that does not depend on the triple (n, m, s).

Proof. First at all we recall that given $n \geq 1$, the language H_n can be recognized by a 2DFA with n states. Thus, we pick three 2DFAs $\mathcal{M}_m, \mathcal{M}_n$ and $\mathcal{M}_s$ recognizing the three languages H_m, H_n and H_s, and such that each one of those three automata has n, m and s states (respectively). Now we construct a dp2DFA $\mathcal{N}$ with two pebbles P_1 and P_2, and which merges together the three automata introduced before. Automaton $\mathcal{N}$ works, on input w, as follows:

Suppose that $\mathcal{N}$ has placed P_1 on cell i, then it checks if the string $w[1...i-1]$ belongs to H_m. If it is not the case it enters a transition state, looks for the pebble, enters the right side, picks up the pebble, places it on the next to right cell and begins once again. Otherwise (i.e. if $w[1...i-1]$ belongs to H_m), automaton $\mathcal{N}$ enters a second different transition state and looks for the right portion of the input string. Notice that, from the exact moment $\mathcal{N}$ enters the right portion of the tape till it picks up the pebble P_1 once again, it is forced to work on the substring $w[i...|w|]$. Along this period of time automaton $\mathcal{N}$ uses P_2 to simulate the pair $\mathcal{M}_n$ and $\mathcal{M}_s$, while checking if $w[i...|w|]$ belongs to $H_n \cdot H_s$. If $w[i...|w|] \in H_n \cdot H_s$, automaton $\mathcal{N}$ halts and accepts the input, otherwise it picks up pebble P_2, picks up pebble P_1, advances one step to the right, places P_1 on this cell and begins once again. ■

We get from the above proposition an interesting corollary

Corollary 32: dp2DFA cannot be linearly simulated by 1p2DFA, even when restricted to the unary case.

Remark 33: We can elaborate on the proof idea used in theorem 27, to get the following more general result: Let $k \geq 2$, and let $m_1, ..., m_k$ be k different prime numbers such that no one of them is a positive integer combination of the others, the language $H_{m_1} \cdot ... \cdot H_{m_k}$ can be recognized using an automaton with $m_1 + ... + m_k$ states and $k-1$ pebbles, while any dp2DFA with $k-2$ pebbles requires $\Omega(\min\{m_i m_j : i,j \leq k \text{ and } i \neq j\})$ states. It implies that for all $k \geq 2$, directed pebble automata with k pebbles cannot be linearly simulated by directed pebble automata with $k-1$ pebbles. It implies that each additional pebble can represent an important gain in computing power. It is important to stress that Globermann and Harel proved a similar result for its model of multipebble automata [11].

Theorem 34: dp2DFAs are able of real-state processing concatenations.

Proof. We only have to elaborate on the proof idea that was used in proposition 31. The rough idea is the following one: suppose that we have two dp2DFA, say $\mathcal{M}_1$ and $\mathcal{M}_2$, each with n_i states and k_i pebbles ($i = 1,2$). We define a new dp2DFA denoted with the symbol $\mathcal{N}$. Automaton $\mathcal{N}$ has $n_1 + n_2 + C$ states, $k_1 + k_2 + 1$ pebbles and works as follows: Suppose that it has detected that the prefix $w[1...i-1]$ belongs to $L(\mathcal{M}_1)$, suppose that it has placed the first $k_1 + 1$ pebbles on the tape and suppose that the last one is placed on cell i. From this exact moment till the moment P_{k_1+1} is picked up again, it works on the suffix $w[i...|w|]$ while simulating the automaton $\mathcal{M}_2$ with the help of the remaining k_2 pebbles ■

It seems that dpDFA cannot real-state process the Kleene star. If we try to use the naive idea employed in the case of concatenations we will promptly realize that we have to use an unbounded number of pebbles. An unbounded number of pebbles seems to be a not admissible resource because , among other things, we need to include some special states in order to handle the provision of pebbles, and it happens that the number of those states increases with the number of pebbles. Thus, such a model of pebble automata seems to be nonfeasible (seems to be nonfinite).

We conjecture that there does not exist a feasible deterministic model of finite automata for which Thompson property holds. Our conjecture implies that there does not exist a feasible deterministic model of finite automata that is able of real-state processing the regular operations. We have that dp2DFAs are able of real-state processing the star free regular expressions and it is the best result that we can achieve so far. Thus, we have

Proposition 35: There exists an algorithm which, on input α (where α is a star free regular expression), computes in linear time a dp2DFA with $O(|\alpha|)$ states and $O(|\alpha|)$ pebbles that recognizes the language $L(\alpha)$.

There is a third computational resource employed by dp2DFA which must be quantified: running time. Two-way automata can work under different running time regimes, and the running time of a given two-way automaton cannot be bounded apriory. Thus, it is natural to ask about the running times of the automata that can be obtained as outputs of the algorithm mentioned in the statement of proposition 35. It is not hard to check that given α, a regular expression, the running time of $\mathcal{M}_\alpha$, which is the dp2DFA computed by the aforementioned algorithm, belongs to $O\left(n^{|\alpha|}\right)$. Moreover, it can be proved that given k, there exists α such that the running time of $\mathcal{M}_\alpha$ belongs to $\Omega\left(n^k\right)$.

Thus, we can conclude that the model of dp2DFAs does not behave well when it comes to the analysis of running time. We conclude with a conjecture

Conjecture 36: There does not exist a class of linear time finite state deterministic automata having real-state processing of star free regular expressions.

VII. Conclusion

One can argue that The Sakoda-Sipser Problem is the question about the state-complexity of simulating a given class of finite automata by another one. Thus, from this very general point of view, The Sakoda-Sipser problem is a question about comparing the state-complexity of different computational tasks when they are analyzed through the lenses of different models of finite automata. We chosen one specific task: processing of regular expressions. Our choice yields a new and meaningful notion: Real-state conversion. The analysis of this new notion allowed us to explain, to some extent, what is special about 1NFAs, and which are the main computational advantages of nondeterminism when one restricts the attention to finite automata. We could prove some preliminary results concerning this new notion, but we feel that it deserves further investigation.

Acknowledgment

The first author would like to thank Universidad Nacional de Colombia and the support provided through the R.P. Hermes 16860.

References

[1] W. Sakoda and M. Sipser, Nondeterminism and the size of two way finite automata, *in STOC 78 Proceedings of the tenth annual ACM symposium on Theory of computing*, 1978, pp. 275286.

[2] A. Ehrenfeucht and P. Zeiger, Complexity measures for regular expressions, *Journal of Computer and System Sciences*, vol. 12, pp. 134146, 1976.

[3] J. Shallit, *A second course in Formal Languages and Automata Theory*. Cambridge, MA: Cambridge University Press, 2009.

[4] K. Thompson, Programming techniques: Regular expression search algorithm, *Communications of the ACM*, vol. 11, no. 6, pp. 419422, 1968.

[5] S. Yu, Q. Zhuang, and K. Salomaa, The state complexities of some basic operations on regular languages, *Theoretical Computer Science*, vol. 125, pp. 315328, 1994.
[6] V. Geffert, C. Mereghetti, and G. Pighizzini, Complementing two-way finite languages, *Information and Computation*, vol. 205, no. 8, pp. 11731187, 2007.
[7] J. Birget, Intersection and union of regular languages and state complexity, *Information Processing Letter*, vol. 43, pp. 185190, 1992.
[8] M. Chrobak, Finite automata and unary languages, *Theoretical Computer Science*, vol. 47, no. 3, pp. 149158, 1986.
[9] J. Chang, O. Ibarra, M. Palis, and B. Ravikunar, On pebble automata, *Theoretical Computer Science*, vol. 44, pp. 111121, 1986.
[10] V. Geffert and L. Istonova, Translation from classical two-way automata to pebble two-way automata, *in 11th International Workshop on Descriptional Complexity of Formal Systems (DCFS 2009)*, Magdeburg, Germany, Jul. 2009, pp. 131140.
[11] N. Globerman and D. Harel, Complexity for two way and multi-pebble automata and their logics, *Theoretical Computer Science*, vol. 169, no. 2, pp. 161184, Dec. 1996.

A Model to Guide Dynamic Adaptation Planning in Self-Adaptive Systems

Andrés Paz* and Hugo Arboleda†
Universidad Icesi, I2T Research Group
Cali, Colombia
*afpaz@icesi.edu.co, †hfarboleda@icesi.edu.co

***Abstract*—Self-adaptive enterprise applications have the ability to continuously reconfigure themselves according to changes in their execution contexts or user requirements. The infrastructure managing such systems is based on IBM's MAPE-K reference model: a *Monitor* and an *Analyzer* to sense and interpret context data, a *Planner* and an *Executor* to create and apply structural adaptation plans, and a *Knowledge manager* to share relevant information. In this paper we present a formal model, built on the principles of constraint satisfaction, to address dynamic adaptation planning for self-adaptive enterprise applications. We formalize, modify and extend the approach presented in [1] for working with self-adaptation infrastructures in order to provide automated reasoning on the dynamic creation of structural adaptation plans. We use a running example to demonstrate the applicability of such model, even in situations where complex interactions arise between context elements and the target self-adaptive enterprise application.**

***Index Terms*—Self-Adaptive Enterprise Applications, Dynamic Adaptation Planning, Automated Reasoning.**

I. Introduction

Currently many Enterprise Applications (EAs) live in dynamic execution contexts, interacting with other systems, and under the influence of stimuli from sources inside or outside the system scope. This may affect their behavior or the levels at which they satisfy agreed quality; however, regardless of these impacts, they still have to fulfill their service quality agreements. On the one hand, the fulfillment of quality agreements is completely and utterly dependent on system architectures, which comprises software architecture, hardware and network infrastructure. On the other hand, in response to ever increasing needs for strengthened responsiveness and resiliency, quality agreements may evolve to reflect this business reality.

Autonomic computing deals with the management of independent components capable of handling both external resources and their internal behavior, which are constantly interacting in accordance with high-level policies. Its required infrastructure usually integrates an autonomic manager, an implementation of the generic control feedback loop from control theory, and managed components. Most autonomic managers are based on the MAPE-K reference model [2], allowing software systems to be adapted to context changes in order to ensure the satisfaction of agreed Service Level Agreements (SLAs). Five elements make up the reference model: *Monitor*, *Analyzer*, *Planner*, *Executor* and *Knowledge Manager*. The *Monitor* continuously senses context conditions and the *Analyzer* interprets and compares the sensed data with SLAs, the *Planner* synthesizes and creates adaptation plans when required, and the *Executor* alters the system's behavior by modifying its structure in accordance with a given adaptation plan. All of them share information through the *Knowledge Manager* element.

In this paper we present a formal model, built on the principles of constraint satisfaction, to address the task of the *Planner* element, *i.e.* dynamic adaptation planning for self-adaptive enterprise applications. Our work in this paper is focused around changing quality agreements while EAs are already operational. This task, however, has a direct impact on system architecture. We consider in this work only the relationships of such quality agreements with software architecture in order to plan the necessary structural adaptations to meet the new quality specifications. We use a running example to demonstrate the applicability of such model, even in situations where complex interactions arise between context elements and the target self-adaptive enterprise application. In the context of product line engineering, decision and resolution models have been used for planning the composition of core assets according to variable configurations that include user requirements, *e.g.*, [3], [4]. All of such approaches, however, deal with problems related to product configuration without taking into account the problem of planning dynamic adaptation of systems.

Some authors have explored different trends for generating reconfiguration plans. For instance [5], [6] use artificial intelligence based on hierarchical task networks and situation calculus, respectively, to plan new web service compositions in an attempt to overcome faults. [7] calculates fuzzy values of quality of service (QoS) levels for available service variants and selects the variants with the nearest QoS levels that fit the context and user requeriements. There are other approaches that implement dynamic adaptation of service compositions, *e.g.*, [8], [9], [10]; however, they neither provide implementation details nor formal specifications of any formal model for planning activities.

In previous work [1], we presented an approach based on constraint satisfaction for product derivation planning in model-driven software product lines. There, we modeled the problem of planning the transformation workflow to derive products as a constraint satisfaction problem. In this paper, we base on such model and we further formalize, modify and extend it for working with self-adaptation infrastructures in order to provide automated reasoning on the creation of structural adaptation plans.

The remainder of this paper is organized as follows. Section II introduces the background of this work. Section III presents our motivating case along with an illustrative example which we use as a running example throughout the following sections. Section IV details our formal model, including the necessary definitions and specifications. Section V describes the automated reasoning that we currently provide. Section VI discusses related work. Finally, Section VII sets out conclusions and outlines future work.

II. Background

A. Autonomic Computing

In [11], IBM researchers Kephart and Chess introduced an architectural approach to realize autonomic computing based on independent elements capable of managing both external resources and their internal behavior. In light of this, autonomic systems are compositions of these autonomic *elements*, constantly interacting in accordance with high-level policies. Each autonomic element is composed of an autonomic manager, an implementation of the generic control feedback loop from control theory, and a managed element, a hardware or software resource, such as a server, a service or a set of interconnected software components.

The autonomic manager, based on the MAPE-K reference model [2], is the infrastructure that allows the software systems to be adapted to unforeseen context changes in order to ensure the satisfaction of agreed Service Level Agreements (SLAs). Comprising this infrastructure is (i) a *Monitor* element that continuously senses relevant context and system control data; (ii) an *Analyzer* element that interprets monitoring events reported by the *Monitor* to determine whether the SLAs are being fulfilled; (iii) a ***Planner*** element that creates a configuration from the variability model according to the context conditions delivered by the *Analyzer* to generate an adaptation plan, which defines the modification required by the deployed system structure and the required parameters to reach a desired system state; (iv) an *Executor* element that realizes adaptation plans, which alters the system's behavior; and (v) a *Knowledge Manager* element sharing relevant information among the other elements.

B. Dynamic Software Product Line Engineering

Software Product Line Engineering (SPLE) is an expanding approach that aims at developing a set of software systems that share common features and satisfy the requirements of a specific domain [12]. While having much in common, product line members still differ in functional and quality requirements. Variability management is the key process in SPLE that is in charge of dealing with the analysis, modeling, design and realization of variants while considering adequate decision making support for building products by using reusable assets.

Variability Models. Variability in SPLE is captured in variability models, such as the Orthogonal Variability Model (OVM) [13], [12]. An OVM is a variability model designed to only document variability; we use OVMs in this paper to document variability in our running example described in Section III-B. In OVMs like the one presented in Figure 1, a variation point (p) represents a variable item in a system and is depicted as a triangle. A variant (v) represents a particular option to instance the variation point and is depicted as a rectangle linked to the variation point by one of three types of relationships. Relationships between variants and variation points may be mandatory, optional or set. A mandatory relationship, depicted in Figure 1 as a solid line, states that if a variation point p is present its child variant v must be present too. An optional relationship, depicted as a dotted line, states that if a variation point p is present its child variant v^p may or may not be present. A set of children variants $\{v_i \mid i = 1, \ldots, z\}$ has a set relationship with their parent variation point p when an interval $[x, y]$ of its children v_i can be included $\{v_i \mid x \leq i \leq y\}$ if their parent is present. This type of relationship is illustrated as variants grouped by an angular solid line with a label describing the interval. Relationships can also exist between variants of different variation points. Such relationships are, namely, requires and excludes; they are drawn as single arrow line and double arrow line respectively. A requires relationship is a cross variant constraint that states that if variant requires variant v_b then if v_a is present, v_b must be present too. An excludes relationship is a cross variant constraint that states that if variant v_a excludes variant v_b then the variants cannot be present at the same time. We give a formal definition of each relationship when we present our proposed model in Section IV.

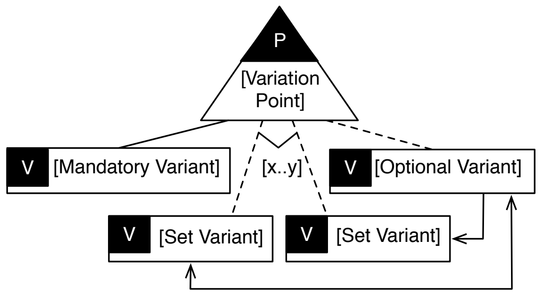


Fig. 1. Orthogonal Variability Model

Dynamic SPLE. Dynamic SPLE [14] extends current product line engineering approaches by moving their capabilities to runtime, helping to ensure that system adaptations lead to desirable properties. It is concerned about the management of reusable and dynamically reconfigurable core assets, facing the challenge of binding variants to such assets, at runtime, when software is required to be adapted according to context changes.

Decision and Resolution Models. When variants are selected by architects at design time (in the context of SPLE), or defined by context conditions at runtime (in the context of dynamic SPLE), concrete core assets must be selected as part of the (re)composition plan. In practice, there is a significant gap between variability at a conceptual level (variation points and variants) and variability at the implementation level (concrete core assets to be deployed). With the objective of closing that gap, *decision* and *resolution models* are used [4], [15]. A decision model relates open decisions and possible resolutions to define the necessary actions to derive product line members in accordance with configurations, which are sets of selected variants. A resolution model is the instance of a decision model, and it is used to create a product line member. In a resolution model all the decisions captured in a decision model are resolved, thus, it defines a product line member including a subset of chosen variants, the core assets required to derive the desired product, and the adaptation that must be performed on the core assets to obtain such product line member.

Variant Interactions. Decision models rapidly become very complex artifacts in the face of many variants and, specially, when *variants interactions* appear. When several variants are combined interactions between them may occur; this means, the presence of one variant affects the behaviour of another. Let suppose a variant v_i is related to a software component c_i, and a variant v_j is related to a software component c_j, an interaction exists when the presence of v_i and v_j in one configuration raises a problem when composing c_i and c_j. Some variant interactions may be benign, planned or desirable, but others, in turn, may have unwanted effects that may disrupt the user from obtaining the expected behavior. Since the variant interactions problem can be arbitrarily complex and computationally difficult to treat, a formal approach is an appropriate and flexible option.

C. Constraint Satisfaction

A great variety of combinatorial problems can be expressed as searching for one or several elements in a vast space of possibilities. In general, the search space is defined as all combinations of possible values for a predefined set of variables. Elements to be searched for are particular values of these variables. In most cases the desired values of the elements are implicitly specified by properties they should satisfy. These properties are known as constraints, which are usually expressed as predicates over some set of variables. Roughly speaking, a problem formulated in this frame is known as a Constraint Satisfaction Problem (CSP) [16].

Solving a CSP consists of two steps: modeling the problem (logical specification) and finding its solutions through a form of search (in this paper we perform a basic backtracking). Modeling involves basically the specification of the variables, their domains and the constraints among them. Solving the CSP through backtracking is an attempt at trying to incrementally build resolution candidates by assigning possible values to the variables. Partial candidates that cannot become a valid solution are discarded. If all variables are bound, a resolution candidate has been found. If, after exploring all possibilities no resolution candidate has been found, then the problem does not have a solution.

III. Motivating Case

A. The SHIFT Framework

Our research group has proposed independent approaches and implementations in the contexts of autonomic computing with the DYNAMICO reference model [17], quality of service (QoS) contract preservation under changing execution conditions with QoS-CARE [18], model-based product line engineering with the FieSta approach [4], [15], automated reasoning for derivation of product lines [1], and the recent (unpublished) contributions regarding quality variations in the automated derivation process of product lines [19]. The required integration of all these efforts in a move to approach automation and quality awareness along the life cycle of enterprise applications has motivated the creation of what we call the *SHIFT Framework*. Figure 2 presents a high-level architectural view of SHIFT's constituting elements.

The `Automated Derivation` region is concerned with providing support for functional and quality configuration and derivation of deployable enterprise applications components and monitoring infrastructure. Generated components are stored in the `Component Repository`, which is managed by a `Knowledge Manager` element; they are an input for the adaptation planning process. The monitoring infrastructure is deployed as part of the `Autonomic Infrastructure` region, which implements the adaptation feedback loop of the DYNAMICO reference model [17].

As part of the `Planner` element, our focus in this paper, SHIFT considers the need for dynamically planning adaptations to application structure based upon quality configurations. Realizing the adaptation plans in the deployed and operating managed Enterprise Application (EA) considers transporting components from their source repository to the corresponding computational resource, undeploying previous versions of them, deploying them into the middleware or application server, binding their dependencies and services, and executing them. In addition, if necessary, to recompile system source code to make measurement interfaces available to the monitoring infrastructure.

In order to obtain the best possible selection of composable components, or *optimum resolution*, when planning an adaptation, we propose in this paper addressing dynamic adaptation planning through a model built on the principles of constraint satisfaction, which will help reasoning upon the set of constraints defined by reachable quality configurations and their relationships with the components in the component repository. Following Section IV will refer to the relationships between components in the component repository and the reachable quality configurations as *decision models*, and all the possible adaptation plans that can be derived from a *decision model* given a specific quality configuration as *resolution models*.

B. Running Example

To illustrate the problem of adapting an EA, while at runtime, when the set of quality agreements (captured as

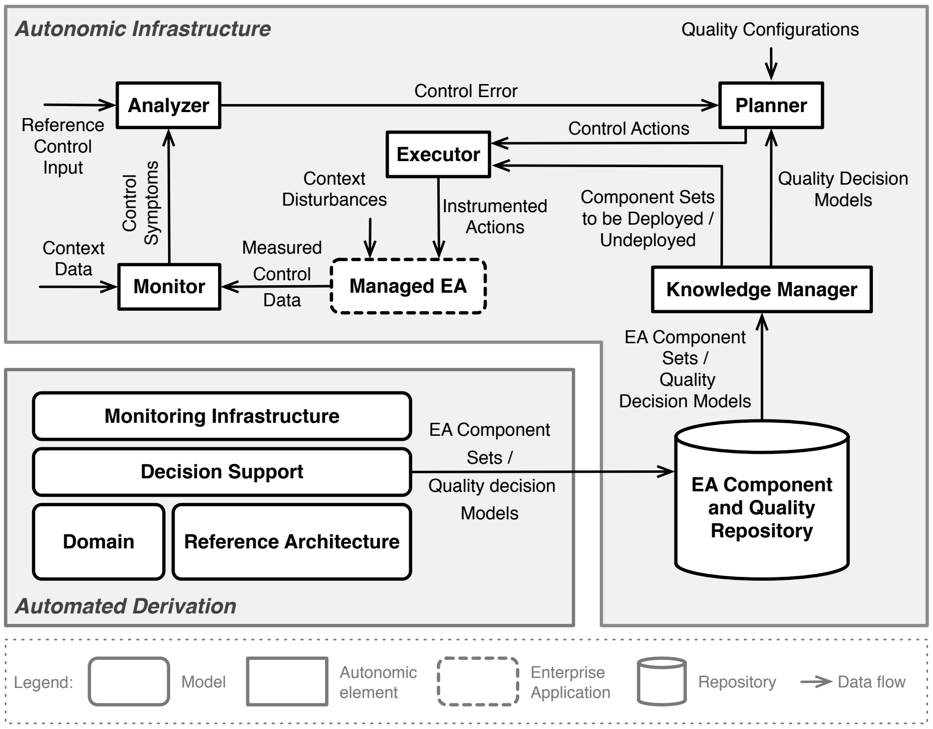
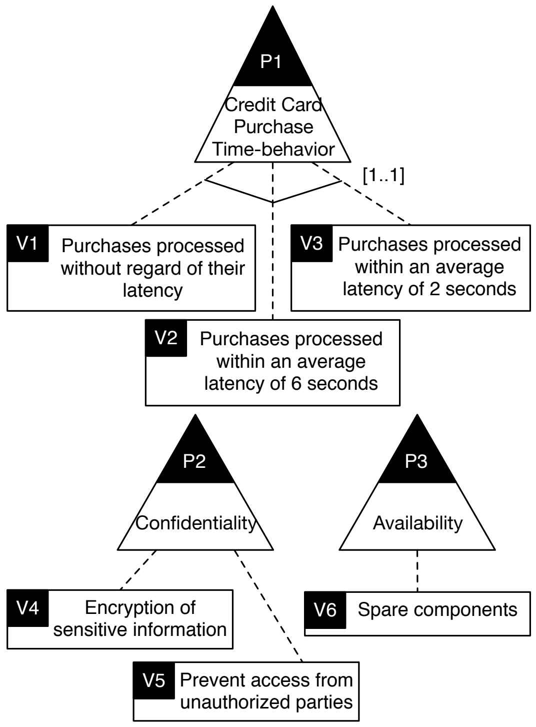


Fig. 2. High-level architectural view of the SHIFT elements.

quality scenarios as explained by Bass *et al.* in [20]) changes, we use the case of a large-scale e-commerce application. We use this case as a running example throughout the following sections. The following sections give the details regarding how the `Planner` element of the SHIFT Framework captures adaptation constraints and reasons upon them to determine possible adaptation plans to satisfy changing context conditions.

With our example e-commerce application there is the need to handle component compositions and adaptations driven by different system quality levels in accordance with varying shopping activities (*e.g.*, special offers on certain products, shopping frenzies). This implies working with varying quality scenarios. Thus, we use the OVM in Figure 3 to capture the different quality scenarios that can be configured for the e-commerce EA. The quality attribute, environment and stimuli fields of a quality scenario represent a variation point. The response field represents a variant. Figure 3 illustrates 3 variation points with all of their variants linked with optional relationships.

Suppose the e-commerce application has been initially deployed fulfilling the requirement of purchase by credit card and the quality configuration corresponds to the selection of quality scenarios `V2` and `V4` detailed in Table I. The *time-behavior* scenario determines an average latency of 6 seconds for purchases with credit card under a load of 1,000 purchases per minute, stochastically. The *confidentiality* scenario specifies all available sensitive information is encrypted to prevent unauthorized access.



P1
Credit Card Purchase Time-behavior
[1..1]
V1 Purchases processed without regard of their latency
V3 Purchases processed within an average latency of 2 seconds
V2 Purchases processed within an average latency of 6 seconds
P2
Confidentiality
P3
Availability
V4 Encryption of sensitive information
V6 Spare components
V5 Prevent access from unauthorized parties

Fig. 3. Variability Model

A component diagram for the implementation of the pur-

TABLE I
QUALITY SCENARIOS FOR THE E-COMMERCE APPLICATION

Quality Attribute	*Performance – Time behavior*
Environment	The application provides a set of services available to concurrent users over the Internet under normal operating conditions.
Stimuli	Users initiate 1,000 purchases with credit card as payment method per minute, stochastically.
Response	Every purchase is processed with an average latency of 6 seconds.

Quality Attribute	*Security – Confidentiality*
Environment	The application provides a set of services that makes sensitive information available to other applications over the Internet.
Stimuli	Another application intercepts data by attacking the network infrastructure in order to obtain sensitive information.
Response	The architecture does not control the other application's access, but information is encrypted in order to prevent access to sensitive information.

chase with credit card requirement is illustrated in Figure 4. This implementation comprises (i) a `Purchase` component that manages the workflow performed for any purchase, (ii) a `Credit Card Authorization` component in charge of performing the workflow to get approval for the transaction with the issuing bank (or credit card association), (iii) a `Risk Tool` component responsible for validating credit card information provided by the customer and the responses sent from the issuing bank, (iv) a `Credit Card Settlement` component that requests the transfer of funds from the issuing bank into the merchant's account, (v) a `Cryptography Manager` component that processes the encryption and decryption of information to and from the issuing bank, and (vi) a `Payment Processor` component managing all communications to the multiple issuing banks. The payment processing behavior exhibited by the previous implementation is specified step by step in Figure 5.

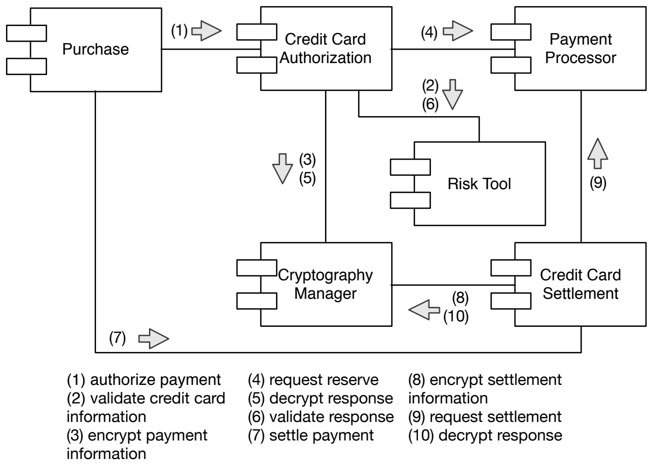


Fig. 5. Component collaboration for initial e-commerce application

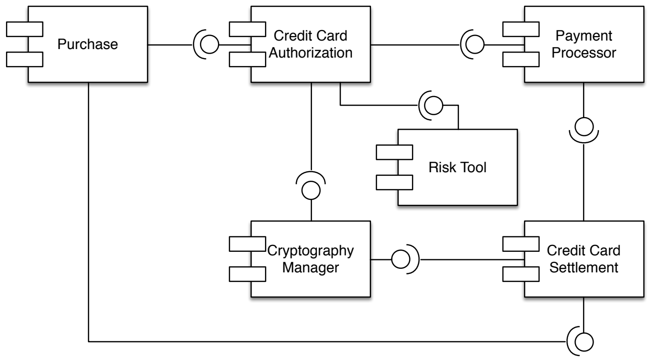


Fig. 4. Partial set of components for initial e-commerce application

For a first adaptation setting suppose now that, while in operation, the application's initial quality configuration has been changed due to an expected peak in system load caused by an upcoming *Cyber Monday* shopping season. Particularly quality scenario `V2` has been replaced by quality scenario `V3`, presented in detail in Table II. Quality scenario `V4` remains selected. In turn, the application's constituent components must be changed to new ones developed with the modified quality configuration in mind.

The adapted implementation for the purchase with credit card requirement is illustrated in Figure 6. This implementation comprises modified versions of the `Purchase`, `Credit Card Authorization` and `Credit Card Settlement` components. These modified components are marked with an asterisk symbol ($\star$). A new component appears, the `Order Manager` component, which provides a consolidated and automated processing of orders. The `Payment Processor` component remains unchanged. The behavior of this implementation is changed due to the structural adaptation performed that streamlined the workflow in comparison with the initial deployment. Figure 7 shows the collaboration steps between the new set

TABLE II
MODIFIED TIME-BEHAVIOR SCENARIO FOR THE E-COMMERCE APPLICATION

Quality Attribute	*Performance – Time behavior*
Environment	The application provides a set of services available to concurrent users over the Internet under normal operating conditions.
Stimuli	Users initiate 20,000 purchases with credit card as payment method per minute, stochastically.
Response	Every purchase with credit card as payment method is processed with an average latency of 2 seconds.

of components.

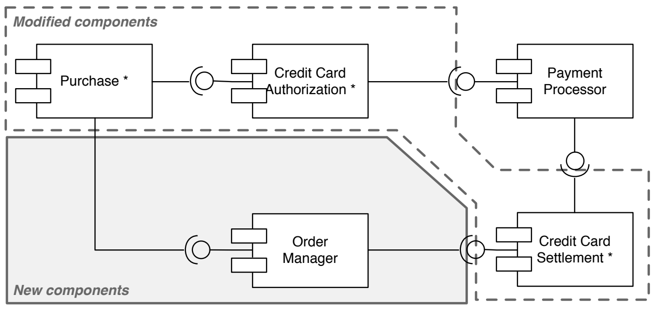


Fig. 6. Partial set of components for adapted e-commerce application

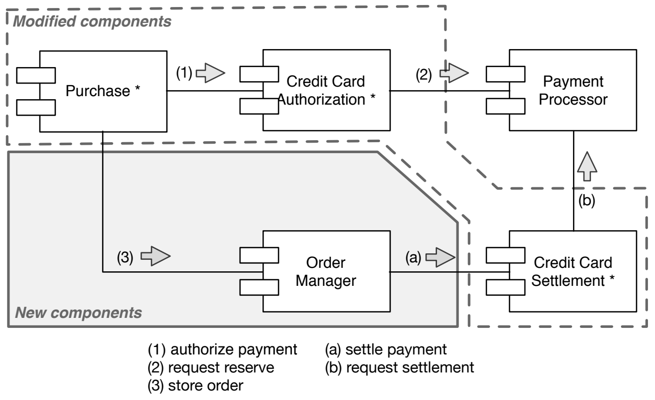


Fig. 7. Component collaboration for adapted e-commerce application

Suppose now, for a second adaptation setting, that to further strengthen the application to cope with the coming sales burst a new quality configuration has been specified selecting quality scenarios `V3`, `V4` and `V6`. The new *availability* scenario `V6` in Table III states that the system initializes and puts into operation spare components when part of the application becomes unavailable. The spare components are initialized from a persistent state before entering into operation. Thus, this response requires the use of persistent storage (*e.g.*, database) to maintain application state and be able to replace failed components. Let's assume that the `Payment Processor` helps meet the *time-behavior* scenario in Table II due to its use of a cache to avoid requests to the database. A variant interaction arises when trying to fulfill both quality scenarios, as the *availability* scenario makes accessing the database mandatory and no caches are permitted. Hence, the *availability* scenario cannot be promoted with the planned adaptation shown in Figure 6. A new solution needs to be designed or the quality scenario needs to be either redefined or dropped.

IV. SELF-ADAPTATION PLANNING

The previous e-commerce application provides an interesting example of the decisions that need to be taken when planning an adaptation to satisfy changing quality scenarios. Manually evaluating all component compositions, their relationships to quality scenarios and quality scenario interactions are costly, time consuming and error-prone; even more when the software system is already operational. In this section we propose an approach addressing dynamic adaptation planning built on the principles of constraint satisfaction.

Benavides *et al.* in [21] propose mapping variability models, particularly feature models, to an equivalent CSP representation in order to deal with the automated analysis of such models. To be able to analyze varying quality scenarios for the creation of adaptation plans we translate input OVMs holding the quality scenarios into a specific CSP representation. Definition 1 formally describes this CSP as a quality model. It is modified from the one presented for the translation of feature models into CSP in [21].

Definition 1: A quality model μ is a three-tuple of the form (Q, W, R); where Q is a finite set of l variables made up of h variation points p and i variants v; W is a finite set of domains made of the variants' configuration states, with a state of 1, if the quality scenario is unselected, or 2, if the quality scenario is selected; and R is a finite set of constraints defined on Q.

$$\begin{aligned} Q = \{ & \{\langle p_k \rangle \mid k = 1, \ldots, h\}, \\ & \{\langle v_i \rangle \mid i = 1, \ldots, n\}\} \end{aligned}$$

$$\begin{aligned} W = \Bigg\{ & \left\{ W_{p_k} = [1..2] \mid \begin{pmatrix} 1 & \text{if } p_k \text{ is unselected} \\ 2 & \text{if } p_k \text{ is selected} \end{pmatrix} \right\}, \\ & \left\{ W_{v_i} = [1..2] \mid \begin{pmatrix} 1 & \text{if } v_i \text{ is unselected} \\ 2 & \text{if } v_i \text{ is selected} \end{pmatrix} \right\} \Bigg\} \end{aligned} \tag{1}$$

$$R = \{r_{mandatory}, r_{optional}, r_{set}, r_{requires}, r_{excludes}\} \tag{2}$$

TABLE III
NEW QUALITY SCENARIO FOR THE E-COMMERCE APPLICATION

Quality Attribute	*Reliability – Availability*
Environment	A subsystem of the application becomes unavailable.
Stimuli	Users initiate transactions to the affected subsystem.
Response	Spare components are initialized and placed into operation.

The set R of Equation 2 contains the following relationship constraints:

Mandatory. A mandatory relationship states that if a variation point p is present its child variant v must be present too.

$$r_{mandatory} = \langle v \geq 2 \Leftrightarrow p \geq 2 \rangle$$

Optional. An optional relationship states that if a variation point p is present its child variant v may or may not be present.

$$r_{optional} = \langle p < 2 \Rightarrow v < 2 \rangle$$

Set. A set of children variants $\{v_i \mid i = 1, \ldots, z\}$ has a set relationship with their parent variation point p when a number of them can be included if their parent is present.

$$r_{set} = \langle x \in [0..f], y \in [1..g] \\ ((g \leq z \wedge p \geq 2) \Rightarrow ((x \times 2) \leq (\sum_{i=1}^{z} v_i) \leq (y \times 2))) \rangle$$

Requires. A requires relationship is a cross variant constraint that states that if variant v_a requires variant v_b then if v_a is present, v_b must be present too.

$$r_{requires} = \langle a, b \in [1..n]((a \neq b) \wedge (v_a \Rightarrow v_b)) \rangle$$

Excludes. An excludes relationship is a cross variant constraint that states that if variant v_a excludes variant v_b then the variants cannot be present at the same time.

$$r_{excludes} = \langle a, b \in [1..n]((a \neq b) \wedge \neg(v_a \wedge v_b)) \rangle$$

In accordance to Definition 1, the OVM in Figure 3 can be translated to the quality model described in Equation 3.

$$\mu_{e-commerce} = (Q_{e-commerce}, W, R) \tag{3}$$

Where

$$Q_{e-commerce} = \{p_1, p_2, p_3, v_1, v_2, v_3, v_4, v_5, v_6\}$$

W and R are as specified in Equations 1 and 2, respectively.

In order to plan an adaptation, values must be assigned to the variables in the Q set conforming to the selection and unselection of quality scenarios. We call this a *quality configuration*. The quality configurations matching the initial scenario and the two adaptation settings for Cyber Monday in Section III are as specified in Equations 4, 5 and 6, respectively.

$$Q^{initial}_{e-commerce} = \{p_1 = 2, p_2 = 2, p_3 = 1, \\ v_1 = 1, v_2 = 2, v_3 = 1, v_4 = 2, v_5 = 1, v_6 = 1\} \tag{4}$$

$$Q^{timebehavior}_{e-commerce} = \{p_1 = 2, p_2 = 2, p_3 = 1, \\ v_1 = 1, v_2 = 1, v_3 = 2, v_4 = 2, v_5 = 1, v_6 = 1\} \tag{5}$$

$$Q^{timebehavior+availability}_{e-commerce} = \{p_1 = 2, p_2 = 2, p_3 = 2, \\ v_1 = 1, v_2 = 1, v_3 = 2, v_4 = 2, v_5 = 1, v_6 = 2\} \tag{6}$$

Promoting a quality scenario may often require several composed components, thus, in this paper we refer as a *componentset* (see Definition 2) to the composition of components promoting a quality scenario. We denote the composition operator as $\oplus$.

Definition 2: A *componentset* c is a composition of g components e.

$$c = \bigoplus_{u=1}^{g} e_u$$

For the example e-commerce application we have identified five *componentsets*: c_1 (see Equation 7), c_2 (see Equation 8), c_3 (see Equation 9), c_4 (see Equation 10) and c_5 (see Equation 11).

$$c_1 = \text{Purchase} \oplus \text{Credit Card Authorization} \oplus \\ \text{Credit Card Settlement} \oplus \text{Risk Tool} \tag{7}$$

$$c_2 = \text{Cryptography Manager} \tag{8}$$

$$c_3 = \text{Payment Processor} \tag{9}$$

$$c_4 = \text{Purchase*} \oplus \text{Credit Card Authorization*} \oplus \\ \text{Credit Card Settlement*} \tag{10}$$

$$c_5 = \text{Order Manager} \tag{11}$$

Table IV shows the relationships established between the quality scenarios (in the remainder of this paper we refer to every variant, *i.e.* response alternative, as one quality scenario) and the identified *componentset*s as presented in Section III. A ✓ indicates the *componentset* requires the quality scenario to be selected in the configuration; on the contrary, an ✗ indicates that the *componentset* requires the quality scenario to be unselected. A "–" indicates the *componentset* is not constraint by the presence of the quality scenario.

TABLE IV
RELATIONSHIPS BETWEEN QUALITY SCENARIOS AND COMPONENTS

Quality Scenarios	*componentset*s				
	c_1	c_2	c_3	c_4	c_5
v_1	✓	–	✓	✗	✗
v_2	✓	–	✓	✗	✗
v_3	✗	✗	✓	✓	✓
v_4	–	✓	–	✓	–
v_5	✗	✗	✗	✗	✗
v_6	✗	✗	✗	✗	✗

One of the main elements of the proposed approach is the decision model. Decision models in our approach relate *componentset*s stored in a component repository (see Figure 2) and quality scenarios to define the necessary actions to adapt an enterprise application in accordance to a configuration of such quality scenarios.

Definition 3: A decision model D is a finite set of $m \times n$ decisions. Each decision d relates one *componentset* c_j with one quality scenario v_i.

$$D = \{\langle d_j^i \rangle \mid j = 1, \ldots, m \wedge i = 1, \ldots, n\}$$

Where

$$d_j^i = \begin{cases} 0 & \text{if } v_i \text{ does not constraint the deployment of } c_j \\ 1 & \text{if } c_j \text{ requires } v_i = 1 \\ 2 & \text{if } c_j \text{ requires } v_i = 2 \end{cases}$$

Table IV maps to the decision model in Equation 12.

$$\begin{aligned} D_{e-commerce} = \{ & d_{c_1}^{v_1} = 2, d_{c_1}^{v_2} = 2, d_{c_1}^{v_3} = 1, d_{c_1}^{v_4} = 0, \\ & d_{c_1}^{v_5} = 1, d_{c_1}^{v_6} = 1, d_{c_2}^{v_1} = 0, d_{c_2}^{v_2} = 0, \\ & d_{c_2}^{v_3} = 1, d_{c_2}^{v_4} = 2, d_{c_2}^{v_5} = 1, d_{c_2}^{v_6} = 1, \\ & d_{c_3}^{v_1} = 2, d_{c_3}^{v_2} = 2, d_{c_3}^{v_3} = 2, d_{c_3}^{v_4} = 0, \\ & d_{c_3}^{v_5} = 1, d_{c_3}^{v_6} = 1, d_{c_4}^{v_1} = 1, d_{c_4}^{v_2} = 1, \\ & d_{c_4}^{v_3} = 2, d_{c_4}^{v_4} = 2, d_{c_4}^{v_5} = 1, d_{c_4}^{v_6} = 1, \\ & d_{c_5}^{v_1} = 1, d_{c_5}^{v_2} = 1, d_{c_5}^{v_3} = 2, d_{c_5}^{v_4} = 0, \\ & d_{c_5}^{v_5} = 1, d_{c_5}^{v_6} = 1\} \end{aligned} \tag{12}$$

A resolution model is a decision model instance, which defines an adaptation plan.

Definition 4: A resolution model S is a finite set of s *componentset* deployments. The deployment s_j is 0 if the *componentset* j should not be deployed, and 1 if the *componentset* j should be deployed.

$$S = \{\langle s_j \rangle \mid j = 1, \ldots, m\}$$

Where

$$s_j = \begin{cases} 0 & \text{if } c_j \text{ should not be deployed} \\ 1 & \text{if } c_j \text{ should be deployed} \end{cases}$$

The resolution for the first adaptation setting in our e-commerce example is presented in Equation 13. The adaptation plan represented in this resolution model indicates that the *componentset*s c_3, c_4 and c_5 should be the ones deployed in order to promote the configured quality scenarios (see Equation 5). According to the decision model in Equation 12, with the available *componentset*s there is no adaptation that can meet the configured quality scenarios in Equation 6 corresponding to the second adaptation setting.

$$\begin{aligned} S_{e-commerce} = \{ & s_1 = 0, s_2 = 0, s_3 = 1, \\ & s_4 = 1, s_5 = 1\} \end{aligned} \tag{13}$$

However, not every possible resolution model is a valid resolution model. A valid resolution model must satisfy the following constraints:

Definition 5: Deployment constraint. A *componentset* must be deployed satisfying the respective deployment condition in the decision model.

$$\forall j \in [1..m] s_j = 1 \Rightarrow \forall i \in [1..n](c_j^i = 0 \vee (c_j^i \neq 0 \wedge c_j^i = v_i))$$

Definition 6: Non-exclusion constraint. Two deployable *componentset*s must not exclude each other.

$$\begin{aligned} & \forall j_1, j_2 \in [1..m](s_{j_1} = s_{j_2} = 1 \wedge j_1 \neq j_2) \\ & \Rightarrow \forall i \in [1..n](c_{j_1}^i = 0 \vee c_{j_2}^i = 0 \vee c_{j_1}^i = c_{j_2}^i) \end{aligned}$$

Definition 7: Completeness constraint. All deployable *componentset*s must take into account all the quality scenarios' states in the quality configuration.

$$\forall i \in [1..n] \exists j \in [1..m](s_j = 1 \wedge c_j^i \neq 0))$$

Definition 8: A self-adaptation plan is a three-tuple of the form (L, T, P); where L is a finite set of variables made up of the quality configuration Q (see Definition 1), the decision model D (see Definition 3) and the set of possible resolution models M (see Definition 4); T is a finite set of domains made up of the domains for the quality configuration (see Definition 1), decision model (see Definition 3) and resolution models (see Definition 4); and P is a finite set of constraints defined on L (see Definitions 1, 5, 6 and 7).

$$\omega_Q^D = (L, T, P)$$

Definition 9: Let ω_Q^D be a self-adaptation plan of the form (L, T, P), its solution space denoted as $sol(\omega_Q^D)$ is made up of all its possible solutions (possible resolution models M). An adaptation is satisfiable if the solution space of ω_Q^D is not empty.

$$sol(\omega_Q^D) = \{\langle S\rangle \mid \forall s_j(s_j \in S \Rightarrow P(s_j) = true)\}$$

V. Automated Reasoning

This section presents how automated reasoning is provided in the *Planner* element. Due to interactions between quality scenarios, and since different component compositions may be available; conflicts between $componentsets$ may arise. Automated reasoning seeks to cope with this issue by providing additional information to get the best possible selection of $componentsets$ when determining an adaptation plan. The proposed approach is able to answer the following questions.

Application. Given a decision model, a quality configuration and a self-adaptation plan, there should be a way of verifying the resolution model's applicability to adapt the specified enterprise application.

Definition 10: Let D be a decision model and Q a quality configuration, a resolution model S is applicable if it is an element of the solutions of the equivalent CSP ω_Q^D.

$$applicable(S) \Leftrightarrow (S \in sol(\omega_Q^D)) \tag{14}$$

Possible resolutions. Once a quality configuration is defined, there should be a way to obtain the potential sets of $componentset$s that promote it.

Definition 11: Let D be a decision model and Q a quality configuration, the potential resolution models that promote Q from D are equal to the solutions of the equivalent CSP ω_Q^D.

$$resolutions(Q, D) = \{\langle S\rangle \mid S \in sol(\omega_Q^D)\} \tag{15}$$

Number of resolutions. A key question to be answered is how many potential resolution models a decision model contains to adapt an enterprise application. The higher the number of resolutions, the more flexible and complex becomes the decision model.

Definition 12: Let D be a decision model and Q a quality configuration, the number of potential resolution models that promote Q from D, or cardinal, is equal to the solution number of its equivalent CSP ω_Q^D.

$$cardinal(Q, D) = \mid sol(\omega_Q^D) \mid \tag{16}$$

Validation. A valid decision model is a model where at least one resolution model can be selected to adapt an enterprise application. That is, a model where ω_Q^D has at least one solution.

Definition 13: A decision model D is valid to adapt an enterprise application promoting quality configuration Q if its equivalent CSP is satisfiable.

$$valid(Q, D) \Leftrightarrow resolutions(Q, D) \neq \emptyset \tag{17}$$

Flexible $componentsets$. A flexible $componentset$ is a $componentset$ that can be applied in self-adaptation plans for the same quality scenario with different combinations of other $componentsets$. Given a set of possible resolution models, there should be a way to find the $componentset$s appearing more than once in such set.

Definition 14: Let M be the set of possible resolution models, the set of flexible $componentset$s in M is equal to the $componentset$s selected to be applicable found in the intersection of M.

$$flexible(M) = \{\langle s\rangle \mid s = 1 \wedge s \in \bigcap M\} \tag{18}$$

Inflexible $componentsets$. An inflexible $componentset$ is a $componentset$ that only makes part of one resolution model. Given a set of possible resolution models, there should be a way to find the inflexible $componentset$s in such set.

Definition 15: Let M be the set of possible resolution models, the set of inflexible $componentset$s in M is equal to the $componentset$s selected to be applicable not found in the intersection of M.

$$inflexible(M) = \{\langle s\rangle \mid s = 1 \wedge s \notin \bigcap M\} \tag{19}$$

Optimum resolution. Finding out the best resolution model according to a criterion is an essential task for self-adaptation in the proposed approach. Given a set of possible resolution models, there should be a way to find the solution that matches the criteria of an objective function. Two objective functions were taken into account. On the one hand, the function that outputs the resolution model with the greater number of applicable $componentset$s to self-adapt an enterprise application; namely max. On the other hand, the function that outputs the resolution model with the least number of applicable $componentset$s to self-adapt an enterprise application; namely min.

Definition 16: Let M be the set of possible resolution models and O an objective function, the optimum solution (max or min) is equal to the optimum space of ω_Q^D.

$$\begin{aligned} max(M, O) &= max(\omega_Q^D, O) \\ min(M, O) &= min(\omega_Q^D, O) \end{aligned} \tag{20}$$

Definition 17: Let ω_Q^D be a CSP, its optimum space, denoted as $max/min(\omega_Q^D, O)$, is made up of all the solutions that maximize or minimize O, respectively.

$$\begin{aligned} max(\omega_Q^D, O) = \{\langle S\rangle \mid \forall S'((S' \in sol(\omega_Q^D) \wedge S' \neq S) \\ \Rightarrow (O(S) \geq O(S')))\} \\ min(\omega_Q^D, O) = \{\langle S\rangle \mid \forall S'((S' \in sol(\omega_Q^D) \wedge S' \neq S) \\ \Rightarrow (O(S) \leq O(S')))\} \end{aligned} \tag{21}$$

VI. Related Work

There are some approaches that have used CSPs for the manipulation of variability models in SPL Engineering. One of the most representative works on the subject was presented in [21], where the authors presented an algorithm to transform feature models into a CSP. The authors proposed to use CSPs to reason on feature models in such a way that they can answer questions such as number of products, filters based on user selections, valid configurations, among others. Several other contributions have been made since then, presenting CSPs as a good complement to SPLs (*e.g.*, [22], [23]). All of them, however, deal with problems related to product configuration without taking into account the problem of planning composition of products.

Some authors have explored different trends for generating reconfiguration plans. For instance, Moore *et al.* [5] use artificial intelligence (AI) based on hierarchical task networks; McIlraith *et al.* [6] propose an AI planner built by adapting and extending Golog [24], which is a logic programming language based on the situation calculus, built on top of Prolog. Other planners, like SHOP2 [25] are hierarchical task network planners, based on the situation calculus. When composing Web services, high level generic planning templates (subplans) and complex goals can be represented by Golog. These approaches, however, do not provide any support for self-adaptive infrastructures. On the other hand, Beggas *et al.* propose in [7] the use of fuzzy logic in adaptation planning. Adaptation controllers calculate fuzzy values for the QoS levels of available service variants, the current context state and user requirements. The variants with the nearest QoS levels that fit the current context state and user requirements will be selected for application.

There are approaches that implement dynamic adaptation of service compositions at the language level *e.g.*, [26], [27]; these can be complex and time-consuming, and with low-level implementation mechanisms for every element of the adaptation infrastructure. Our work is more closely related to approaches using models at runtime, *e.g.*, [8], [9], [10], which implement, tacit or explicitly, the MAPE-K reference model. The recent work of Alférez *et al.* [10] summarizes good practices implementing the MAPE-K reference model. They center their attention on service re-composition at runtime using dynamic product line engineering practices for assembling and re-deploying complete applications according to context- and system-sensed data. Application changes are reflected into the service composition by adding or removing fragments of Business Process Execution Language (WS-BPEL) code, which can be deployed at runtime. In order to reach adaptations, the authors argue that they use Constraint Programming for verifying at design time the variability model and its possible configurations; however, they neither provide implementation details nor formal specifications of any CSP model for planning activities.

VII. Conclusions and Future Work

In this paper we presented a formal model based on the principles of constraint satisfaction for supporting the creation of *Planner* elements in self-adaptation infrastructures. We use CSPs to reason on the set of constraints defined by reachable configurations and their relationships with the components stored in the component repository. We provided formal definitions of the concepts of *quality model*, *decision model*, *resolution model*, *deployment constraint*, *non-exclusion constraint*, *completeness constraint*, *self-adaptation plan* and *solution space*. Our formal model of the *Planner* allows us to answer the following questions: *application*, *possible resolutions*, *number of resolutions*, *validation*, *flexible componentsets*, *inflexible componentsets*, and *optimum resolution*. We used a running example, in the context of enterprise applications and a self-adaptive framework, to demonstrate the applicability of the model, even in situations where complex interactions arise between context elements and the target self-adaptive enterprise application.

As future work, we will extend the model for reasoning on the process of binding components while they are redeployed on system infrastructures. We will also implement a support tool for the model and integrate it into a self-adaptation infrastructure. Other challenges to face in the near future are to perform a validations of our implementation with a case study.

Acknowledgments

This work has been partially supported by grant 0369-2013 from the Colombian Administrative Department of Science, Technology and Innovation (Colciencias) under project SHIFT 2117-569-33721. We thank Miguel Jiménez and Gabriel Tamura for their contributions on the project and the architecture of the SHIFT framework.

References

[1] H. Arboleda, J. F. Díaz, V. Vargas, and J.-C. Royer, "Automated reasoning for derivation of model-driven spls," in *SPLC'10 MAPLE'10*, 2010, pp. 181–188.

[2] IBM, "An architectural blueprint for autonomic computing," *IBM White Paper*, 2006.

[3] P. Tessier, S. Gérard, F. Terrier, and J. M. Geib, "Using Variation Propagation for Model-Driven Management of a System Family," ser. LNCS 3714, 2005, pp. 222–233.

[4] H. Arboleda, R. Casallas, and J.-C. Royer, "Dealing with Fine-Grained Configurations in Model-Driven SPLs," in *Proc. of the SPLC'09*. San Francisco, CA, USA: Carnegie Mellon University, Aug. 2009, pp. 1–10.

[5] C. Moore, M. Xue Wang, and C. Pahl, "An architecture for autonomic web service process planning," in *Emerging Web Services Technology Volume III*, ser. Whitestein Series in Software Agent Technologies and Autonomic Computing, W. Binder and S. Dustdar, Eds. Birkhaeuser Basel, 2010, pp. 117–130.

[6] S. A. McIlraith and T. C. Son, "Adapting golog for composition of semantic web services," in *Proceedings of the Eighth International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, Toulouse, France, 2002, pp. 482–496.

[7] M. Beggas, L. Médini, F. Laforest, and M. T. Laskri, "Towards an ideal service qos in fuzzy logic-based adaptation planning middleware," *Journal of Systems and Software*, vol. 92, pp. 71 – 81, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121213001738

[8] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic qos management and optimization in service-based systems," *IEEE Trans. on Software Engineering*, vol. 37, no. 3, pp. 387–409, 2011.

[9] D. Menasce, H. Gomaa, S. Malek, and J. P. Sousa, "Sassy: A framework for self-architecting service-oriented systems," *IEEE Software*, vol. 28, no. 6, pp. 78–85, 2011.

[10] G. H. Alférez, V. Pelechano, R. Mazo, C. Salinesi, and D. Diaz, "Dynamic adaptation of service compositions with variability models," *Systems and Software*, vol. 91, no. 1, pp. 24–47, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2013.06.034

[11] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[12] K. Pohl, G. Böckle, and F. J. v. d. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[13] F. Bachmann, M. Goedicke, J. Leite, R. Nord, K. Pohl, B. Ramesh, and A. Vilbig, "A Meta-model for Representing Variability in Product Family Development," in *Software Product-Family Engineering*, 2004, pp. 66–80.

[14] S. Hallsteinsen, M. Hinchey, S. Park, and K. Schmid, "Dynamic software product lines," *Computer*, vol. 41, no. 4, pp. 93–95, April 2008.

[15] H. Arboleda and J.-C. Royer, *Model-Driven and Software Product Line Engineering*, 1st ed. ISTE-Wiley, 2012.

[16] E. Tsang, *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[17] N. M. Villegas, G. Tamura, H. A. Müller, L. Duchien, and R. Casallas, "DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems," *LNCS*, vol. 7475, pp. 265–293, 2013.

[18] G. Tamura, R. Casallas, A. Cleve, and L. Duchien, "QoS contract preservation through dynamic reconfiguration: A formal semantics approach," *Science of Computer Programming*, vol. 94, pp. 307–332, 2014.

[19] D. Durán and H. Arboleda, "Quality-driven software product lines," Master's thesis, Universidad Icesi, 2014. [Online]. Available: http://bibliotecadigital.icesi.edu.co/biblioteca_digital/handle/10906/77492

[20] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2012.

[21] D. Benavides, A. Ruiz-Cortés, and P. Trinidad, "Automated reasoning on feature models," *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, vol. 3520, pp. 491–503, 2005.

[22] J. White, D. Benavides, B. Dougherty, and D. Schmidt, "Automated reasoning for multi-step software product line configuration problems," in *Proceedings of the 13th International Software Product Line Conference (SPLC'09)*, San Francisco, US, August 2009, pp. 1–10.

[23] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro, "Automated error analysis for the agilization of feature modeling," *J. Syst. Softw.*, vol. 81, no. 6, pp. 883–896, 2008.

[24] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl, "GOLOG: A logic programming language for dynamic domains," *The Journal of Logic Programming*, vol. 31, no. 1-3, pp. 59 – 83, 1997, reasoning about Action and Change.

[25] U. Kuter, E. Sirin, B. Parsia, D. Nau, and J. Hendler, "Information gathering during planning for Web Service composition," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 2-3, pp. 183 – 205, 2005.

[26] L. Baresi and S. Guinea, "Self-supervising bpel processes," *IEEE Trans. on Software Engineering*, vol. 37, no. 2, pp. 247–263, 2011.

[27] N. C. Narendra, K. Ponnalagu, J. Krishnamurthy, and R. Ramkumar, *Run-time adaptation of non-functional properties of composite web services using aspect-oriented programming*. Springer, 2007.

Acreditación de carreras de Ingenierías en Chile; el caso de Ingeniería Civil en Computación e Informática de la Universidad de Tarapacá, Arica Chile

Hector Beck-Fernández[1]
[1]Escuela Universitaria de Ing. Industrial, Informática y de Sistemas. Arica Chile
email: *hbeck@uta.cl*

Schedule: Tue 20th@17:45, **Room:** C

Se presentan el modelo de acreditación en la Educación Superior en Chile, el procedimiento para alcanzar la acreditación de carreras de Ingeniería, el rol que desempeñan las Agencias de Acreditación y los criterios que se deben satisfacer para ser carrera acreditada. Para ilustrar el proceso completo se presenta el caso de la carrera "Ingeniería Civil en Computación e Informática (ICCI)" de la Universidad de Tarapacá, ubicada en la ciudad de Arica Chile, la cual ya ha sido acreditada en dos oportunidades (2005 y 2010) y actualmente realizó su proceso de autoevaluación con miras a una tercera acreditación. El actual plan de estudios de la carrera ICCI está orientado al desarrollo de competencias concordantes con la nomenclatura de IEEE/ACM y para la definición del perfil profesional se realizó un mapeo desde el desarrollo de competencias a resultados de aprendizajes utilizando el marco de trabajo otorgado por el Syllabus del Conceiving - Designing - Implementing - Operating (CDIO) los que fueron finalmente distribuidos entre las asignaturas.

De la Ingeniería en Computación a las Licenciaturas: reformas curriculares en la Universidad de la República, Uruguay

Ariel Sabiguero Yawelak[1]
[1]Universidad de la República. Uruguay
email: *asabigue@fing.edu.uy*

Schedule: Tue 20th@18:15, **Room:** C

De la Ingeniería en Computación a las Licenciaturas: reformas curriculares en la Universidad de la República, Uruguay

Discusión y resultados

Ernesto Cuadros-Vargas[1], Ariel Sabiguero Yawelak[2]
[1]Universidad Católica San Pablo. Peru,
[2]Universidad de la República. Montevideo Uruguay
email: *ecuadros@spc.org.pe, asabigue@fing.edu.uy*

Schedule: Wed 21st@17:15, **Room:** C

Discusión y resultados de la problemática existente en la región. Propuestas de nuevas estrategias para mejorar y llevar la región a un nivel más competitivo en computación tomando como referencia los últimos avances en el área a nivel mundial.

Digital Equity and Gender Issues in Latin America

Gabriela Marín[1]
[1]Universidad de Costa Rica. Costa Rica
email: *gabriela.marin@ucr.ac.cr*

Schedule: Tue 20th@17:45, **Room:** B

Latin America and the Caribbean is a middle-income region, with the majority of its 42 countries and territories belonging to that category. However, it is a heterogeneous region, ranging from low income countries, as Haiti, to countries which have higher income and are regarded as more developed, like Chile, Mexico, Argentina and Brazil. According to SEDLAC, 34.3 % of the Latin American population is in the middle class (with incomes between $10 a day and $50 a day), and 25.3 % are still under the moderate poverty line of $ 4 a day.

In the last decades, the region has worked to achieve the Millennium Development Goals (MDGs): poverty has been reduced to lower levels, more girls are in school, child mortality has dropped, and diseases are being fought. However, gender issues remain. Too many women still die in childbirth, and more needs to be done to boost gender parity in employment and decision making, as well as, access to education and reproductive health services. Inequality remains a key problem. Progress in the Region has been weaker amongst women, youth, indigenous peoples, afro-descendants and rural populations.

Digital divide is rooted in the very issues that constrain Latin America's overall economic development - income inequality, lack of infrastructure and a still-nascent technological knowledge base according to UNDP. The region, as a whole, suffers from a poor legal framework for the development of the ICT sector, heavy administrative burdens, almost in-existing government prioritization for ICT development, low Internet penetration rates, and pervasive brain drain which undermines the potential for faster growth of the economies' ICT sectors.

An overview on gender inequality on access to technology tools, computers, Internet and education in the region is presented. Moreover, women's low participation on the ICT sector is depicted. Cultural, professional and technological barriers imposed on women participation are analyzed, and some possible actions to reduce such gender biases are proposed. Latin American initiatives to try to promote gender digital equity and the presence of women on the ICT sector are also described.

Acciones a futuro en Latinoamérica - discusión general

Andrea Delgado[1], Raquel Patiño[2]
[1]Universidad de la República. Montevideo Uruguay,
[2]Universidad Católica San Pablo. Arequipa Peru
email: *adelgado@fing.edu.uy, rpatino@ucsp.edu.pe*

Schedule: Tue 20th@18:15, **Room:** B

De acuerdo al avance de tecnología, diversos profesionales van desarrollando cualidades y talentos que ayuden a contribuir con el crecimiento actual. Hoy en día, es importante discutir las cualidades tanto de hombres como de mujeres de apostar por introducirse en este mundo tecnológico. Este espacio estará dedicado a una discusión abierta entre todas y todos quienes asistan a la sesión, para intercambiar ideas y coordinar iniciativas respecto a acciones que puedan impulsarse a futuro en Latinoamérica para promover una mayor equidad de género y una mayor participación de las mujeres en la informática latinoamericana, tanto en la academia como en la industria.

Espelho virtual interativo para simulação de maquiagem

Filipe Morgado Simões de Campos, Carlos H. Morimoto
Departamento de Ciência da Computação - IME
Universidade de São Paulo
São Paulo, Brasil
Email: {fmsc, hitoshi}@ime.usp.br

***Abstract*—A makeup simulator can be used to create interactive virtual environments which users can try makeup products in a fast, low-cost and flexible way. To succeed, such simulator should fulfill four characteristics, flexibility to apply makeup products, real-time processing, realistic simulation and be low-cost. The contribution of the project was an interaction model for a makeup simulator and an algorithm to simulate foundation, eye shadow and lipstick, both respecting the four characteristics mentioned above.**

***Resumo*—Um simulador de maquiagem pode ser utilizado para criar um ambiente virtual interativo capaz de tornar o processo de escolha e experimentação de maquiagens algo rápido, barato e flexível. Para seu sucesso, tal simulador deve respeitar quatro critérios, a flexibilidade na aplicação dos produtos, processamento em tempo real, realismo da simulação e baixo custo. O projeto desenvolvido pode contribuir com uma proposta de um modelo de interação para um simulador de maquiagem e um algoritmo para a simulação de base, batom e sombra, ambos respeitando as quatro características citadas anteriormente.**

***Keywords*—makeup simulation; image processing; human computer interaction.**

***Palavras-chave*—simulação de maquiagem; processamento de imagens; interação humano computador.**

I. Introdução

A. Motivação

A maquiagem faz parte do dia a dia das pessoas, mas experimentar e escolher tais produtos para cada ocasião pode ser um processo demorado e complexo. Um simulador de maquiagem pode ajudar uma pessoa a realizar essas tarefas de forma mais prática.

Computacionalmente, simular maquiagem continua sendo um problema desafiador como pode ser observado nos artigos apresentados em II. Comercialmente este trabalho pode ser o início da base tecnológica de um produto inovador para o mercado de cosméticos que nacionalmente fatura US$42 bilhões tornando-se o terceiro maior do mundo e, mantendo o ritmo de crescimento atual, será o segundo maior mercado mundial até 2017 [1].

B. Objetivo

Esse trabalho tem como objetivo desenvolver um ambiente virtual interativo para simulação de maquiagem que permita a escolha dos produtos e aplicação desses na imagem do usuário. Tal simulador deve possibilitar que o usuário experimente um produto de maquiagem de forma mais fácil e conveniente do que utilizando os cosméticos reais, podendo observar os resultados de forma mais rápida (sem a necessidade de preparação, além da aplicação e remoção da maquiagem ocorrerem instantaneamente) e barata (não é necessário comprar o produto ou gastar produtos já existentes).

Existem diversos cenários em que esse ambiente virtual pode ser útil. No ambiente doméstico, quando o usuário for se maquiar ele pode testar combinações de todas as maquiagens que possui antes de definir quais serão as maquiagens escolhidas para a ocasião. Ainda, ele também poderia testar outras maquiagens que não possui, mas que combinam com as que o usuário já tem, realizando uma compra *online* para utilizar na próxima ocasião. O ambiente virtual também pode ser usado na Internet permitindo a criação de novos estilos de maquiagem com a colaboração de amigos em redes sociais.

Em uma loja de produtos de maquiagem, a simulação pode ser útil para mostrar o resultado de diversos produtos, demonstrar como estes podem ser combinados entre si e ensinar ao cliente o correto uso e respectivas combinações. Além disso, pode-se destacar também que o processo de experimentar tais produtos se torna mais higiênico, já que não é necessário manusear e aplicar um mesmo produto físico em mais de uma pessoa.

Já em um salão de beleza, pode-se mostrar para o cliente como ele ficará após a aplicação das maquiagens que o maquiador está considerando, com o intuito de verificar sua satisfação antes de ser maquiado.

Outra abordagem seria utilizar o simulador como ferramenta para o treinamento de profissionais desse setor.

C. Definição do problema

Para criar o ambiente virtual do simulador de maquiagem desejado, podemos considerar duas grandes questões a serem resolvidas: a interação com esse sistema e o algoritmo responsável pela simulação da maquiagem.

Com relação à interação, o sistema deve possibilitar que o usuário se maqueie de forma semelhante ao que ele já faz com produtos reais e em um ambiente familiar. Dado que um dos locais mais comuns para se maquiar é em frente ao espelho, este foi escolhido como metáfora para a interação do simulador. Assim, o sistema funciona permitindo a escolha da maquiagem e também sua aplicação sobre a imagem da face do usuário como se ele estivesse maquiando seu reflexo no espelho. Para que a aplicação da maquiagem seja semelhante ao que é realizado com produtos reais, o simulador deve oferecer a liberdade para que o usuário aplique a maquiagem que ele desejar em qualquer região de sua face.

Por se tratar de um espelho, a imagem a ser exibida deve corresponder ao reflexo da imagem da pessoa no espelho e portanto deve ser um vídeo e não uma imagem fixa. Isso gera dois problemas a serem resolvidos: o posicionamento da maquiagem virtual na face do usuário deve acompanhar o movimento da face (sem a necessidade de intervenção humana para marcação de pontos faciais) e também deve ser realizado em tempo real de forma que a taxa de quadros por segundo do vídeo não seja afetada.

Com relação ao algoritmo para a simulação de maquiagem, ele deve ser capaz de simular como a face do usuário ficará após a aplicação da maquiagem. Tal simulação deve mimetizar as alterações das propriedades ópticas e de textura da pele de forma convincente, já que o usuário espera ver sua imagem como se ele estivesse se vendo em um espelho. Além disso, o método deve ser capaz de ser processado em tempo real para que possa ser utilizado em vídeo, deve ser flexível para permitir a aplicação independente de diversas maquiagens por qualquer região da face do usuário e não deve requisitar o uso de equipamentos especiais que encareceriam a solução.

D. Desafios

Os desafios desse projeto são: realismo, tempo real, flexibilidade e baixo custo. Para o primeiro, a simulação de maquiagem deve alterar de forma adequada as propriedades ópticas e de textura da pele para que seu resultado seja convincente, para o segundo, a simulação de maquiagem deve ser computada rápida o suficiente para que ela possa ser usada como um reflexo de espelho, para o terceiro, a interação com o simulador deve ser natural, se assemelhando com o que as pessoas já estão acostumadas ao interagir com um espelho comum, porém permitindo escolher e experimentar diversos produtos com praticidade e, para o quarto, o sistema não deve utilizar equipamentos de alto custo.

E. O Projeto

Este projeto consiste em um espelho virtual interativo para simulação de maquiagem. Com relação à interação, foi criada uma interface que é composta por um monitor sensível a toque fazendo o papel de espelho e, posicionado acima dele, um sensor RGBD para capturar o vídeo do usuário. A interação com o sistema ocorre por meio do toque no monitor, permitindo a escolha da maquiagem desejada e também sua aplicação na imagem do espelho. A interface permite que o usuário possa aplicar a maquiagem da forma desejada em sua face criando efeitos diversos, por exemplo, intensificando determinadas regiões ou passando mais de um produto de maquiagem na mesma região. Além disso, uma vez que a maquiagem for aplicada, o sistema é capaz de manter seu posicionamento durante a movimentação da face do usuário sem que isso interfira na fluidez da imagem do espelho e sem a necessidade de marcadores.

A representação computacional de cada produto de maquiagem será chamada de *maquilet* e conterá as propriedades responsáveis pelo efeito que esse produto causa

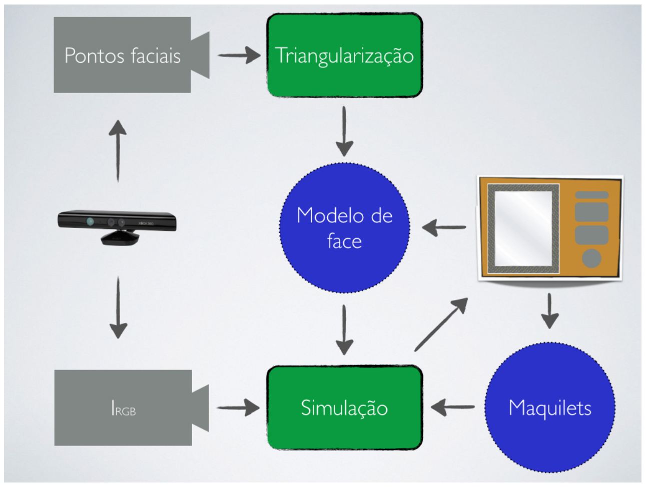


Figura 1. Arquitetura geral.

na pele. A figura 1 apresenta a arquitetura geral do sistema proposto. Nela nota-se que o sensor RGBD utilizado (ilustrado pelo Kinect) fornece dois tipos de dados (cinza), o vídeo da câmera RGB que conterá a imagem da face do usuário, I_{RGB}, e a localização de pontos faciais da face do usuário que são utilizados na triangularização (verde). Em azul, temos os modelos a serem usados na simulação, o modelo de face é criado a partir dos dados da triangularização. Já as maquiagens a serem simuladas são modeladas pelos *maquilets* correspondentes a essas maquiagens. A partir da imagem da face do usuário juntamente com o modelo de face e os *maquilets*, o módulo Simulação (verde) realiza a simulação de maquiagem na imagem da face do usuário que é exibida pela interface (laranja) possibilitando a interação do usuário com o sistema.

O objetivo com a triangularização é obter uma maneira para manter o posicionamento da maquiagem aplicada na face do usuário durante a movimentação de seu rosto. Quanto menor forem os triângulos gerados pela triangularização, mais preciso será o mapeamento da maquiagem na face do usuário. No protótipo desenvolvido, o mapeamento foi realizado atribuindo para cada triângulo cada *maquilet* que foi aplicado nessa posição e que deve ser simulado nessa região da face ocupada pelo triângulo.

Para a simulação de maquiagem, o método desenvolvido simula a aplicação de sombra, base e batom que podem ser aplicados individualmente em qualquer região da imagem da face. Além dessa flexibilidade, o algoritmo é capaz de ser computado em tempo real, não necessita de equipamentos especiais para seu funcionamento e apresenta resultados convincentes.

Para esse projeto, a simulação de maquiagem se concentrará em alterar a textura da pele e mudar sua cor. Assim, o módulo de simulação foi dividido em dois submódulos, Textura e Cor. O primeiro é responsável por tornar o tom de pele mais uniforme e suavizar ou realçar os detalhes da pele e o segundo pela introdução da cor da maquiagem.

Para a simulação da textura, a imagem da face é dividida em cinco camadas distintas, cada uma representando uma faixa diferente de frequência, sendo que o número de camadas foi obtido de forma empírica. Cada uma dessas camadas é suavizada com intensidades distintas para mimetizar o efeito da maquiagem na textura da pele.

Para a simulação de cor, a imagem é transformada para o espaço de cor La*b*, já que ele se mostrou melhor para separar a informação de cor contida na imagem do que outros espaços como o RGB ou HSL. Para o La*b*, cada um dos canais torna-se uma camada e é tratada de forma independente. O canal L permanece como o original, já que o objetivo é alterar apenas as cores da imagem preservando sua iluminação. Para o canal a* e b*, é calculada uma média ponderada para cada pixel com o mesmo canal de uma imagem com a cor do *maquilet* correspondente.

F. Contribuições

As principais contribuições deste trabalho estão relacionadas com a proposta de um modelo de interação para um simulador de maquiagem e um método para simular a aplicação de maquiagens em faces em tempo real.

O modelo de interação proposto neste trabalho contempla uma interface que faz alusão a um espelho para aplicar independentemente os produtos de maquiagem desejados em qualquer região da face tocada no espelho. Por estar maquiando a imagem do usuário, o sistema deve realizar a simulação em um vídeo da imagem do usuário e deve manter o correto posicionamento da maquiagem conforme o usuário movimenta a face em frente ao espelho sem o uso de marcadores. Como descrito em II-B, foi encontrado apenas um trabalho de Interação Humano Computador que faz alusão ao espelho e simula maquiagem. Nesse trabalho o reflexo do espelho é um modelo 3D do usuário que mimetiza seus movimentos e a maquiagem é simulada nesse modelo. Além disso, existe a necessidade de marcadores para o adequado funcionamento do sistema. Ou seja, o usuário precisa passar por algumas etapas preparatórias antes de iniciar o uso do sistema como criar seu modelo 3D, que não será tão realista quanto a imagem da câmera, e posicionar marcadores.

O método para simulação de maquiagem em faces proposto neste trabalho se diferencia dos demais trabalhos descritos em II-A, pois tais trabalhos não apresentam ao mesmo tempo as quatro características do nosso método, que são: flexibilidade (permitir aplicar cada maquiagem individualmente em qualquer região da face), tempo real (rápido o suficiente para aplicar maquiagem em um vídeo), baixo custo (não necessita de equipamentos especiais) e realismo (resultado da simulação é convincente).

II. Trabalhos Correlatos

Nesta seção serão apresentados trabalhos relacionados com esse projeto no contexto do algoritmo de simulação de maquiagem e da interação com o simulador de maquiagem.

A. Simulação de maquiagem

Alguns métodos já foram propostos com objetivos relacionados à simulação de maquiagem. Eles podem ser divididos em dois grupos: transferência de maquiagem e aplicação de maquiagem. Em geral, trabalhos de transferência de maquiagem se baseiam em entender o que corresponde à maquiagem em uma imagem de uma face já maquiada para transferir tal maquiagem para uma outra face. Tipicamente, tais trabalhos têm como limitações iguais condições de iluminação e pose da face entre as imagens. Além disso, as possibilidades de maquiagem estão restritas às faces exemplos já maquiadas. Os trabalhos de aplicação, apesar de terem objetivos semelhantes a este artigo, tratam a aplicação da maquiagem de forma simples, não obtendo um realismo adequado ou então necessitam de equipamentos especiais para seu funcionamento.

1) Transferência de maquiagem: Seja A^* a imagem contendo uma face F com maquiagem M; A a imagem contendo F, porém sem maquiagem e B uma outra imagem contendo uma face G, possivelmente de outra pessoa e sem maquiagem. A transferência de maquiagem é definida como o processo de compor uma imagem B^* com a face G maquiada da mesma forma que F em A^*.

Para que o procedimento seja bem sucedido, deve-se realizar apenas a transferência da maquiagem M presente em A^* para a G em B, ou seja, sardas, pintas e manchas existentes em A^* não devem estar presentes na imagem resultante da transferência, B^*, porém essas mesmas características, se existirem em B, devem ser preservadas em B^*.

O método proposto por Tong et. al. [2] usa duas imagens modelo da mesma pessoa para aprender os efeitos da maquiagem na face, uma sem maquiagem, A, e outra com maquiagem, A^*. As duas imagens necessitam das mesmas condições de iluminação e pose do rosto.

Antes da transferência de maquiagem propriamente dita, ocorre um procedimento para remoção de sobrancelha e cílios, bem como sardas, pintas e manchas na pele. Assim, o procedimento de transferência pode ser realizado sem a intervenção dessas características faciais e ao final do procedimento as características presentes em B são devolvidas para a face em B^*.

A transferência da maquiagem é computada pixel a pixel a partir de A e A^* e representa a mudança de cor e refletância da pele criada pela maquiagem após ela ser aplicada.

O método de Tong et. al. [2] necessita de uma imagem modelo com e sem maquiagem sob condições controladas de iluminação e pose. Isso torna a aplicação do método de transferência de maquiagem restrita a situações onde é possível obter imagens desse tipo. Outro ponto a ser notado é que a solução proposta se baseia em aprender a diferença de uma imagem com e sem maquiagem comparando uma com a outra, ou seja, a mudança da pele decorrente da aplicação da maquiagem para uma determinada pessoa e não um modelo genérico da maquiagem usada que poderia ser utilizado para simular a aplicação de qualquer maquiagem.

Scherbaum et. al. [3] descrevem um sistema que consiga sugerir o estilo de maquiagem que um especialista em maquiagem escolheria para uma determinada face. Para isso, foi construído um banco de dados de imagens de pessoas sem e com maquiagem feita por um profissional para que fosse possível aprender como esse profissional trabalha cada face e também o efeito causado pela maquiagem em cada uma delas. O aprendizado do efeito da maquiagem é realizado de forma semelhante ao trabalho de Tong et. al. [2], porém a aplicação da maquiagem é simulada em um modelo 3D criado a partir de padrões de luz aplicados na face da pessoa. Para esse procedimento, a imagem da face é quebrada em várias camadas, entretanto, todas elas só podem ser obtidas com o método utilizado para a criação do modelo 3D, limitando a aplicação do método.

O método proposto por Guo e Sim [4] usa apenas uma imagem modelo, A^*, para aprender os efeitos da maquiagem na face. A dependência de condições semelhantes de iluminação e pose do rosto ainda existe, mas ocorre somente entre A^* e B.

Diferentemente do trabalho de Tong et. al, o método de Guo e Sim se baseia em analisar A^* para extrair o efeito causado pela maquiagem.

Tal processo é realizado decompondo A^* e B em três camadas, para então transferir a informação de cada camada de uma face para a camada correspondente da outra face. As camadas utilizadas são:

- Estrutura da face: Contém a disposição das características faciais como olhos, boca e nariz.
- Detalhes da pele: Contém a textura da pele incluindo falhas, sinais e rugas.
- Cor: Contém a representação da cor de forma isolada com relação às outras camadas.

Para realizar a transferência de maquiagem, a camada de detalhe da pele resultante é calculada por meio de uma soma ponderada, a camada de cor é calculada utilizando uma média ponderada e, para transferir os efeitos de destaque e sombreamento causados pela maquiagem presentes na camada de estrutura, foi adaptado um método de edição baseado em gradiente, com o objetivo de adicionar apenas as grandes mudanças presentes na camada de estrutura de A^* para B^*. Ao assumir isto, foi considerada iluminação uniforme em A^*, caso contrário esta também seria transportada para a imagem final.

Apesar do artigo de Guo e Sim [4] ir além do trabalho de Tong et. al. [2], pois tenta extrair da face modelo o que seria a maquiagem e não apenas aprender a diferença entre a face modelo sem e com maquiagem, os dois artigos tratam sobre transferência de maquiagem e assim dependem de um exemplo da maquiagem já aplicada para realizar sua simulação. Dessa forma, eles não introduziram um modelo genérico que permitisse simular os efeitos da maquiagem no rosto de uma pessoa. Ambos também necessitam que a pose da face alvo seja parecida com a da imagem modelo e ficam restritos a aplicar o estilo de maquiagem da imagem modelo, não sendo possível simular a aplicação de diferentes maquiagens de forma independente.

2) Simulação de maquiagem: Dhall et. al. [5] tratam sobre aplicação automática de maquiagem baseado no gênero e cor de pele da pessoa. Sobre o escopo da aplicação de maquiagem, a pele é preparada utilizando uma suavização Gaussiana seguida de uma dilatação, com o objetivo de remover pequenas marcas e sinais na pele, para então aplicar a maquiagem na face. A simulação ocorre através da modificação dos valores de matiz e saturação do espaço de cor HSV para que eles sejam os mesmos dos padrões adequados para cada gênero e cor de pele que se tem salvo em um banco de dados. Para a maquiagem nos lábios, utiliza-se para cada pixel no espaço de cor RGB uma média ponderada.

Kim e Choi [6] criaram um sistema interativo para aplicar maquiagem em um modelo 3D com feedback háptico e visão estéreo. No que diz respeito à aplicação da maquiagem, ela é feita através de uma média ponderada entre as cores da textura da pele capturada pelos *scanners* 3D utilizados no projeto e as cores da maquiagem desejada.

Huang et. al. [7] apresentam uma abordagem baseada em modelos físicos que simulam o comportamento da luz ao interagir com a pele. Para o funcionamento do método, é necessário medir propriedades ópticas da pele e das maquiagens a serem simuladas para que seus respectivos modelos possam ser criados. Tais medidas necessitam de equipamentos especiais.

Os trabalhos de Dhall et. al. [5], Kim e Choi [6], se limitam a modelar a simulação dos efeitos da maquiagem na pele com uma média ponderada no espaço de cor HSV ou RGB. Essa abordagem, em geral, não trás resultados realistas. Já o trabalho de Huang et. al. [7], apesar de ser computacionalmente eficiente quando implementado em GPU, necessita conhecer propriedades ópticas da pele em que a simulação ocorrerá para seu funcionamento. Isso requer o uso de equipamentos especiais para cada usuário, o que limita a aplicação do método.

B. *Interação Humano Computador*

O artigo de Kim e Choi [6] se propõe a criar um sistema interativo para aplicar maquiagem em um modelo 3D com *feedback* háptico e visão estéreo. O usuário vestiria um óculos especial para enxergar seu próprio modelo 3D em um monitor estéreo e utilizaria um dispositivo háptico para interagir com o modelo, sendo que esse dispositivo faria o papel de, por exemplo, um pincel de maquiagem utilizado na aplicação do cosmético. O modelo 3D da face da pessoa é adquirido com escâneres 3D comerciais que captam a nuvem de pontos e textura da face. O sistema simula aplicação de base, maquiagens coloridas e brilho labial utilizando a pressão feita no dispositivo háptico para determinar a quantidade de maquiagem a ser aplicada. Também é possível remover a maquiagem aplicada.

No artigo de Iwabuchi et al. [8], os autores se propõem a criar um espelho inteligente de maquiagem com o objetivo de tornar o processo de aplicação desses produtos mais fácil e divertido. Tal sistema não contempla simulação de maquiagem, sendo apenas um espelho com funcionalidades a mais do que um espelho comum para auxiliar o usuário durante a aplicação da maquiagem.

No artigo de Rahman et al. [9], os autores se propõem a desenvolver uma interface para um espelho inteligente de maquiagem que torne o momento da escolha desses produtos algo mais conveniente e eficiente. Assim, foi desenvolvido um sistema chamado pelos autores de *sensory augmented smart interaction mirror* (SIM), que contempla um monitor fazendo o papel de espelho juntamente com a simulação da maquiagem escolhida pelo usuário, além da sugestão de outros produtos de maquiagem como, por exemplo, com menor preço, baseado no histórico de produtos provados. A imagem da face da pessoa apresentada no espelho é um modelo 3D que mimetiza em tempo real os movimentos feitos pelo usuário ao utilizar o sistema. Tal modelo precisa ser criado antes de se utilizar o SIM.

Para seu funcionamento, o sistema possui uma câmera comum para capturar a imagem do usuário e uma câmera infravermelha utilizada para rastrear a face (são colocados dois emissores de infravermelho, um em cada orelha do usuário) e as mãos do usuário (são colocados emissores de infravermelho nos produtos). Para obter a informação de que produto o usuário deseja aplicar, todos os produtos utilizados no sistema possuem um marcador RFID. Assim, quando o usuário quer usar o produto, ele o aproxima do leitor RFID para que o sistema carregue a textura adequada para essa maquiagem.

O trabalho de Hanafusa et al. [10] se propõe a desenvolver um sistema de maquiagem que tenha uma interface adequada para portadores de deficiência visual. Assim, foi desenvolvido um sistema que ensina deficientes visuais a aplicar maquiagem por meio de instruções sonoras, verifica se o batom foi aplicado corretamente e também verifica o formato da sobrancelha após a maquiagem. Para verificar a correta aplicação da maquiagem, o sistema compara a imagem depois da aplicação do produto com uma imagem capturada sem maquiagem ou com ela aplicada corretamente para verificar se existem diferenças indesejáveis.

No artigo de Nakagawa et al. [11], tem-se como objetivo propor um sistema que ajude o usuário a experimentar novos métodos e variações de maquiagem com os produtos utilizados no dia a dia. Criou-se então um aparato chamado de Sistema Inteligente de Maquiagem (*Smart Makeup System, SMS*) para facilitar o compartilhamento entre amigos de seus históricos de maquiagens por meio de fotos ou informações sobre os produtos utilizados. O sistema é composto por três espelhos, um ao centro, um à esquerda e o da direita com um espaço no centro com um monitor e uma câmera USB. Além disso, existe um leitor de RFID na cesta de produtos, um computador e uma lâmpada no topo do espelho central.

C. Resumo

Para o contexto do algoritmo de simulação de maquiagem, podemos analisar a tabela I. Nela pode-se perceber que nenhum dos trabalhos abordados possui ao mesmo tempo as quatro características desejadas para o método de simulação de maquiagem que são: flexibilidade, baixo custo, realismo e tempo real. Na tabela, o critério de "Flexibilidade" diz respeito à possibilidade de aplicar uma maquiagem qualquer desejada pelo usuário em qualquer região da face, ou seja, a aplicação da maquiagem não está presa a um estilo ou produtos de maquiagem específicos. O critério de "Baixo custo" significa que o algoritmo para simular a maquiagem requer ou não o uso de algum equipamento especial (o artigo pode ter usado um dispositivo especial como em [6], mas ele não era estritamente necessário para o funcionamento do algoritmo). Para ser considerado "Realista", o artigo deve apresentar resultados convincentes para a simulação e quanto ao critério "Tempo real", o artigo deve descrever que seu algoritmo pode ser computado em tempo real.

Com relação aos dois principais artigos que tratam sobre transferência de maquiagem, apesar do artigo [4] ir além do trabalho [2], pois tenta extrair da face modelo o que seria a maquiagem e não apenas aprender a diferença entre a face modelo sem e com maquiagem, os dois artigos tratam sobre transferência de maquiagem e assim dependem de um exemplo da maquiagem já aplicada para realizar sua simulação. Dessa forma, eles não introduziram um modelo genérico que permitisse simular os efeitos da maquiagem no rosto de uma pessoa. Ambos também necessitam que a pose da face alvo seja parecida com a da imagem modelo e ficam restritos a aplicar o estilo de maquiagem da imagem modelo, não sendo possível simular a aplicação de diferentes maquiagens de forma independente. Essas características relacionadas à transferência de maquiagem também estão presentes no trabalho [3], já que a simulação é baseada no método descrito em [2].

Quanto aos outros artigos, os trabalhos [5] e [6], se limitam a modelar a simulação dos efeitos da maquiagem na pele com uma média ponderada no espaço de cor HSV ou RGB. Essa abordagem, em geral, não trás resultados realistas. Já em [7], apesar de ser computacionalmente eficiente quando implementado em GPU, necessita conhecer propriedades ópticas da pele em que a simulação ocorrerá para seu funcionamento. Isso requer o uso de equipamentos especiais para cada usuário, o que limita a aplicação do método.

Para o contexto da interação do simulador de maquiagem, podemos analisar as tabelas II e III. Na primeira tabela temos o comparativo com características da interação que estão relacionadas com a simulação de maquiagem e na segunda tabela temos algumas outras características importantes a serem consideradas.

Na tabela II temos a coluna "Simula maquiagem", nela indicamos se a interface proposta possui ou não a simulação de maquiagem. Para "Aplicação", diferenciamos os trabalhos que permitem aplicar uma maquiagem virtual qualquer livremente pela face dos que não permitem isso, nessa característica tivemos um trabalho que não tornou claro em seu texto como que a maquiagem era aplicada, então ele foi marcado como N.E. (Não Especificado). "Apresentação" significa a forma com que a imagem do usuário é apresentada para ele, ela pode ser por meio de um modelo 3D ou a própria imagem do usuário captada utilizando uma câmera. Quanto à coluna "Marcadores", esta especifica se o artigo utilizou algum tipo de marcador para rastrear a face ou algum outro objeto de interesse. Por fim, a coluna "Pronto para uso" esclarece se é possível usar a interface proposta imediatamente ou se é necessário algum processo de calibração ou criação de modelos antes de começar o uso do sistema.

Na tabela III temos a coluna "Alusão ao espelho", nela indicamos se a interface proposta faz uso ou não da metáfora do espelho para a interação. Para o critério "Social", indicamos os trabalhos apresentam recursos sociais como compartilhamento de fotos, sugestão e avaliação de maquiagens utilizadas por outros usuários. "Sugestão de produtos", significa que o modelo de interação proposto continha a sugestão de maquiagem para o usuário. Quanto à coluna "Custo", esta especifica o custo do projeto de acordo com os equipamentos utilizados, sendo que foi classificado como baixo custo os projetos que utilizaram apenas um computador e uma câmera simples (até dois mil reais), médio os que utilizaram outros sensores de custo não elevado, além do computador e câmera (até três mil reais), e alto para os que utilizaram equipamentos de custo elevado (maior que três mil reais). Por fim, a coluna Deficientes

TABELA I
COMPARATIVO ENTRE OS MÉTODOS PARA SIMULAÇÃO DE MAQUIAGEM.

Trabalho	Tipo	Flexível	Baixo custo	Realista	Tempo real
[2]	Transferência	Não	Sim	Possivelmente	Não especificado
[4]	Transferência	Não	Sim	Possivelmente	Não especificado
[3]	Transferência	Não	Sim	Possivelmente	Não especificado
[5]	Simulação	Sim	Sim	Não	Sim
[6]	Simulação	Sim	Sim	Não	Sim
[7]	Simulação	Sim	Não	Possivelmente	Sim

TABELA II
PRIMEIRO COMPARATIVO ENTRE OS TRABALHOS SOBRE INTERAÇÃO NO CONTEXTO DE SIMULAÇÃO DE MAQUIAGEM. N.E. SIGNIFICA NÃO ESPECIFICADO NO ARTIGO.

Trabalho	Simula maquiagem	Aplicação	Apresentação	Marcadores	Pronto para uso
[6]	Sim	Sim	Modelo 3D	Não	Não
[8]	Não	-	Imagem	Sim	Sim
[9]	Sim	N.E.	Modelo 3D	Sim	Não
[10]	Não	-	Imagem	-	Sim
[11]	Não	-	Imagem	-	Sim

TABELA III
SEGUNDO COMPARATIVO ENTRE OS TRABALHOS SOBRE INTERAÇÃO NO CONTEXTO DE SIMULAÇÃO DE MAQUIAGEM.

Trabalho	Alusão ao espelho	Social	Sugestão de produtos	Custo	Deficientes
[6]	Não	Não	Não	Alto	Não
[8]	Sim	Sim	Não	Médio	Não
[9]	Sim	Não	Sim	Médio	Não
[10]	Não	Não	Não	Baixo	Sim
[11]	Não	Sim	Sim	Médio	Não

esclarece se o projeto possui alguma abordagem que auxilie portadores de deficiência.

O modelo de interação proposto neste trabalho contempla uma interface que faz alusão a um espelho para aplicar independentemente os produtos de maquiagem desejados em qualquer região da face tocada no espelho. Por estar maquiando a imagem do reflexo do espelho, o sistema deve realizar a simulação em um vídeo da imagem do usuário e deve manter o correto posicionamento da maquiagem conforme o usuário movimenta a face em frente ao espelho sem o uso de marcadores. Ao analisar a tabela, logo pode-se perceber que existe apenas um trabalho que faz alusão ao espelho e também simula maquiagem, [9]. Nesse trabalho o reflexo do espelho é um modelo 3D do usuário que mimetiza seus movimentos e a maquiagem é simulada nesse modelo. Além disso, existe a necessidade de marcadores para o adequado funcionamento do sistema. Ou seja, o usuário precisa passar por algumas etapas preparatórias antes de iniciar o uso do sistema como criar seu modelo 3D, que não será tão realista quanto a imagem da câmera, e posicionar marcadores.

O outro trabalho que realiza simulação de maquiagem é o artigo [6] que se propõe a criar um sistema interativo para aplicar maquiagem em um modelo 3D com *feedback* háptico e visão estéreo e portanto não faz alusão ao espelho. O dispositivo háptico utilizado serve para aplicar a maquiagem no modelo 3D do usuário e além dele outros equipamentos são utilizados encarecendo os custos como escâneres 3D e monitor 3D.

Entre os outros artigos, o trabalho de [8] apesar de fazer alusão ao espelho, não realiza a simulação de maquiagem. O trabalho apresenta uma espécie de espelho com funcionalidades a mais que um espelho comum e utiliza marcadores para os objetos utilizados. O trabalho [11] tem um enfoque mais social e de recomendação de produtos. Esse artigo não usa a metáfora do espelho, já que faz uso de espelhos reais para que o usuário se maqueie de forma real e use o monitor apenas para se fotografar, compartilhar suas imagens e ver indicações de produtos. Por fim, o artigo [10] ensina um deficiente visual a aplicar a maquiagem e depois confere se ela foi aplicada adequadamente. Apesar de ser um trabalho bastante interessante, ele não se encaixa tão bem nos critérios, já que tem um enfoque diferente do tema desse trabalho.

III. SISTEMA

Neste trabalho, o termo maquiagem será utilizado para representar os produtos de maquiagem e o verbo maquiar representará a aplicação desses produtos.

Para modelar o problema de simulação de maquiagem, seja I a imagem de entrada sem maquiagem contendo a face F, R a imagem resultante após a simulação de maquiagem, M o conjunto de maquiagens a serem aplicadas e m cada uma das maquiagens pertencentes a M. Cada m corresponde à representação computacional da maquiagem real utilizada para a simulação, um *maquilet*. Além disso, seja m_α a intensidade padrão de cada *maquilet* a ser aplicado e m_{asc} a imagem correspondente à máscara que define o local de aplicação de m e suas variações de intensidade. Tanto I quanto R podem ser descritos adicionando $_{RGB}$ para designar que estão no espaço de cor RGB (I_{RGB}, R_{RGB}) ou $_{La^*b^*}$ para designar que estão no espaço de cor La*b* ($I_{La^*b^*}$, $R_{La^*b^*}$).

Assim, podemos definir a simulação de maquiagem como o processo que compõem a imagem R com a face F maquiada com M a partir de I, $I \xrightarrow{M} R$.

São três os tipos de maquiagens consideradas: base, sombra e batom. Cada uma possui suas características próprias e consequentemente alteram a pele de forma particular. A base é utilizada para uniformizar o tom de pele do rosto, corrigir alterações de cores e melhorar o relevo da pele disfarçando pequenas imperfeições. Para suavizar pintas, acne, olheiras ou manchas no rosto o procedimento correto seria utilizar um corretivo além da base, então não se deve esperar que toda base seja capaz de esconder todo e qualquer detalhe ou alteração do tom de pele. Existem diversos tipos de base, como *spray*, líquida, cremosa e bastão, porém não vamos nos ater às especificidades de cada um desses tipos e sim ao comportamento geral da base. Assim, este trabalho considera como base, um produto que possui como característica suavizar os detalhes faciais e homogeneizar os tons de pele de quem a utiliza, porém sem que ocorra um cobertura total da pele.

A sombra é utilizada para colorir as pálpebras. Também existem tipos diferentes de sombra, como em pó, cremosa e líquida, cada uma com suas próprias características. Porém, da mesma forma que foi feito para a base, este trabalho considera a característica geral da sombra e assim sua simulação se restringirá a introduzir cor na região aplicada.

O batom, além de colorir os lábios, pode ter função hidratante e até protetora contra o sol. Também pode-se encontrar diversos tipos de batons, alguns mais cremosos, outros mais brilhantes, mais opacos, criando um aspecto plastificado ou com maior cobertura dos detalhes do lábio. Da mesma forma que as maquiagens anteriores, vamos nos ater ao comportamento mais comum do batom e assim esse trabalho considera apenas a introdução de cor nos lábios.

A. Simulação de maquiagem

O método proposto para a simulação de maquiagem é capaz de simular o resultado da aplicação de maquiagem na pele da face em uma imagem do rosto do usuário, simulando a alteração das propriedades óticas e de textura da pele causadas pela maquiagem, o que modifica a forma com que a face é percebida.

Como ideia geral, o processo de simulação funciona decompondo a imagem da face em diversas camadas que são tratadas independentemente para simular o efeito desejado da maquiagem real. Cada camada, ou grupo de camadas, separa cada característica a ser simulada e é responsável por tratar essa questão específica.

Usando essa abordagem, pode-se isolar cada questão, como textura da pele, cor da pele, especularidade, pelos faciais e iluminação não uniforme, tratando-as de forma independente sem que uma interfira na outra. Além disso, o uso dessa abordagem facilita futuras melhorias e expansões do método, já que basta criar novas camadas, e também permite um melhor controle do equilíbrio entre qualidade da simulação e custo computacional do método, permitindo utilizar ou não utilizar determinadas camadas de acordo com a capacidade de processamento do dispositivo.

Para esse projeto, a simulação de maquiagem se concentrará em alterar a textura da pele e mudar sua cor. Assim, o módulo de simulação foi dividido em dois submódulos, Textura e Cor. O primeiro é responsável por tornar o tom de pele mais uniforme e suavizar ou realçar os detalhes da pele e o segundo pela introdução da cor da maquiagem. Dadas as características adotadas para cada tipo de maquiagem a ser simulada não será necessário realçar detalhes faciais, apenas suavizá-los, assim, não será apresentado neste trabalho o tratamento correspondente ao realce de textura.

O primeiro módulo do processo de simulação, chamado de Textura, é responsável pela suavização dos detalhes faciais e a homogeneização do tom de pele, características da aplicação de base. A partir de I_{RGB} tal imagem é dividida em cinco camadas distintas, cada uma representando uma faixa diferente de frequência, sendo que o número de camadas foi obtido de forma empírica. A imagem da camada com os detalhes faciais mais sutis, maior frequência, C_{fA}, não deve ser alterada já que esses detalhes caracterizam a face do indivíduo e também são preservados pela maquiagem real. O mesmo acontece para a imagem que representa a faixa de baixa frequência, C_{fB}, ou seja, ela também não é alterada. Assim, restam três outras faixas de frequências em que a simulação atua. Todas as suavizações seguem o referencial de intensidade estabelecido por m_{asc}, porém existe uma diferenciação entre a intensidade da suavização aplicada para cada camada. A maior suavização ocorre para a camada de frequência intermediária 3, C_{f3}, principal responsável pela homogeneização do tom de pele. A camada de frequência intermediária 2, C_{f2}, recebe uma suavização um pouco menor e a camada de frequência intermediária 1, C_{f1}, uma suavização menor do que a anterior. Com isso, consegue-se mimetizar o efeito da maquiagem suavizando os detalhes e tons de pele sem que isso resulte em uma aparência artificial. O diagrama de blocos desse módulo pode ser observado na figura 2.

Para dividir I_{RGB} em cinco faixas de frequência, é utilizado o filtro gaussiano que atua como um filtro passa baixa. Ao aplicar o filtro gaussiano em I_{RGB}, tem-se uma nova imagem $G1$. Ao subtrair $G1$ de I_{RGB}, teremos a imagem com a primeira faixa de frequência, C_{fA}, que conterá a faixa das altas frequências do espectro dessa imagem. Continuando o processo, aplicamos novamente o filtro gaussiano na imagem $G1$ criando $G2$. Ao subtrair $G2$ de $G1$, temos uma imagem correspondente a uma outra faixa de frequências intermediárias dessa imagem, C_{f1}. Repetindo o procedimento, obtém-se C_{f2}, C_{f3} e C_{fB}.

Já o segundo módulo, chamado de Cor, é responsável pela simulação da cor da maquiagem. Para isso, I é transformada para o espaço de cor La*b*, $I_{La^*b^*}$. Neste espaço, cada um dos canais torna-se uma camada e é tratada de forma independente. O canal L, C_L, permanece como o original, já que o objetivo é alterar apenas as cores da imagem preservando sua iluminação. Para o canal a*, C_{a^*}, é calculada uma média ponderada para cada pixel com o mesmo canal de uma imagem com a cor do *maquilet* correspondente. De forma análoga, o mesmo procedimento é realizado para o canal b*, C_{b^*}. Por fim, pode-se converter a imagem novamente para o espaço de cor RGB, I_{RGB}, caso seja esse o espaço utilizado na interface. O diagrama de blocos desse módulo pode ser observado na figura 3.

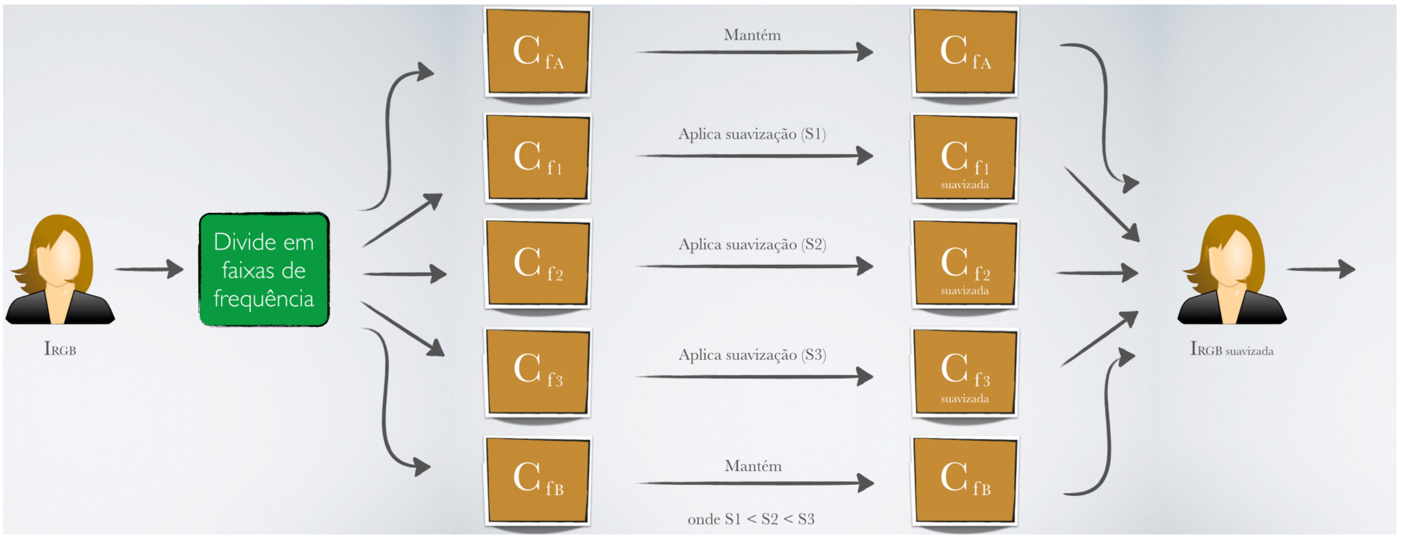
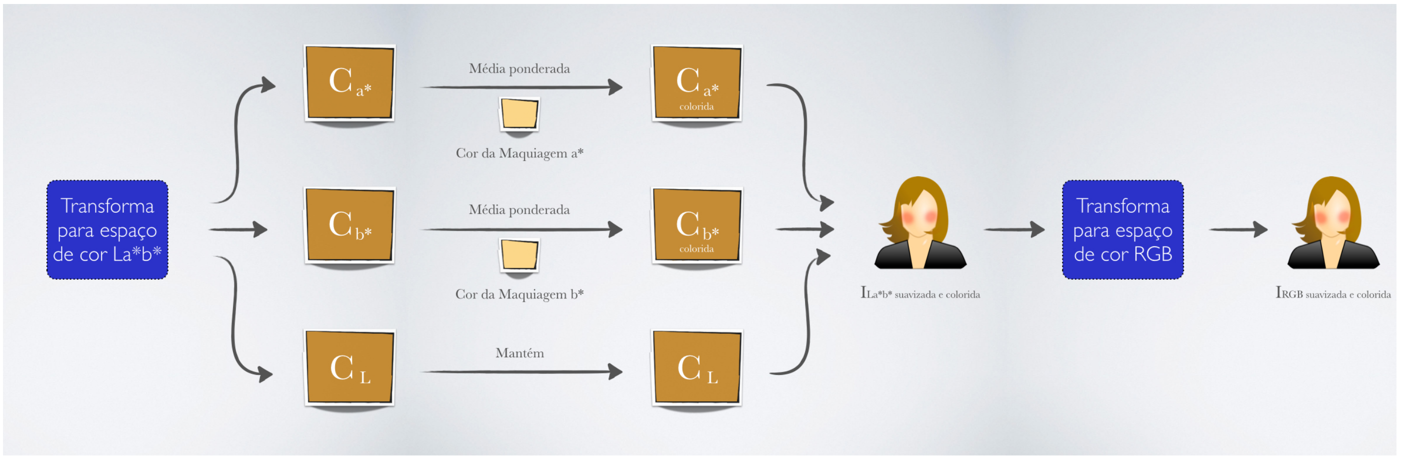


Figura 2. Fluxo da simulação de maquiagem - Textura.



Transforma para espaço de cor La*b*
C a*
C b*
C L
Média ponderada
Cor da Maquiagem a*
Média ponderada
Cor da Maquiagem b*
Mantém
C a* colorida
C b* colorida
C L
ILa*b* suavizada e colorida
Transforma para espaço de cor RGB
IRGB suavizada e colorida

Figura 3. Fluxo da simulação de maquiagem - Cor.

A média ponderada em questão é calculada de acordo com a equação 1 onde r_p é o valor resultante para cada pixel da imagem que representa a aplicação de m, I_p o valor da cor inicial para cada pixel da imagem de entrada I e m_c o valor de cor do *maquilet* que está sendo aplicado.

$$r_p = (1 - \alpha_p) * I_p + \alpha_p * m_c \tag{1}$$

O valor de α_p não é constante para todos os pixels, pois deve acomodar o efeito da variação da intensidade da maquiagem aplicada no local, possibilitando intensificar a aplicação do produto em determinadas regiões do rosto em que a maquiagem for aplicada mais de uma vez ou então suavizar outras regiões criando degradês. O valor de α_p é calculado seguindo a equação 2

$$\alpha_p = m_\alpha * (m_{asc_p}/vmp) \tag{2}$$

sendo m_{asc_p} o valor da intensidade do pixel da máscara do *maquilet* que é criada de acordo com o tempo e número de vezes que o usuário toca a tela do monitor sensível ao toque. Como os valores possíveis para cada pixel de m_{asc} são representados por um *byte* e o valor de m_α está compreendido entre 0 e 1, o valor do pixel da máscara foi dividido pela constante vmp (valor máximo do pixel) que vale 255 para que o valor de α_p também esteja entre 0 e 1.

Com relação à cor de m, ela pode ser escolhida livremente pelo usuário para que ele crie a maquiagem que deseja ou então ele pode escolher valores pré-determinados que simulam maquiagens reais.

Com esses dois módulos, o algoritmo criado é capaz de simular a aplicação de base, sombra e batom. A simulação para o *maquilet* de base utiliza todo o algoritmo, Textura e Cor, já os *maquilets* de sombra e batom necessitam apenas da etapa Cor, pois seu comportamento na pele afeta predominantemente as cores.

B. *Interação*

A interação para o simulador de maquiagem foi pensada de tal forma que ela fosse semelhante à interação que uma pessoa tem ao se maquiar utilizando um espelho. Assim, o intuito é que o usuário seja capaz de se maquiar livremente, escolhendo os tipos de maquiagens, suas cores e intensidades para combiná-las e aplicá-las em qualquer lugar de sua face. Além da liberdade de aplicação, o aparato a ser usado deve fazer a alusão ao espelho e utilizar os equipamentos disponíveis no laboratório. Desta forma, a interface a ser desenvolvida é constituída por um computador com monitor sensível a toque e um sensor RGBD (Kinect).

Acima do monitor, posiciona-se o sensor RBGD, responsável por prover as imagens e outros dados para a simulação. O computador com monitor sensível a toque fará o papel de espelho e também exibirá a interface para a escolha da maquiagem e suas características. Para isso, a interface foi dividida em duas regiões, a região de espelho e a região de controles como pode ser observado na figura 4. Utilizando o toque na tela, pode-se escolher o produto e outras características para a aplicação da maquiagem que é realizada tocando o monitor na região do rosto desejada para sua aplicação. Caso o usuário permaneça com seu dedo em um mesmo local da face ou toque essa região múltiplas vezes, a maquiagem será intensificada proporcionalmente ao tempo ou ao número de vezes em que o local foi tocado. Para facilitar a aplicação, a região dos controles permite que o usuário possa congelar a imagem do espelho para aplicar os cosméticos. Desta forma, não existe a necessidade de manter a face imóvel enquanto a maquiagem é aplicada. Além disso, também existe a possibilidade de salvar a imagem da simulação de maquiagem, alterar entre modos de exibição (por exemplo comparar antes e depois da simulação) e definir o tipo de traçado a ser simulado por meio da escolha do produto de maquiagem, seu modo de aplicação (aplicar, suavizar, remover), instrumento, quantidade e pressão com que o produto de maquiagem será aplicado na simulação.

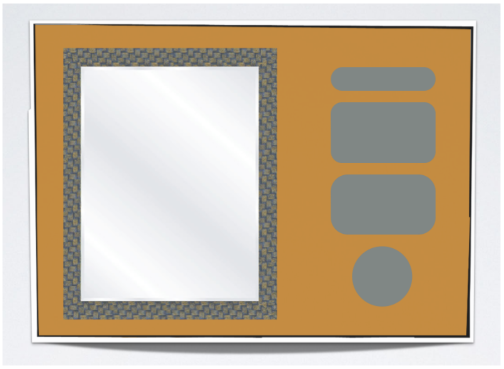


Figura 4. À esquerda a região do espelho e à direita a região dos controles.

IV. Prototipação

Para que fosse possível obter maior flexibilidade na escolha das técnicas utilizadas e consequentemente possibilitar a percepção de todos os detalhes da aplicação da maquiagem na pele, o ideal seria utilizar computadores de altíssimo desempenho, um sensor RGBD com grande resolução de imagem e qualidade óptica e também implementações específicas para o propósito do projeto para que se pudesse explorar todo o poder computacional e toda a qualidade de imagem disponível.

Entretanto, os recursos de tempo e de *hardware* não são ilimitados. Assim, utilizou-se como sensor RGBD o Microsoft

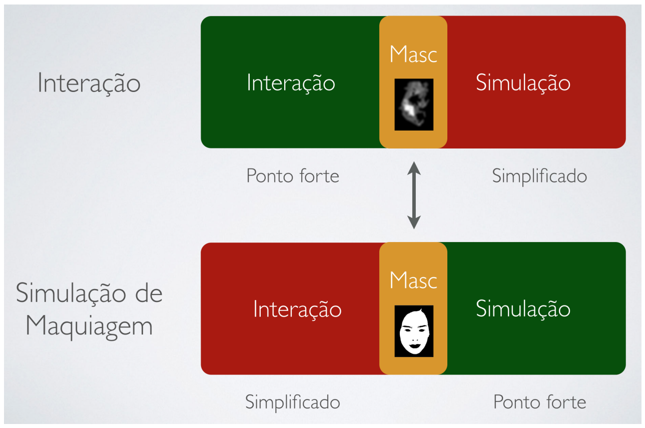


Figura 5. Pontos fortes e fracos dos protótipos da Interação e da Simulação de Maquiagem, bem como seu elo de ligação para junção dos dois protótipos em um só, quando a capacidade computacional do *hardware* permitir.

Kinect e como monitor sensível a toque um dos computadores do laboratório que possui tal funcionalidade.

Para contornar as limitações de recursos, foram criados dois protótipos: um focado no algoritmo para a simulação de maquiagem, Simulação de Maquiagem, e outro focado na interação, Interação. Portanto, a Interação utilizará um procedimento bastante simples para simular a maquiagem com o objetivo de poupar processamento e a Simulação de Maquiagem ficará restrita apenas ao método para simulação da maquiagem, também poupando processamento por não conter outros procedimentos utilizados para a interação. Nos dois casos, os protótipos foram implementados utilizando a linguagem de programação C++, bem como a biblioteca OpenCV [12] que contém diversas funções básicas de processamento de imagens e visão computacional.

Apesar de independentes, os dois protótipos são facilmente combináveis já que o sistema foi pensando como um todo e apenas separado durante a prototipação para explorar melhor os recursos disponíveis e assim demonstrar a viabilidade do projeto. O elo de ligação entre os dois protótipos são os modelos (face e *maquilets*), representados na figura 5 pela m_{asc}, necessitando apenas da evolução do poder computacional do *hardware* para que se possa unir os dois utilizando o ponto forte de cada um deles.

A. Interação

Para a prototipação do projeto, dada a separação da interação do algoritmo de simulação de maquiagem, foi necessário simplificar o módulo Simulação apresentado na figura 1 para a Interação. A simplificação consistiu em tornar o método de simulação de maquiagem computacionalmente mais econômico o que também reduziu o realismo da simulação, mas garantiu que o vídeo usado como imagem do espelho fosse fluido e permitisse a utilização apropriada da interface.

Para possibilitar a interação, foi desenvolvido um *software* específico para essa parte do projeto que deve ser capaz de:

- gerar um vídeo em tempo real para a imagem a ser usada como espelho exibindo os efeitos da maquiagem;

- simular a maquiagem de forma simplificada para economizar processamento;
- encontrar os mesmos pontos faciais em cada quadro do vídeo da imagem do rosto do usuário de forma automatizada e sem o uso de marcadores;
- gerar a triangularização baseado nos pontos faciais encontradas para a face;
- aplicar cada tipo de maquiagem independentemente dos demais permitindo sua composição interativamente.

Além disso, tem-se as seguintes limitações:

- simulação de sombra, batom e base;
- condições controladas de iluminação para que ela seja uniforme;
- usuário não usará óculos ou algum outro acessório que possa cobrir toda ou parte de sua face;
- usuário não possuirá pelos faciais cobrindo características relevantes da face;
- Kinect possui uma limitação de funcionamento em que o usuário deve ficar a pelo menos 80cm do dispositivo para que ele funcione adequadamente.

Para criar o *software* da Interação foi utilizada a versão 1.8 do SDK da Microsoft para o Kinect [13]. O sistema utiliza 82ms para cada quadro do vídeo a ser computado o que resulta em 12 quadros por segundo para o vídeo da interface. Esses resultados foram obtidos por meio de testes realizados em um *notebook* com processador Core i7 2,6GHz utilizando a imagem do próprio Kinect que possui 640x480 pixels. Para a Interação, não se fez uso de processamento paralelo para diminuir o tempo de computação do sistema.

1) Triangularização e Modelo de face: Como pode ser observado na figura 1 o Kinect é o sensor RGBD que fornece a imagem e os dados de rastreamento da face (pontos faciais) que são obtidos utilizando o SDK do Kinect. Com os pontos faciais e mais algumas funções do mesmo SDK, é possível obter uma triangularização inicial da face.

A triangulação obtida usando o SDK do Kinect fornece 206 triângulos espalhados por toda a face como pode ser observado nas figuras 6a e 6b. Como o objetivo com a triangularização é obter uma maneira para manter o posicionamento da maquiagem aplicada na face do usuário durante a movimentação de seu rosto, quanto maior for o número de triângulos gerados pela triangularização, mais preciso será o mapeamento da maquiagem na face do usuário. Nesse protótipo, o mapeamento foi realizado atribuindo para cada triângulo cada *maquilet* que foi aplicado nessa posição e que deve ser simulado nessa região da face ocupada pelo triângulo.

Para aumentar a precisão, os 206 triângulos foram subdivididos uma vez, sendo que, para cada subdivisão, a região era repartida em 6 novos triângulos totalizando 1236 triângulos ao final do procedimento, que podem ser observados na figura 6c e 6d. A subdivisão foi realizada da seguinte forma, dado um triângulo T, com vértices A, B e C, pontos médios X (aresta $\overline{AB}$), Y (aresta $\overline{AC}$) e Z (aresta $\overline{BC}$) e com baricentro G os triângulos formados são AGX, GXB, BGZ, ZGC, CGY e YGA. A figura 7 ilustra um exemplo de subdivisão.

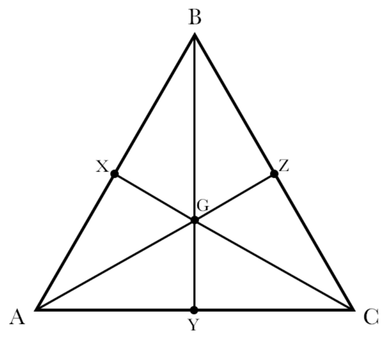


Figura 7. Um exemplo de subdivisão de um triângulo ABC.

A partir das informações sobre os pontos faciais, sua triangularização e locais de aplicação da maquiagem é construído o Modelo de face usado pela simulação de maquiagem.

2) Simulação de maquiagem: No módulo Simulação, é realizada a simulação da aplicação de maquiagem de forma simplificada utilizando os dados do Modelo de face e dos *Maquilets* a cada quadro do vídeo provido pelo Kinect.

Para a simulação da maquiagem deve-se criar para cada *maquilet* a ser simulado sua m_{asc}, a máscara responsável por definir o local de aplicação de m e suas variações de intensidade (figura 8a). A máscara possui as mesmas dimensões da imagem RGB do Kinect, 640x480 pixels, e define o local e a intensidade da maquiagem de acordo com o valor do pixel nesse local. Pixel com valor zero representa ausência de maquiagem e pixel com valor maior do que zero representa o oposto, sendo que o valor do pixel retrata a intensidade de m nessa posição da imagem. Então, para cada triângulo que possui um *maquilet* associado, a m_{asc} correspondente é atualizada para que seja maquiada a região delimitada pelo triângulo com a intensidade descrita pelo *maquilet*.

Caso o usuário permaneça com o dedo em um mesmo local da face ou toque mais de uma vez em um mesmo triângulo com o mesmo *maquilet*, a cor correspondente ao m dessa região será intensificada gradualmente, de acordo com o número de vezes em que a maquiagem for aplicada nesse local. Para o caso de *maquilets* diferentes em um mesmo triângulo, as cores serão misturadas, já que a aplicação da maquiagem é realizada em sequência uma para cada *maquilet*.

Para evitar possíveis imprecisões com relação a qual pixel corresponde a qual triângulo nas regiões de borda entre eles, foi aplicada a operação morfológica de dilatação em m_{asc}. Além disso, para evitar regiões de transição brusca entre as áreas que recebem maquiagem (os triângulos) e as que não recebem, foi aplicada uma suavização gaussiana em m_{asc}. Um exemplo do resultado desse procedimento pode ser visto na figura 8b.

A aplicação da maquiagem foi modelada como uma média ponderada, no espaço de cor RGB, entre o valor do pixel da imagem da face (apenas para os pixels delimitados por m_{asc}) e o valor de cor da maquiagem desejada. Por fim, a imagem resultante desse processo é exibida pela interface. Um exemplo do resultado pode ser observado em 8c.

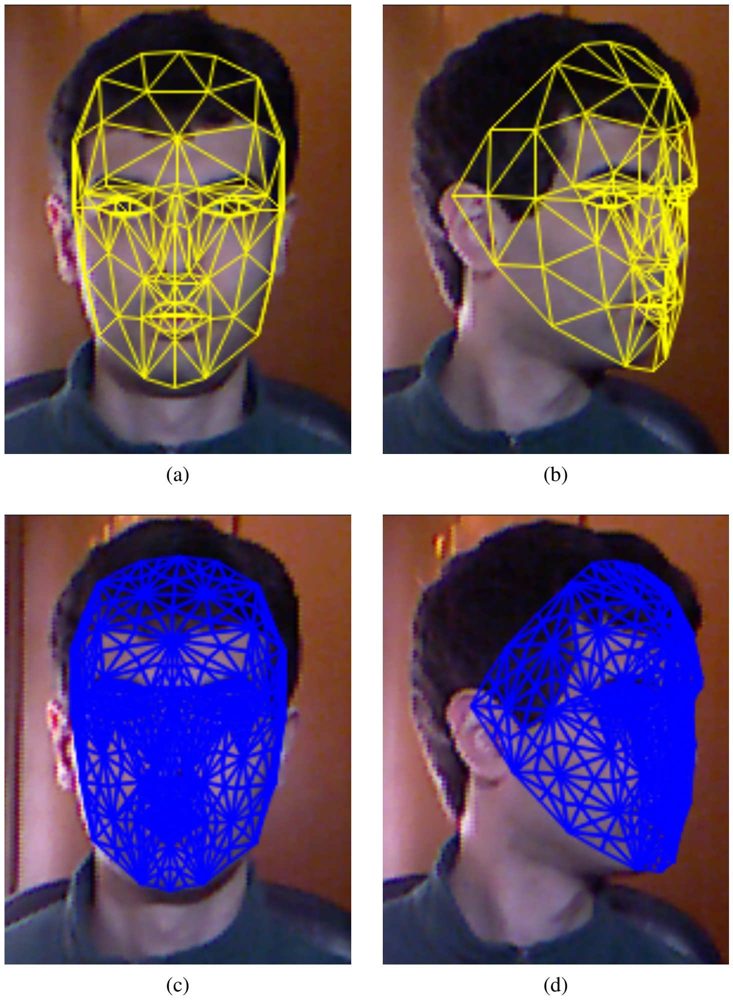


Figura 6. (a) triangularização obtida apenas com as funções do SDK do Kinect, vista frontal; (b) triangularização obtida apenas com as funções do SDK do Kinect, vista lateral; (c) triangularização final, vista frontal; (d) triangularização final, vista lateral.

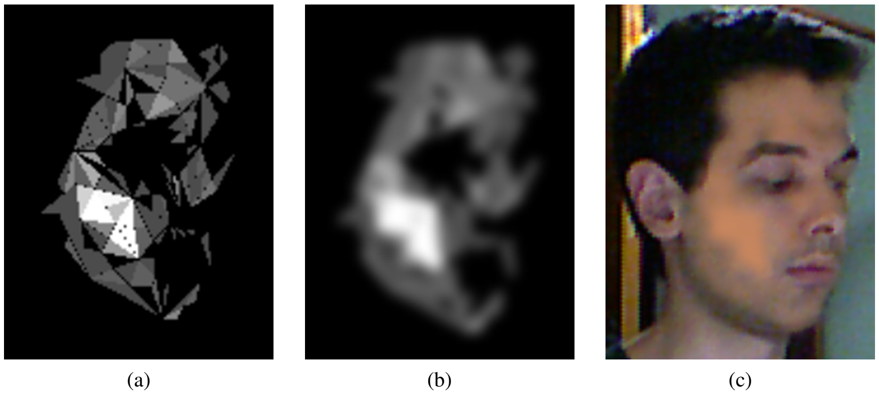


Figura 8. (a) Exemplo de máscara para o protótipo da interação criada a partir do toque do usuário na imagem da face; (b) Mesma máscara de (a) após operação de dilatação e suavização; (c) Resultado da aplicação de maquiagem para o protótipo da interação considerando a m_{asc} apresentada em (b).

3) Interface e maquilets*:* A interface do protótipo desenvolvido foi baseada em um monitor *widescreen* e portanto tem proporção de 16:9. Das 16 partes para a largura, 10 foram utilizadas para a região da imagem da face do usuário em alusão ao espelho na parte esquerda do monitor, região do espelho, e as 6 partes restantes foram alocadas para as opções necessárias para a escolha das maquiagens desejadas e suas características, região de controles. Devido à pequena resolução da imagem RGB do Kinect, foi necessário aumentá-la em 5 vezes para que a face cobrisse toda a região dedicada

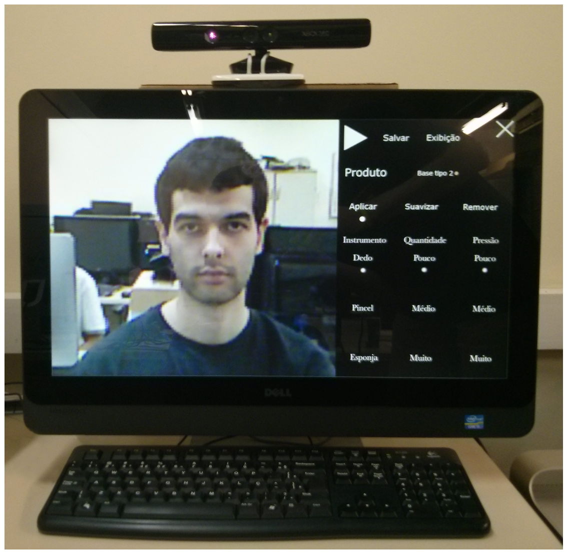


Figura 9. Disposição dos equipamentos do protótipo da interação exibindo também a região de espelho e a região de controles da interface. O teclado não é necessário para a utilização do sistema, é usado apenas durante o uso comum do computador.

a ela.

A interface apresenta as opções para todo tipo de controle necessário para simular a maneira com que a maquiagem escolhida é aplicada ou removida da pele. Para a prototipação, os esforços foram concentrados nas funcionalidades básicas, como a escolha da maquiagem e sua aplicação. A partir das escolhas relacionadas aos produtos de maquiagem que o usuário faz na interface, são criados os *maquilets* correspondentes a cada um desses produtos. Eles contém informações do tipo de produto, cor e intensidade da maquiagem que eles representam.

Pode-se ver a disposição dos equipamentos para a interação na figura 9 e também a região de espelho e de controles da interface.

B. Simulação de maquiagem

Considerando esse módulo de forma independente do resto do sistema, algumas adaptações foram necessárias. Na imagem da arquitetura geral, 1, o módulo Simulação recebia um vídeo e fazia uso do Modelo de Face e de *Maquilets*. Agora, no lugar do vídeo temos uma imagem da face do usuário, o Modelo de Face não é utilizado já que o posicionamento da maquiagem é fixo e associado à m_{asc}, e continuamos a utilizar os *maquilets*.

Com relação à máscara m_{asc}, ela é específica para cada imagem em que se simulará a maquiagem, já que ela depende do posicionamento e morfologia da face. A máscara pode ser criada manualmente por meio da interface do protótipo desenvolvido para essa parte do sistema, permitindo que o usuário aplique a maquiagem nas regiões da face que ele desejar. Além desta forma, a máscara também pode ser criada em um programa de edição de imagens e depois utilizada no *software* desenvolvido. Isto seria útil para criar modelos padrões de maquiagem para que o usuário não tenha o trabalho de definir a região de aplicação de m realizando apenas retoques dessa máscara dentro do protótipo. A figura 10 apresenta alguns exemplos de máscaras utilizadas para o algoritmo.

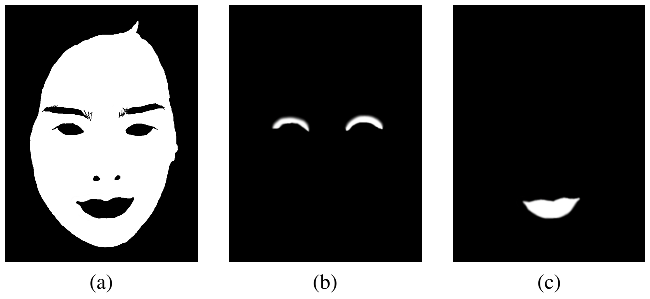


Figura 10. Exemplos de máscaras, m_{asc}, para a aplicação de cada produto de maquiagem no protótipo da simulação de maquiagem. (a) exemplo de máscara para a aplicação de base; (b) exemplo de máscara para a aplicação de sombra; (c) exemplo de máscara para a aplicação de batom.

Para que o algoritmo de simulação de maquiagem seja bem sucedido, ele deve ser capaz de simular o resultado da aplicação de maquiagem na pele da face em uma imagem do rosto do usuário. Isso simulará a alteração das propriedades óticas e de textura da pele causadas pela maquiagem. As limitações existentes são as mesmas da interação, exceto pela limitação referente ao uso do Kinect, já que ele não foi utilizado para o protótipo da simulação de maquiagem.

1) Simulação de maquiagem - Textura: Como explicado anteriormente, o módulo de Textura é responsável pela suavização dos detalhes faciais e a homogeneização do tom de pele. Para solucionar essa etapa da simulação, a imagem original foi dividida em algumas faixas de frequências que foram suavizadas.

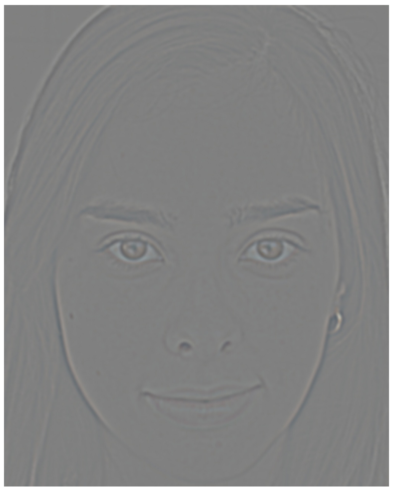
Para as suavizações, tanto as responsáveis por dividir a imagem da face em faixas de frequências quanto as responsáveis pelo efeito da maquiagem, foi utilizado, no lugar do filtro gaussiano, o *box filter*. Essa substituição se deu, pois o resultado dos dois filtros são semelhantes [14], porém o tempo de computação da suavização com o filtro utilizado é menor. Isso contribuiu para que fosse possível realizar a simulação mais rapidamente sem compromisso da qualidade do resultado da simulação.

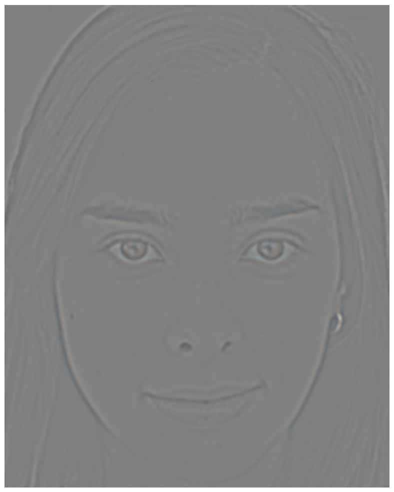
Neste módulo da simulação de maquiagem, separamos a imagem inicial em 5 camadas. Cada camada é representada por uma imagem de uma faixa diferente de frequência da imagem inicial. Devido a essa diferença de faixas de frequência, temos as camadas C_{fA}, C_{f1}, C_{f2} e C_{f3} contendo detalhes faciais dos mais sutis aos menos sutis e uma camada final, C_{fB}, que também pode ser chamada de residual, já que ela contém todo o restante da informação. Dessa forma, estamos separando detalhes faciais do resto da imagem da face para que seja possível trabalhar neles sem alterar outras caraterísticas da imagem. Para voltar a formar a imagem completa, deve-se somar cada uma das camadas.

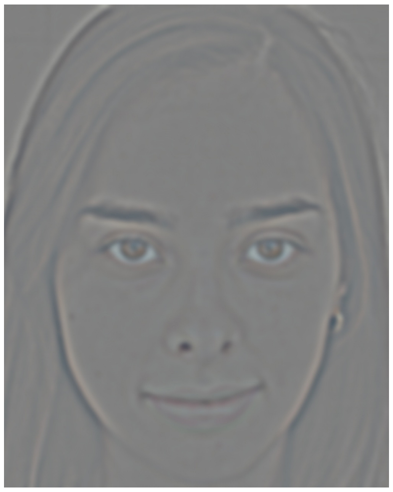
A figura 11 apresenta o resultado do procedimento de divisão da imagem em faixas de frequência. Em 11h e 11g tem-se as duas camadas que não são alteradas, C_{fB}, baixa frequência, e C_{fA}, alta frequência, respectivamente. As outras três camadas, 11a, 11c e 11e são suavizadas na simulação para atingir os objetivos citados acima para essa etapa.

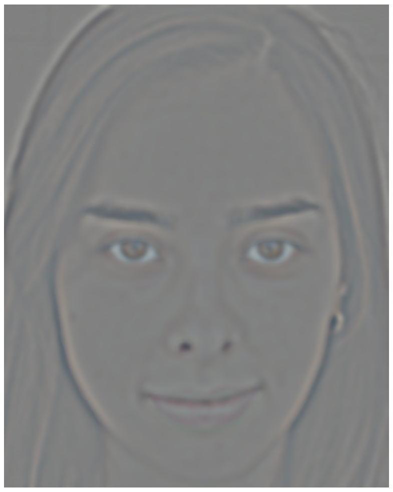
(a)

(b)

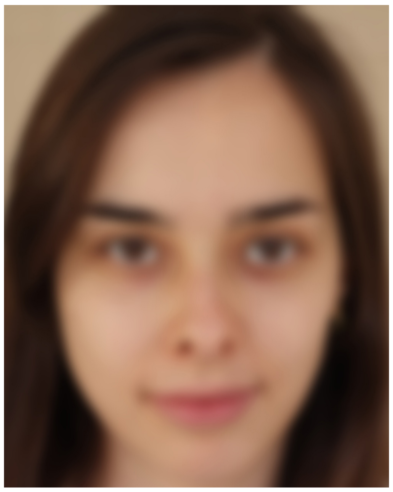

(c)


(d)


(e)


(f)

(g)


(h)

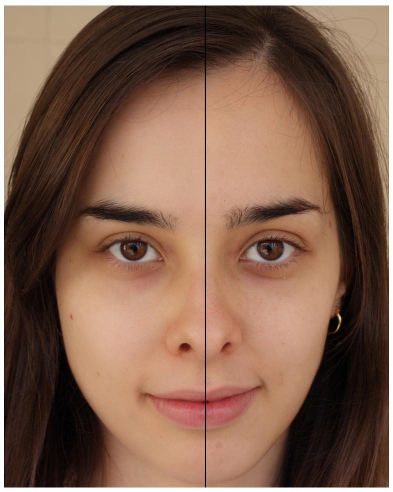

(i)

Figura 11. (a) camada de frequência intermediária 1, C_{f1}; (b) camada de frequência intermediária 1, C_{f1}, após suavização; (c) camada de frequência intermediária 2, C_{f2}; (d) camada de frequência intermediária 2, C_{f2}, após suavização; (e) camada de frequência intermediária 3, C_{f3}; (f) camada de frequência intermediária 3, C_{f3}, após suavização; (g) camada de alta frequência, C_{fA}; (h) camada residual, C_{fB}; (i) esquerda: resultado da simulação levando em conta apenas o módulo Textura, direita: imagem original.

A camada residual, C_{fB}, representa o que resta após serem removidas quatro camadas de detalhes faciais da imagem. Assim, essa camada possui características globais da face da pessoa como a maior parte de sua cor e os traços básicos da face. Essa camada é deixada intacta já que os detalhes faciais a serem modificados não estão presentes e também não se obtém ganhos significativos de homogeneização da pele se ela fosse alterada.

A camada C_{f3} contém os detalhes faciais menos sutis. Nessa camada, detalhes como os poros faciais normalmente não são claramente visíveis, porém pode-se observar melhor as diferenças de tons de pele que são importantes para a percepção de uma pele mais homogênea. Assim, o tratamento dessa camada é o principal responsável pela homogeneização do tom de pele, para isso ela recebe a maior suavização das três camadas tratadas, já que estamos trabalhando com os detalhes mais grosseiros da face.

A camada C_{f2} contém os detalhes faciais de intensidade intermediária. Nela ainda é possível observar algumas diferenças mais sutis de tons de pele e começam a aparecer detalhes faciais menores. Nesta camada suavizamos com intensidade menor do que na camada anterior, já que as diferenças de tom de pele são mais sutis e alguns detalhes da textura da pele já estão presentes.

A camada C_{f1} contém detalhes faciais mais finos do que a anterior. A suavização a ser utilizada aqui deve ser sutil para não descaracterizarmos a face.

A camada C_{fA} apresenta os detalhes faciais mais sutis. Nela pode-se observar claramente pelos faciais, fios de cabelo, alguns contornos de regiões da face entre outras pequenas perturbações da pele. Esta camada é mantida intacta pois ela apresenta os detalhes faciais mais importantes para a caracterização da pele e consequentemente do indivíduo. Assim, se os detalhes forem modificados a pele perde seu aspecto natural e começa a ser percebida como manipulada artificialmente.

Os detalhes faciais que aparecem em cada uma dessas camadas dependem de cada indivíduo. Algumas pessoas possuem poros faciais que estarão mais visíveis na camada C_{f2} outras na C_{f1}, o que também vale para rugas e outros detalhes faciais. Para a simulação de maquiagem, o objetivo não é remover toda e qualquer ruga ou poro da face como é comum em fotografias de revistas. No contexto desse trabalho, estamos interessados em mimetizar o efeito da maquiagem, portanto, se aplicarmos uma base em uma face com, por exemplo, rugas muito intensas não se deve esperar que elas desapareçam, mas sim que sejam suavizadas. Dependendo de quão severa são as características faciais, elas estarão presentes em uma ou mais camadas e o efeito da maquiagem será aplicado se restringindo ao que a maquiagem real faria. O fato de decompormos em camadas com detalhes de intensidades diferentes e suavizá-las com intensidades diferentes é o que contribui para se aproximar do comportamento real da maquiagem.

A figura 11 também exibe os resultados parciais da simulação de maquiagem levando em conta apenas o primeiro módulo da simulação, ou seja, ainda sem a adição de cores. Em 11i tem-se o resultado do módulo Textura à esquerda, e à direita a imagem original, I. Em 11b, 11d e 11f as camadas C_{f1}, C_{f2} e C_{f3} após suavização.

Como pode ser observado, as imagens das faixas de frequência sofrem suavizações que abrandam detalhes faciais e ajudam a tornar mais uniforme as diferenças de tons de pele. Observando a comparação entre a imagem inicial e a final da face, nota-se que a pele está mais uniforme. Observando regiões específicas, pode-se perceber que o tom de pele está mais homogêneo principalmente na região da testa, queixo e a na região do nariz e abaixo dos olhos. Apesar dessas suavizações, ainda é possível observar pequenos detalhes faciais que caracterizam a pele e consequentemente a face da pessoa.

2) Simulação de maquiagem - Cor: O módulo Cor da simulação de maquiagem trabalha a mudança de cor que ocorre na pele. Para simularmos essa alteração, foram testados diferentes espaços de cor. A ideia para mimetizarmos o efeito de cor era alterar os canais responsáveis pela cor em cada um dos espaços de cor testados com a média ponderada definida na equação 1.

A figura 12 apresenta o resultado desse procedimento para RGB, HSL e La*b*, os espaços de cor testados. Nela é possível perceber que os espaços de cor RGB e HSL tiveram resultados semelhantes enquanto que o La*b* mostrou melhores resultados. Para o La*b*, pode-se notar que ele foi capaz de separar melhor a informação de cor da iluminação, o que resultou em um melhor tratamento para as regiões sombreadas, como próximo ao cabelo e olhos em detalhe na figura. Em RGB ou HSL, essas regiões se tornaram mais claras, inclusive sendo possível perceber o fim da região em que o algoritmo foi aplicado na transição da pele para o cabelo. Com isso, a face obteve um aspecto mais caricaturado, já que a iluminação natural não foi respeitada, perdendo então o realismo da simulação. Assim, o espaço de cor La*b* foi utilizado para esse segundo módulo da simulação.

Cada uma das camadas criadas nesse módulo pode ser vista na coluna da esquerda da figura 13, em que 13e é a camada C_L, 13a é a camada C_{a^*} e 13c é a camada C_{b^*}. Na coluna da esquerda, podemos notar a mesma camada após o procedimento da média ponderada, sendo 13b o resultado para a camada C_{a^*} e 13d o resultado para a camada C_{b^*}. Neste exemplo, a média ponderada foi realizada igualmente para toda a imagem, ou seja, sem respeitar m_{asc}. É possível perceber também que os canais correspondentes às cores não possuem a área sombreada na região da testa próxima ao cabelo, enquanto que isso se faz presente no canal L demonstrando a melhor separação entre iluminação e cor o que nos levou a melhores resultados ao utilizar esse espaço de cor.

3) Desempenho: Para paralelizar a execução do código, foi utilizado o OpenMP [15]. Os testes de desempenho para essa forma de paralelismo foram conduzidos em um *notebook* com processador Core i7 2,6GHz com 4 núcleos. O tempo de execução médio de todo o algoritmo para a simulação de maquiagem para uma imagem de 1018x990 pixels pode ser observado na Tabela IV.

Pode-se perceber que ocorreu um ganho significativo de desempenho com o paralelismo. O maior ganho se deu na etapa responsável pela divisão da imagem em faixas de frequências. Essa primeira etapa do módulo Textura precisava de 60ms para realizar seu processamento antes da paralelização e depois 22ms. Entretanto, nem todas as etapas do código

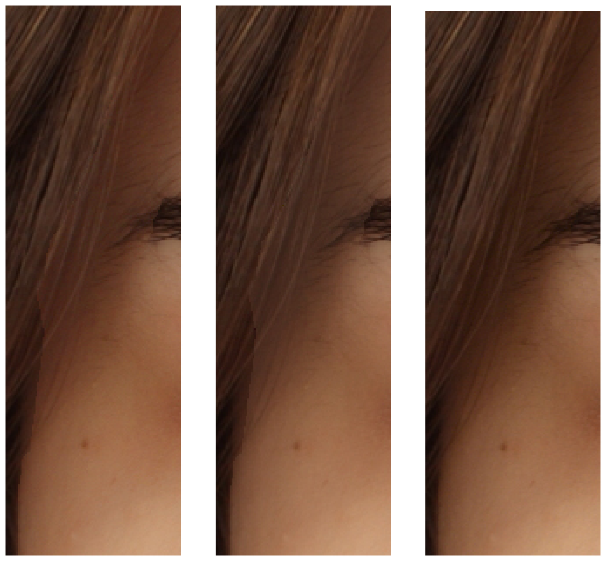


Figura 12. Resultado da simulação utilizando os espaços de cor HSL, RGB e La*b*. Para facilitar a visualização das diferenças entre os resultados dos três espaços de cor, foram recortados da imagem do resultado de cada uma delas a região entre o olho direito e o cabelo. (a) resultado para a simulação utilizando o espaço de cor HSL; (b) resultado para a simulação utilizando o espaço de cor RGB e (c) resultado para a simulação utilizando o espaço de cor La*b*.

TABELA IV
DESEMPENHO DO ALGORITMO DE SIMULAÇÃO DE MAQUIAGEM COM E SEM PARALELIZAÇÃO EM CPU PARA UMA IMAGEM DE 1018X990 PIXELS.

	Tempo (ms)	FPS
Sem paralelização	140	7
Com paralelização	83	12

foram beneficiadas já que algumas funções do OpenCV, como *cvtColor* e *bitwise_and* não obtiveram ganhos ao tentar usá-las de forma paralela. Provavelmente o OpenCV não permite que essas funções atuem de forma paralela ou a biblioteca já as implementou utilizando todo o potencial da CPU e portanto não são obtidos ganhos ao tentar paralelizá-las.

Vale ressaltar que para o contexto do módulo de Simulação de Maquiagem, estamos realizando a simulação em uma imagem do usuário e não em um vídeo. Logo, não temos a necessidade de realizar o procedimento de decomposição da imagem em faixas de frequências mais de uma vez, o que nos permite descontar do tempo total de execução o custo relativo a essa etapa que, como já foi dito, é de 22ms. Assim, temos 61ms (16FPS) para realizar a aplicação da maquiagem na imagem do usuário com 1018x990 pixels, que é um tamanho suficiente para exibir a imagem em um monitor *Full HD* descontando o espaço utilizado pelos cabeçalhos das janelas do *software*.

V. RESULTADOS

No vídeo em https://www.youtube.com/watch?v=c-Y6xUXsfQM pode-se observar o protótipo da interação. Na região dos controles, a primeira linha contém funcionalidades gerais do sistema como pausar a imagem do espelho, salvar a imagem do espelho, escolher modo de exibição e fechar o sistema. Abaixo, se encontra a primeira escolha que o usuário fará no simulador, o produto de maquiagem a ser aplicado. Como isso ocorre com frequência durante a interação, sua região é grande e pode ser tocada facilmente. Para realizar a escolha da maquiagem, basta tocar no produto desejado na nova região que é exibida no meio do monitor e dispõe das opções existentes de base, batom e sombra. Após ter escolhido a maquiagem, na próxima linha o usuário poderá escolher se deseja aplicar, suavizar ou remover o produto. Dependendo de sua opção, os controles adequados aparecerão abaixo para definir a maneira com que o cosmético é aplicado na face. Para a aplicação, é possível escolher entre o instrumento utilizado para aplicar a maquiagem (pincel, esponja e dedo), a quantidade de maquiagem a ser utilizada (pouca, média, muita) e a pressão com que o instrumento seria pressionado contra a pele da face (baixa, média, alta). Para a suavização, é possível escolher o instrumento a ser usado na aplicação (porém apenas entre dedo e pincel) e a pressão, enquanto que para a remoção não existem opções a serem escolhidas. A escolha do tipo de maquiagem também pode alterar as opções disponíveis, caso algum batom seja escolhido, o usuário poderá apenas aplicar ou remover (não terá a opção de suavizar) o produto e definir apenas a pressão para a sua aplicação, já que o instrumento a ser utilizado para a aplicação desse produto é o próprio batom e a quantidade de material não é variável. Para base e sombra todas as opções estão disponíveis.

Para aplicar a maquiagem, após escolher o produto e as características para sua aplicação, basta tocar com o dedo na região da face desejada. Então, o usuário poderá aplicar os produtos de maquiagem que ele escolheu em qualquer lugar da face, inclusive combinando esses produtos ou então realçando um mesmo produto, caso ele passe o dedo mais de uma vez na mesma região. Para que a face fosse grande o suficiente para permitir a interação por meio do toque para a aplicação da maquiagem, foi necessário aumentar em cinco vezes a imagem obtida pela câmera RGB do Kinect. Devido a esse fator e à qualidade da câmera e lente do dispositivo, a imagem utilizada como espelho não possui boa aparência, mas possibilita a interação com o *hardware* disponível no momento.

Além do realismo da simulação de maquiagem para a interação não ser bom, o que era esperado já que se trata de uma simplificação e utiliza uma imagem de baixa resolução, pode-se notar formatos geométricos e também é possível perceber certa imprecisão ao aplicar a maquiagem na face ao utilizar o protótipo. Ambos, apesar de serem efeitos indesejados, são consequência do número de triângulos utilizados. Da forma com que a simulação foi proposta, quanto maior o número de triângulos na face, melhor será a precisão da maquiagem e menos triangular será sua simulação. Ao utilizar um *hardware* de melhor desempenho, pode-se utilizar mais triângulos simultaneamente mantendo ou melhorando a taxa de quadros por segundo.

Quanto ao desempenho do rastreamento de face do Kinect, ele demonstrou algumas falhas em sua precisão, mas em termos gerais teve um funcionamento adequado para mostrar a viabilidade da solução.

Na figura 14, temos o resultado da simulação para batom, sombra e base. Para a simulação de base, pode-se observar a suavização de detalhes faciais e homogeneização do tom de pele. Os detalhes faciais e as diferenças de tom de pele foram suavizados tornando a pele mais uniforme, porém ainda é possível observar pequenos detalhes faciais que caracterizam a pele e consequentemente a face da pessoa. A adição da cor da base também ajuda a tornar a cor da pele mais homogênea se aproximando do efeito real da base. Além dos efeitos já citados sobre a base, é possível observar a adição de cor referente à sombra na região da pálpebra e do batom nos lábios. As cores da base, batom e sombra utilizados foram, respectivamente: RGB(209,153,120), RGB(177,3,4) e RGB(73,73,73).

Apesar de não definidas desta forma para esse trabalho, sombras ou batons que também possuam uma componente que altere a textura da pele, como suavizar ou realçar os detalhes faciais, também podem ser criadas utilizando a mesma arquitetura do método proposto.

Na figura 15, podemos comparar os resultados para a base obtidos com a aplicação real da maquiagem e sua simulação. A cor da base utilizada para a simulação foi RGB(144,112,89). Para obter esse tom de cor, que deve ser o mais próximo possível da maquiagem real para a comparação dos resultados, inicialmente verificou-se o valor de cor de alguns pixels do resultado da pele maquiada com maquiagem real, para então obter um valor inicial de cor para a simulação. Mais alguns justes de cor foram feitos até chegar no tom de cor final.

Na figura, é possível observar que o resultado da simulação utilizando o método proposto se aproxima da maquiagem real. Podemos observar também o resultado para o método sendo executado além de suas limitações, neste caso, iluminação não uniforme. Na região próxima ao nariz e logo abaixo dos lábios, em que a iluminação não estava uniforme, pode-se perceber que a região de sombra foi suavizada o que não ocorre na maquiagem real.

VI. Conclusão

Esse trabalho apresenta um espelho virtual interativo para simulação de maquiagem que permite a escolha e aplicação desses produtos na imagem do usuário. Tal simulador tem como intuito possibilitar que o usuário experimente um produto de maquiagem de forma mais fácil e conveniente do que utilizando os cosméticos reais, podendo observar os resultados de forma mais rápida (sem a necessidade de preparação, além da aplicação e remoção da maquiagem ocorrerem instantaneamente) e barata (não é necessário comprar o produto ou gastar produtos já existentes). Durante o desenvolvimento do projeto, duas grandes questões foram abordadas, a Interação e a Simulação de Maquiagem, estas serão discutidos independentemente nos próximos parágrafos.

A interação com o sistema ocorre por meio do toque no monitor que faz o papel de espelho, permitindo a escolha da maquiagem desejada e também sua aplicação na imagem do espelho. A interface permite que o usuário possa aplicar a maquiagem da forma que ele quiser em sua face criando efeitos diversos, por exemplo, intensificando determinadas regiões ou passando mais de um produto de maquiagem na mesma região. Além disso, uma vez que a maquiagem foi aplicada, o sistema é capaz de manter o posicionamento da maquiagem durante a movimentação da face do usuário sem que isso interfira na fluidez da imagem do espelho e sem a necessidade de marcadores.

O protótipo desenvolvido segue as características citadas no parágrafo anterior e portanto está de acordo com o objetivo inicial do trabalho. Embora tenha sido possível contornar as limitações de *hardware* para mostrar a viabilidade da interação proposta, alguns sinais dessas limitações são perceptíveis. A imagem da face obtida com o Kinect possui baixa resolução o que degrada a qualidade da imagem observada pelo usuário. Além disso, a capacidade de processamento do computador utilizado também limitou o número de triângulos a serem usados para o posicionamento da maquiagem. Com o avanço natural dos dispositivos utilizados, ganha-se melhor qualidade de imagem e poder de processamento, o que resulta na melhoria da qualidade de imagem do espelho e também possibilita o uso de algoritmos computacionalmente mais caros melhorando o rastreamento da face, sua triangularização e consequentemente o posicionamento da maquiagem aplicada na face do usuário. Entretanto, diferentemente de outros trabalhos, a interface proposta não tem a necessidade de etapas preparatórias, como a criação de um modelo 3D da face do usuário ou uso de marcadores, e, portanto, se encontra pronta para o uso imediato do usuário.

Além da evolução do *hardware* já citada, também existem outras melhorias que podem ser realizadas em trabalhos futuros do simulador. No lugar da aplicação da maquiagem utilizando o toque no monitor, poderiam ser utilizados gestos na face, o que seria mais próximo da forma com que uma pessoa se maqueia na vida real. Outra melhoria consiste na simulação do resultado da maquiagem sob iluminação não uniforme e também diferentes tipos de iluminação como luz do sol, nublado, luz provinda de uma lâmpada incandescente ou fluorescente. Além disso, o algoritmo de rastreamento dos pontos faciais poderia ser substituído por outro método que não precisasse utilizar o Kinect, apenas uma câmera comum, o que ajudaria a diminuir o custo do projeto e abriria a possibilidade dele ser utilizado em dispositivos móveis. Por fim, podemos citar também o uso de homografia para alinhar a direção do olhar do usuário com a imagem do espelho.

Além da Interação, este trabalho apresenta um método para simular a aplicação de base, batom e sombra em uma imagem da face de uma pessoa sem maquiagem. Para realizar a simulação utiliza-se a ideia de decompor cada propriedade a ser simulada em uma ou mais camadas que podem ser tratadas independentemente. Para a simulação de base, além da simulação de cor, realizada no espaço de cor La*b*, foi feita uma homogeneização do tom de pele e uma suavização dos detalhes faciais por meio de suavizações em algumas camadas de faixas de frequência da imagem da face. Para batom e sombra, a simulação restringiu-se à introdução de cor nas regiões maquiadas.

Diferentemente dos trabalhos de transferência de maquiagem, nossa simulação permite a livre aplicação de tipos diferentes de maquiagem em qualquer área do rosto de forma independente, sem a necessidade de um modelo pré-existente e sem uma pose específica para o rosto (flexibilidade). Com relação aos trabalhos de simulação, nosso método vai além do que foi proposto por esses trabalhos, tornando o resultado mais realista e sem a necessidade de equipamentos especiais (custo). Além disso, o algoritmo proposto também pode ser computado em tempo real o que confere a presença das quatro características desejadas para o algoritmo simultaneamente atingindo o objetivo do algoritmo.

Apesar dos bons resultados, ainda existe espaço para trabalhos futuros. Ao utilizar a abordagem de camadas, novas camadas podem ser incorporadas no método proposto para abordar aspectos ainda não trabalhados como o tratamento de regiões especulares, a remoção dos pelos faciais antes da simulação da maquiagem na pele, iluminação não uniforme, tipos diferentes de fontes de luz e outros tipos de maquiagens. Também podemos citar a implementação do algoritmo em GPU para o tornar ainda mais rápido, a adição no módulo de Textura do realce da textura da pele e não somente a suavização e também um método para treinar um *maquilet* para que ele simule uma maquiagem real.

Como resultado deste trabalho, um artigo completo foi publicado no Simpósio de Realidade Virtual e Aumentada (SVR) 2014 [16] e outro artigo curto foi publicado no Simpósio Brasileiro Sobre Fatores Humanos em Sistemas Computacionais (IHC) 2014 [17]. Além disso, o trabalho teve repercussão em meios de notícias como [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29] entre outras republicações em blogs menores e é um projeto da *startup* fundada pelo autor do trabalho.

Referências

[1] DCI, Diário Comércio Indústria & Serviços, "Mercado brasileiro de cosméticos deve ser vice-líder até 2017, aponta pesquisa," 2013. [Online]. Available: http://www.dci.com.br/comercio/mercado-brasileiro-de-cosmeticos-deve-ser-vicelider-ate-2017,-aponta-pesquisa-id353256.html

[2] W.-S. Tong, C.-K. Tang, M. Brown, and Y.-Q. Xu, "Example-based cosmetic transfer," in *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*, 29 2007-nov. 2 2007, pp. 211 –218.

[3] K. Scherbaum, T. Ritschel, M. Hullin, T. Thormahlen, V. Blanz, and H.-P. Seidel, "Computer-suggested facial makeup," *Computer Graphics Forum*, 2011. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2011.01874.x

[4] D. Guo and T. Sim, "Digital face makeup by example," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, june 2009, pp. 73 –79.

[5] A. Dhall, G. Sharma, R. Bhatt, and G. Khan, "Adaptive digital makeup," in *Advances in Visual Computing*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2009, vol. 5876, pp. 728–736. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10520-3_69

[6] J.-S. Kim and S.-M. Choi, "Interactive cosmetic makeup of a 3d point-based face model," *IEICE Transactions on Information and Systems*, vol. E91.D, no. 6, pp. 1673–1680, 2008.

[7] C.-G. Huang, T.-S. Huang, W.-C. Lin, and J.-H. Chuang, "Physically-based cosmetic rendering," in *International Conference on Computer Animation and Social Agents*, 2013.

[8] E. Iwabuchi, M. Nakagawa, and I. Siio, "Smart makeup mirror: Computer-augmented mirror to aid makeup application," in *Proceedings of the 13th International Conference on Human-Computer Interaction. Part IV: Interacting in Various Application Domains*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 495–503. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02583-9_54

[9] A. Rahman, T. Tran, S. Hossain, and A. El Saddik, "Augmented rendering of makeup features in a smart interactive mirror system for decision support in cosmetic products selection," in *Distributed Simulation and Real Time Applications (DS-RT), 2010 IEEE/ACM 14th International Symposium on*, oct. 2010, pp. 203 –206.

[10] A. Hanafusa, S. Terada, Y. Miki, C. Sasagawa, T. Ikeda, and T. Fuwa, "Makeup support system for visually impaired persons: Overview of system functions." in *ICCHP (2)*, ser. Lecture Notes in Computer Science, K. Miesenberger, J. Klaus, W. L. Zagler, and A. I. Karshmer, Eds., vol. 6180. Springer, 2010, pp. 338–345. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14100-3_50

[11] M. Nakagawa, K. Tsukada, and I. Siio, "Smart makeup system: supporting makeup using lifelog sharing," in *Proceedings of the 13th international conference on Ubiquitous computing*, ser. UbiComp '11. New York, NY, USA: ACM, 2011, pp. 483–484. [Online]. Available: http://doi.acm.org/10.1145/2030112.2030182

[12] G. Bradski, "Opencv," *Dr. Dobb's Journal of Software Tools*, 2000.

[13] Microsoft, "Kinect for windows sdk." [Online]. Available: http://www.microsoft.com/en-us/kinectforwindowsdev/Downloads.aspx

[14] "Box blur," dec 2013, page Version ID: 517348792. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Box_blur&oldid=517348792

[15] O. A. R. Board, "Openmp application program interface," May 2008. [Online]. Available: http://openmp.org/wp/

[16] F. M. S. d. Campos and C. H. Morimoto, "Virtual makeup: foundation, eye shadow and lipstick simulation," in *Proceedings of the 16th Symposium on Virtual and Augmented Reality*, ser. SVR'14, 2014.

[17] F. M. S. Campos and C. H. Morimoto, "Interactive virtual mirror for makeup simulation," in *Proceedings of the 13th Brazilian Symposium on Human Factors in Computer Systems*, ser. IHC'14, 2014.

[18] TV Record, Jornal Fala Brasil, "Espelho virtual: programa simula resultado da maquiagem Edição de 1 de Maio de 2015," May 2015. [Online]. Available: http://noticias.r7.com/fala-brasil/videos/espelho-virtual-programa-simula-resultado-da-maquiagem-01052015

[19] Agência USP de Notícias, "Espelho virtual facilita escolha de produtos de beleza," 2015. [Online]. Available: http://www.usp.br/agen/?p=203487

[20] IME-USP, "Espelho virtual interativo para simulação de maquiagem," 2014. [Online]. Available: http://www.ime.usp.br/index.php?option=com_content&view=article&id=838&catid=19&Itemid=679&lang=pt-br

[21] USP, "Espelho virtual criado no ime tem simulação instantânea de maquiagem," 2015. [Online]. Available: http://www5.usp.br/88587/espelho-virtual-desenvolvido-no-ime-permite-simulacao-instantanea-de-maquiagem/

[22] USP Inovação, "Espelho virtual facilita escolha de produtos de beleza." 2015. [Online]. Available: http://uspinovacao.blogspot.com.br/2015/04/espelho-virtual-facilita-escolha-de.html

[23] Jornal Correio Braziliense, "E o computador se transforma em camarim. Edição de 30 de Março de 2015," March 2015. [Online]. Available: http://www.correiobraziliense.com.br/app/noticia/tecnologia/2015/03/30/interna_tecnologia,477507/e-o-computador-se-transforma-em-camarim.shtml

[24] Jornal Hoje em Dia, "Espelho, espelho meu! Edição de 26 de Abril de 2015," April 2015. [Online]. Available: http://www.hojeemdia.com.br/bela/espelho-espelho-meu-1.314099

[25] CRUESP, "Espelho virtual criado no ime tem simulação instantânea de maquiagem," 2015. [Online]. Available: http://www.cruesp.sp.gov.br/?p=7989

[26] Jornal Empresas e Negócios, "Espelho virtual facilita escolha de produtos de beleza. Edição de 18 de Março de 2015," 2015. [Online]. Available: http://www.jornalempresasenegocios.com.br/pagina_06_ed_2858.pdf

[27] Jornal do Comércio, "Espelho virtual facilita escolha de produtos de beleza. Edição de 31 de Março de 2015," March 2015. [Online]. Available: http://www.correiobraziliense.com.br/app/noticia/tecnologia/2015/03/30/interna_tecnologia,477507/e-o-computador-se-transforma-em-camarim.shtml

[28] EM.com.br, "Espelho virtual simula a prova da maquiagem," 2015. [Online]. Available: http://www.em.com.br/app/noticia/tecnologia/2015/04/12/interna_tecnologia,636681/espelho-virtual-simula-a-prova-da-maquiagem.shtml

[29] VDI – Associação de Engenheiros Brasil-Alemanha, "Simulador virtual de maquiagem," 2015. [Online]. Available: http://vdibrasil.com/home/home-atualidades-detalhe/simulador-virtual-de-maquiagem/75c3308931fb756e36faa6358524de4b/

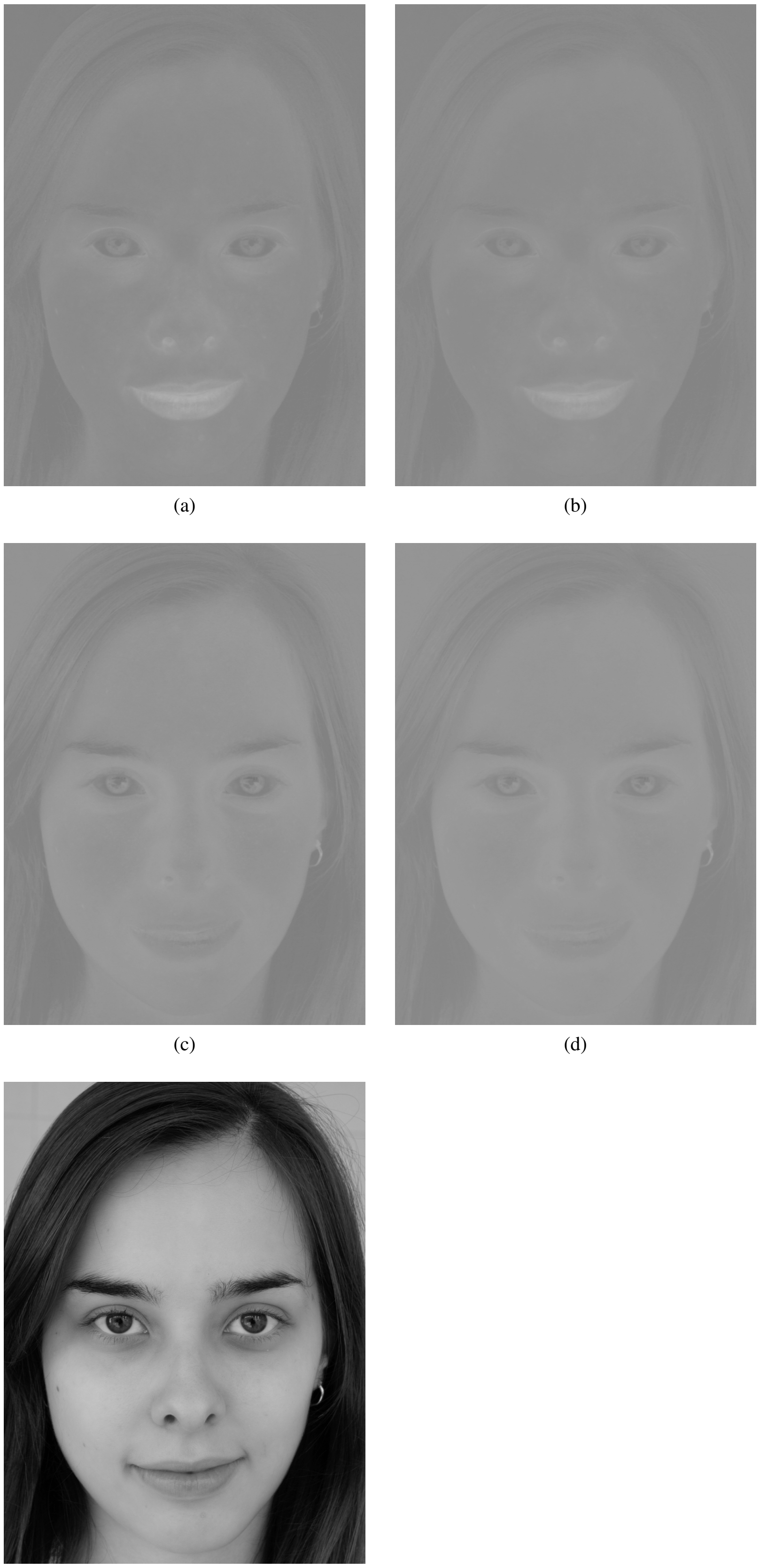
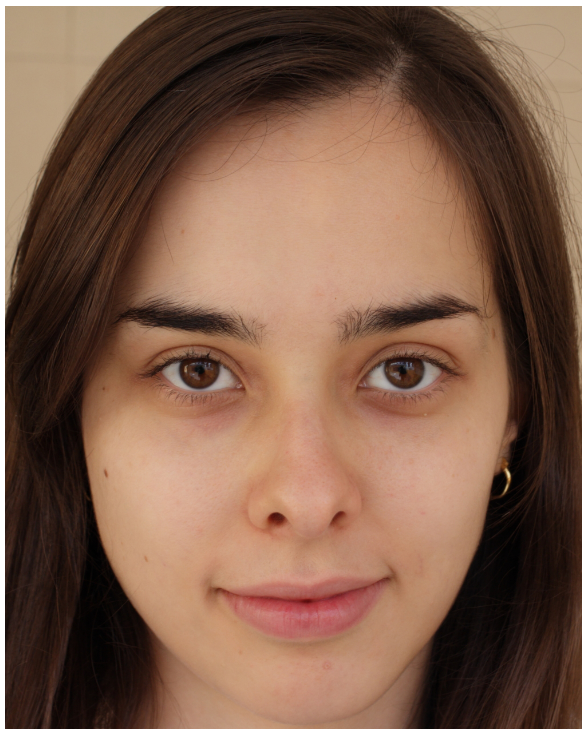
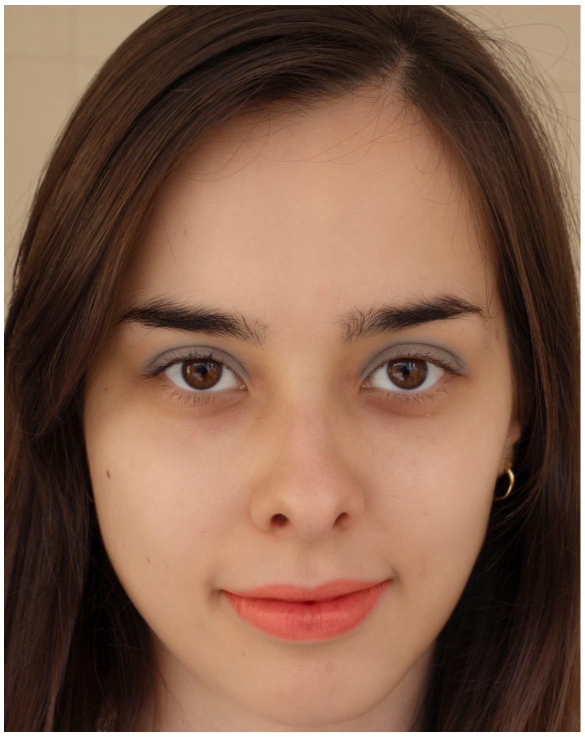


Figura 13. Camadas geradas utilizando o espaço de cor La*b* e resultados após média ponderada. (a) camada referente ao canal a*, C_{a*}; (b) resultado da camada referente ao canal a* após média ponderada; (c) camada referente ao canal b*, C_{b*}; (d) resultado da camada referente ao canal b* após média ponderada; (e) camada referente ao canal L, C_L.


(a)


(b)

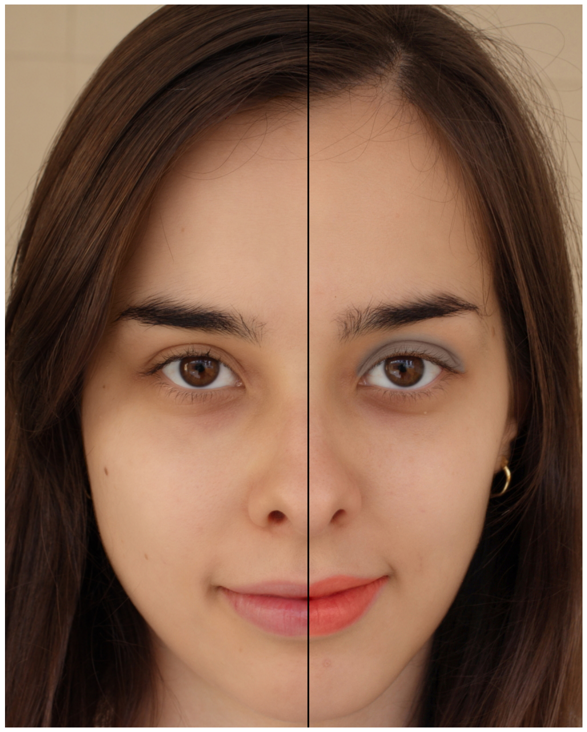

(c)

Figura 14. Resultado da simulação para batom, sombra e base. (a) Imagem original; (b) Resultado da simulação; (c) Comparativo, lado esquerdo: original, lado direito: maquiada.

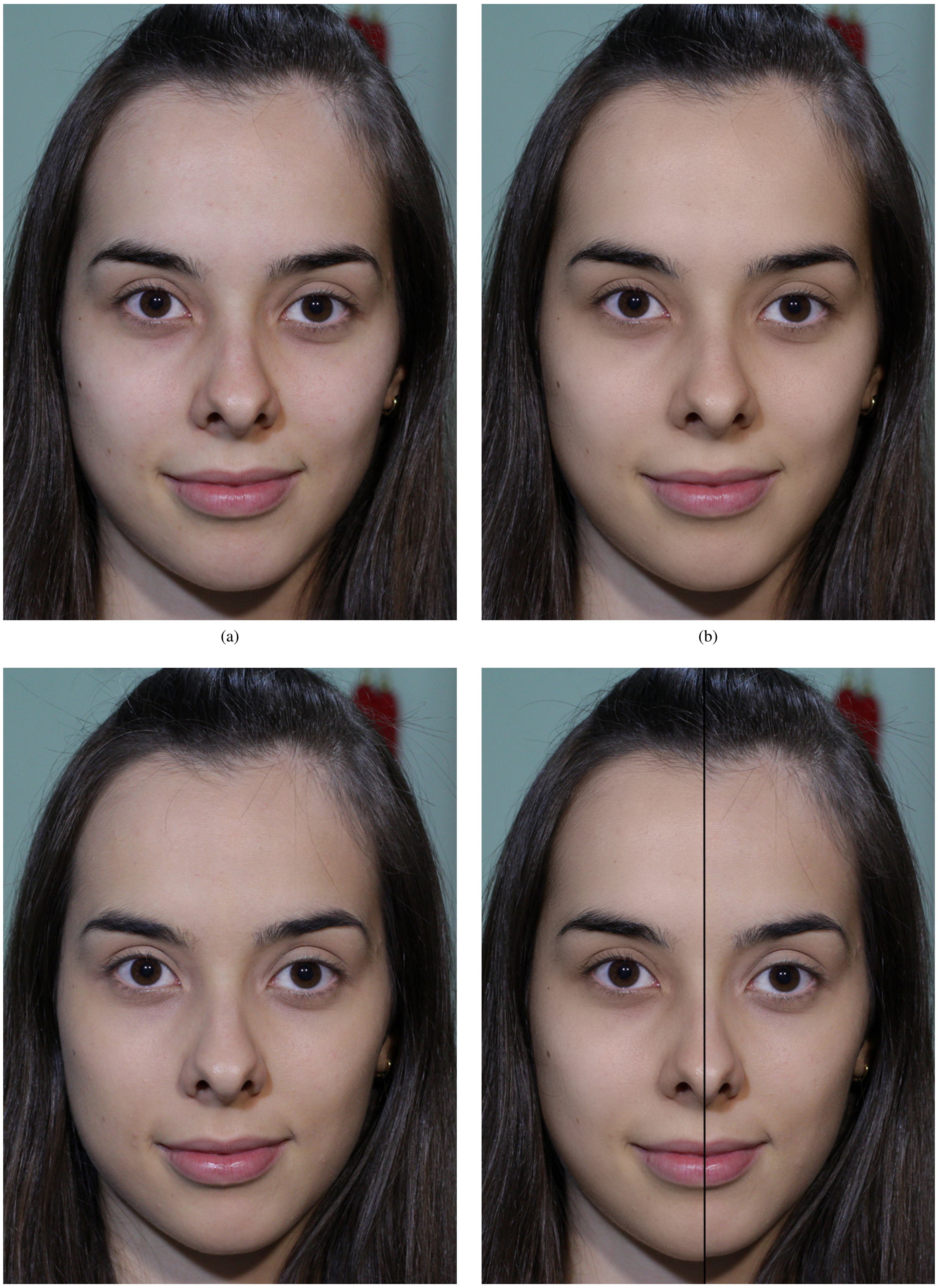


Figura 15. Comparativo entre o resultado da simulação e maquiagem real. (a) Imagem sem maquiagem; (b) Resultado da simulação de maquiagem; (c) Maquiagem real; (d) Comparativo, lado esquerdo: simulada, lado direto: real

Image Segmentation by Image Foresting Transform with Non-smooth Connectivity Functions

Lucy A. Choque Mansilla
Department of Computer Science,
University of São Paulo (USP),
05508-090, São Paulo, SP, Brazil.
Email: lucyacm@vision.ime.usp.br

***Abstract*—Image segmentation, such as to extract an object from a background, is very useful for medical and biological image analysis. In this work, we propose new methods for interactive segmentation of multidimensional images, based on the Image Foresting Transform (IFT), by exploiting for the first time non-smooth connectivity functions (NSCF) with a strong theoretical background. The new algorithms provide global optimum solutions according to an energy function of graph cut, subject to high-level boundary constraints (polarity and shape), or consist in a sequence of paths' optimization in residual graphs. Our experimental results indicate substantial improvements in accuracy in relation to other state-of-the-art methods, by allowing the customization of the segmentation to a given target object.**

Keywords*—graph cut; image foresting transform; oriented image foresting transform; non-smooth connectivity function; geodesic star convexity

I. Introduction

Image segmentation is one of the most fundamental and challenging problems in image processing and computer vision [1]. In medical image analysis, accurate segmentation results commonly require the user intervention because of the presence of structures with ill-defined borders, intensity non-standardness among images, field inhomogeneity, noise, artifacts, partial volume effects, and their interplay [2], [3]. The high-level, application-domain-specific knowledge of the user is also often required in the digital matting of natural scenes, because of their heterogeneous nature [4], [5]. These problems motivated the development of several methods for semi-automatic segmentation [6], [7], [8], [9], [10], [11], [12], aiming to minimize the user involvement and time required without compromising accuracy and precision.

One important class of interactive image segmentation comprises seed-based methods, which have been developed based on different theories, supposedly not related, leading to different frameworks, such as *watershed from markers* [9], [13], [14], *random walks* [15], *fuzzy connectedness* [16], [11], *graph cuts* [7], [17], *distance cut* [4], *image foresting transform* (IFT) [18], and *grow cut* [19]. The study of the relations among different frameworks, including theoretical and empirical comparisons, has a vast literature [20], [21], [22], [23]. However, these methods in most studies are restricted to undirected graphs, and the most time-efficient methods, including the IFT, present a lack of boundary regularization constraints. Moreover, the quality of their segmentation results with minimal user intervention, strongly depends on an adequate estimate of the weights assigned to the graph's arcs [24].

The main contribution of this work is a theoretical development to support the usage of ***non-smooth connectivity functions*** (NSCF) in the IFT, opening new perspectives in the research of image processing using graphs, since NSCF were, until now, avoided in the literature. More specifically, we prove that some NSCF can lead to optimum results according to a graph-cut measure on a digraph [25], [26] or consist in a sequence of paths' optimization in residual graphs. We have as main results:

1) The customization of the segmentation by IFT to match the global and local features of a target object:
    (a) The design of more adaptive and flexible connectivity functions, which allow better handling of images with strong inhomogeneity by using dynamic weights.
    (b) The orientation of the object's intensity transitions, i.e., bright to dark or the opposite (boundary polarity).
    (c) The shape constraints to regularize the segmentation boundary (geodesic star convexity constraint).
2) The development of an interactive segmentation tool within the software, called *Brain Image Analyzer* (BIA), to support research in neurology involving volumetric magnetic resonance images of a 3T scanner from the FAPESP-CInApCe (Figure 1).
3) A total of four conference papers were published in international events of high regard [26], [27], [28], [29], and one journal paper was published in the *IEEE Transactions on Image Processing* (impact factor: 3.111) [25].

For the sake of completeness in presentation, Section II includes the relevant previous work of image segmentation by IFT. In Sections III, IV, V and VI, we present the main contributions covered in the master's dissertation [30]: The classification of NSCF, the use of adaptive weights via NSCF, the boundary polarity through digraphs, and the elimination of false delineations by shape constraints. Our conclusions are stated in Section VII.

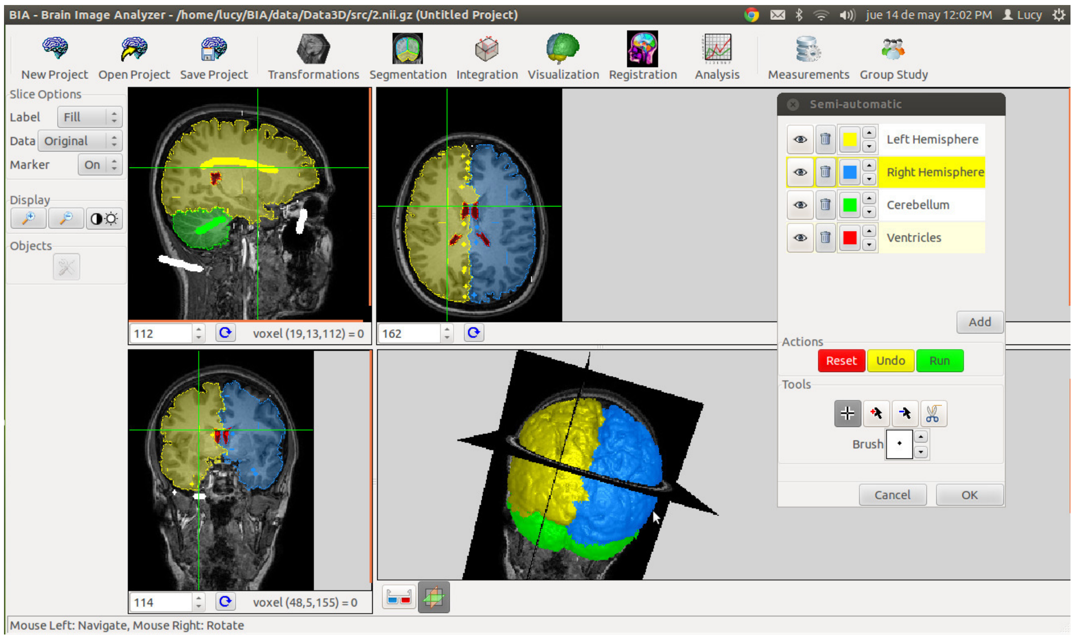


Fig. 1. Example of an interactive segmentation in progress of multiple objects in a 3D MRI image for user-selected markers using the software Brain Image Analyzer (BIA).

II. Image Foresting Transform (IFT)

An image 2D/3D can be interpreted as a weighted digraph $G = \langle \boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{I}}, \boldsymbol{\xi}, \omega \rangle$ whose nodes $\boldsymbol{\mathcal{V}}$ are the image pixels in its image domain $\boldsymbol{\mathcal{I}} \subset \mathbb{Z}^N$, and whose arcs are the ordered pixel pairs $(s,t) \in \boldsymbol{\xi}$ (e.g., 4-neighborhood, or 8-neighborhood, in case of 2D images, and 6-neighbors in 3D). The digraph G is symmetric if for any of its arcs (s,t), the pair (t,s) is also an arc of G. We have an undirected graph when $\omega(s,t) = \omega(t,s)$ in a symmetric graph G. We use $(s,t) \in \boldsymbol{\xi}$ or $t \in \xi(s)$ to indicate that t is adjacent to s. Each arc $(s,t) \in \boldsymbol{\xi}$ may have a weight $\omega(s,t) \geq 0$, such as a dissimilarity measure between pixels s and t (e.g., $\omega(s,t) = |I(t) - I(s)|$ for a single channel image with values given by $I(t)$).

For a given image graph G, a path $\pi_t = \langle t_1, t_2, \ldots, t_n = t \rangle$ is a sequence of adjacent pixels with terminus at a pixel t. A path is *trivial* when $\pi_t = \langle t \rangle$. A path $\pi_t = \pi_s \cdot \langle s,t \rangle$ indicates the extension of a path π_s by an arc (s,t). The notation $\pi_{s \rightsquigarrow t} = \langle t_1 = s, t_2, \ldots, t_n = t \rangle$ may also be used, where s stands for the origin and t for the destination node.

A *predecessor map* is a function P that assigns to each pixel t in $\boldsymbol{\mathcal{V}}$ either some other adjacent pixel in $\boldsymbol{\mathcal{V}}$, or a distinctive marker nil not in $\boldsymbol{\mathcal{V}}$ — in which case t is said to be a *root* of the map. A *spanning forest* is a predecessor map which contains no cycles — i.e., one which takes every pixel to nil in a finite number of iterations. For any pixel $t \in \boldsymbol{\mathcal{V}}$, a spanning forest P defines a path π_t^P recursively as $\langle t \rangle$ if $P(t) = nil$, and $\pi_s^P \cdot \langle s,t \rangle$ if $P(t) = s \neq nil$.

A *connectivity function* computes a value $f(\pi_t)$ for any path π_t, usually based on arc weights. A path π_t is *optimum* if $f(\pi_t) \leq f(\tau_t)$ for any other path τ_t in G. By taking to each pixel $t \in \boldsymbol{\mathcal{V}}$ one optimum path with terminus t, we obtain the optimum-path value $V(t)$, which is uniquely defined by $V(t) = \min_{\forall \pi_t \text{ in } G}\{f(\pi_t)\}$. A path $\pi_{t_n} = \langle t_1, t_2, \ldots, t_n \rangle$ is *complete optimum* if all paths $\pi_{t_i} = \langle t_1, t_2, \ldots, t_i \rangle$, $i = 1, 2, \ldots, n$ are optimum paths.

The IFT takes an image graph G, and a path-cost function f; and assigns one optimum path π_t to every pixel $t \in \boldsymbol{\mathcal{V}}$ such that an *optimum-path forest* P is obtained — i.e., a spanning forest P where all paths π_t^P, $t \in \boldsymbol{\mathcal{V}}$, are optimum. However, f must be *smooth* (Definition 1), otherwise, the paths may not be optimum [18].

Definition 1 (Smooth path-cost function). *A path-cost function f is* smooth *if for any pixel $t \in \boldsymbol{\mathcal{I}}$, there is an optimum path π_t, which either is trivial, or has the form $\pi_s \cdot \langle s,t \rangle$ where*

(C1) $f(\pi_s) \leq f(\pi_t)$,
(C2) *π_s is optimum,*
(C3) *C2 is valid and for any optimum path π'_s ending at s, $f(\pi'_s \cdot \langle s,t \rangle) = f(\pi_t)$.*

The cost of a trivial path $\pi_t = \langle t \rangle$ is usually given by a handicap value $H(t)$, while the connectivity functions for non-trivial paths follow a path-extension rule. For example:

$$f_{\max}(\pi_t = \pi_s \cdot \langle s,t \rangle) = \max\{f_{\max}(\pi_s),\ \omega(s,t)\} \quad (1)$$

$$f_{sum}(\pi_t = \pi_s \cdot \langle s,t \rangle) = f_{sum}(\pi_s) + \omega_2(s,t) \quad (2)$$

$$f_{\omega}(\pi_t = \pi_s \cdot \langle s,t \rangle) = \omega(s,t) \quad (3)$$

where $\omega(s,t) \geq 0$ and $\omega_2(s,t) \geq 0$ are fixed arc weights.

We consider image segmentation from two seed sets, $\boldsymbol{\mathcal{S}_o}$ and $\boldsymbol{\mathcal{S}_b}$ ($\boldsymbol{\mathcal{S}_o} \cap \boldsymbol{\mathcal{S}_b} = \emptyset$), containing pixels selected inside and outside the object, respectively. The search for optimum paths is constrained to start in $\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{S}_o} \cup \boldsymbol{\mathcal{S}_b}$ (i.e., $H(t) = 0$ for all $t \in \boldsymbol{\mathcal{S}}$, and $H(t) = +\infty$ otherwise). The image is partitioned into two optimum-path forests — one rooted at the internal seeds, defining the object, and the other rooted at the external seeds, representing the background. A label, $L(t) = 1$ for all $t \in \boldsymbol{\mathcal{S}_o}$ and $L(t) = 0$ for all $t \in \boldsymbol{\mathcal{S}_b}$, is propagated to all unlabeled pixels during the computation [18].

In the IFT, the optimum-path forest may not be unique [18]. For example, if two or more seeds lead to a pixel t through optimum paths with the same cost , then more than one forest may be optimum. Paths $\pi_{r_1 \rightsquigarrow t}$ and $\tau_{r_2 \rightsquigarrow t}$ with the same label (i.e., $\{r_1, r_2\} \subset \boldsymbol{\mathcal{S}_o}$ or $\{r_1, r_2\} \subset \boldsymbol{\mathcal{S}_b}$) are not a problem, because they lead to the same segmentation result and any solution is satisfactory. For paths with different labels, we have the basis of the real *tie zones*. The *tie zones* are the maximal set of *tie-zone pixels* [1], which forms a subtree in some optimum-path forest [31].

III. IFT with Non-Smooth Connectivity Functions

Clearly, from Definition 1, we have that a connectivity function is not smooth if it doesn't satisfy at least one of the conditions C1, C2 or C3. For example the functions $f_{\Sigma\max}$, $f_{\sum|\triangle I|}$, $f_{\max|\triangle I|}$, $f_{\updownarrow}$ and $f^{bkg}_{\max}$ [2] violate C2 and C3:

$$\begin{aligned} f_{\Sigma\max}(\pi_t = \langle t \rangle) &= \begin{cases} 0, & \text{if } t \in \boldsymbol{\mathcal{S}}, \\ +\infty, & \text{otherwise.} \end{cases} \\ f_{\Sigma\max}(\pi_t = \pi_s \cdot \langle s,t \rangle) &= f_{\Sigma\max}(\pi_s) + f_{\max}(\pi_t) \end{aligned} \quad (4)$$

$$\begin{aligned} f_{\sum|\triangle I|}(\pi_t = \langle t \rangle) &= \begin{cases} 0, & \text{if } t \in \boldsymbol{\mathcal{S}}, \\ +\infty, & \text{otherwise.} \end{cases} \\ f_{\sum|\triangle I|}(\pi_t = \pi_{r \rightsquigarrow s} \cdot \langle s,t \rangle) &= f_{\sum|\triangle I|}(\pi_{r \rightsquigarrow s}) + |I(t) - I(r)| \end{aligned} \quad (5)$$

$$\begin{aligned} f_{\max|\triangle I|}(\pi_t = \langle t \rangle) &= \begin{cases} 0, & \text{if } t \in \boldsymbol{\mathcal{S}}, \\ +\infty, & \text{otherwise.} \end{cases} \\ f_{\max|\triangle I|}(\pi_t = \pi_{r \rightsquigarrow s} \cdot \langle s,t \rangle) &= \max\{f_{\max|\triangle I|}(\pi_{r \rightsquigarrow s}), \\ &\qquad |I(t) - I(r)|\} \end{aligned} \quad (6)$$

$$\begin{aligned} f_{\updownarrow}(\pi_t = \langle t \rangle) &= \begin{cases} 0, & \text{if } t \in \boldsymbol{\mathcal{S}}, \\ +\infty, & \text{otherwise.} \end{cases} \\ f_{\updownarrow}(\pi_t = \pi_s \cdot \langle s,t \rangle) &= f_{Imax}(\pi_t) - f_{Imin}(\pi_t) \end{aligned} \quad (7)$$

where f_{Imax} and f_{Imin} are functions that take the maximum and minimum intensity values along the path, respectively:

$$\begin{aligned} f_{Imax}(\pi_t = \langle t \rangle) &= I(t) \\ f_{Imax}(\pi_t = \pi_s \cdot \langle s,t \rangle) &= \max\{f_{Imax}(\pi_s), I(t)\} \end{aligned} \quad (8)$$

and

$$\begin{aligned} f_{Imin}(\pi_t = \langle t \rangle) &= I(t) \\ f_{Imin}(\pi_t = \pi_s \cdot \langle s,t \rangle) &= \min\{f_{Imin}(\pi_s), I(t)\} \end{aligned} \quad (9)$$

[1]A pixel t is a *tie-zone pixel* if there exist two complete optimum paths $\pi_{r_1 \rightsquigarrow t}$ and $\tau_{r_2 \rightsquigarrow t}$ such that $r_1 \in \boldsymbol{\mathcal{S}_b}$ and $r_2 \in \boldsymbol{\mathcal{S}_o}$.

[2]The function $f^{bkg}_{\max}$ incorporates a tie-breaking rule inside its path-value definition to resolve ties of the function $f_{\max}$ (Equation 1) by favoring background seeds. Since this is the same behavior exhibited by the *Iterative Relative Fuzzy Connectedness* (IRFC), the formulation of $f^{bkg}_{\max}$ corresponds to an alternative IFT-based codification for the IRFC method [32].

where $I(t)$ is the intensity of a pixel t.

$$\begin{aligned} f^{bkg}_{\max}(\pi_t = \langle t \rangle) &= \begin{cases} -1 & \text{if } t \in \boldsymbol{\mathcal{S}_o} \cup \boldsymbol{\mathcal{S}_b} \\ +\infty & \text{otherwise} \end{cases} \\ f^{bkg}_{\max}(\pi_t = \pi_{r \rightsquigarrow s} \cdot \langle s,t \rangle) &= \begin{cases} Expr1 & \text{if } r \in \boldsymbol{\mathcal{S}_o} \\ Expr2 & \text{if } r \in \boldsymbol{\mathcal{S}_b} \end{cases} \end{aligned} \quad (10)$$

$$\begin{aligned} Expr1 &= \max\{f^{bkg}_{\max}(\pi_{r \rightsquigarrow s}), 2 \times \omega(s,t) + 1\} \\ Expr2 &= \max\{f^{bkg}_{\max}(\pi_{r \rightsquigarrow s}), 2 \times \omega(s,t)\} \end{aligned}$$

In [29], we formally classified several non-smooth functions according to the conditions C1, C2 and C3 (Definition 1), and C4 (Definition 2).

Definition 2 (Condition C4)**.** *A path-value function f satisfies the condition C4, if for any node $s \in \boldsymbol{\mathcal{I}}$ the following condition is verified $\forall t \in \xi(s)$:*

- *For any paths π_s and π'_s ending at s, if $f(\pi_s) = f(\pi'_s)$, then we have $f(\pi_s \cdot \langle s,t \rangle) = f(\pi'_s \cdot \langle s,t \rangle)$.*

For a general image graph, the classification of various non-smooth functions into the sets $\boldsymbol{C_1}$, $\boldsymbol{C_2}$, $\boldsymbol{C_3}$, and $\boldsymbol{C_4}$ (such that a function f is in a set $\boldsymbol{C_i}$ if and only if it satisfies the condition C_i) is shown in the proposed diagram illustrated in Figure 2. The functions $f^{\circlearrowleft}_{\omega}$, $f_{i,\omega}$ and $f_{o,\omega}$ were studied in [33], [26], and f_I are defined in [29].

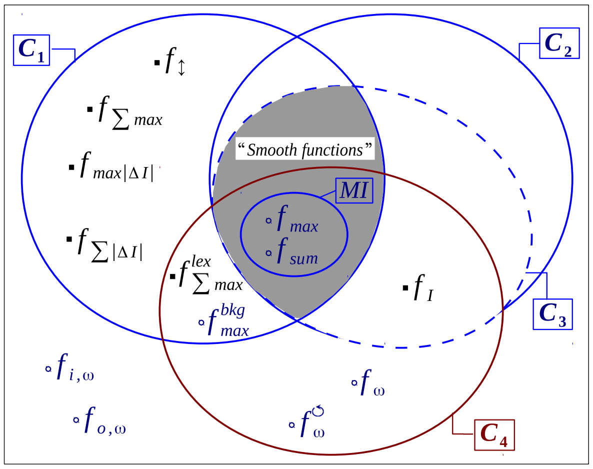


Fig. 2. Schematic representation of the relations between smooth and non-smooth connectivity functions: $\boldsymbol{C_1}$, $\boldsymbol{C_2}$, $\boldsymbol{C_3}$ and $\boldsymbol{C_4}$ are sets of connectivity functions that satisfy these respective conditions for a general graph.

Some functions in the subset $\boldsymbol{C_1} \setminus (\boldsymbol{C_2} \bigcup \boldsymbol{C_4})$ have corresponding variations in the region $(\boldsymbol{C_1} \cap \boldsymbol{C_4}) \setminus \boldsymbol{C_2}$ of the diagram. That is possible by using a second cost component with comparisons in lexicographic order. For example, the lexicographical function $f^{lex}_{\Sigma\max}$ (Equation 11) is a variation of the non-smooth function $f_{\Sigma\max}$ (Equation 4).

$$\begin{aligned} f^{lex}_{\Sigma\max}(\pi_t = \langle t \rangle) &= \begin{cases} (0,0), & \text{if } t \in \boldsymbol{\mathcal{S}}, \\ (+\infty, +\infty), & \text{otherwise.} \end{cases} \\ f^{lex}_{\Sigma\max}(\pi_t = \pi_s \cdot \langle s,t \rangle) &= (f_{\Sigma\max}(\pi_t), f_{\max}(\pi_t)). \end{aligned} \quad (11)$$

IV. Adaptive weights via NSCF

Methods based on IFT [18] have been successfully used in the segmentation of 1.5 Tesla MR datasets [34], [35]. However, inhomogeneity effects are stronger in higher magnetic fields (Figure 3), and it is extremely important to define the optimal solution for these images. NSCFs are more adaptive to cope with the problems of field inhomogeneity, which are common in MR images of 3 Tesla [36].

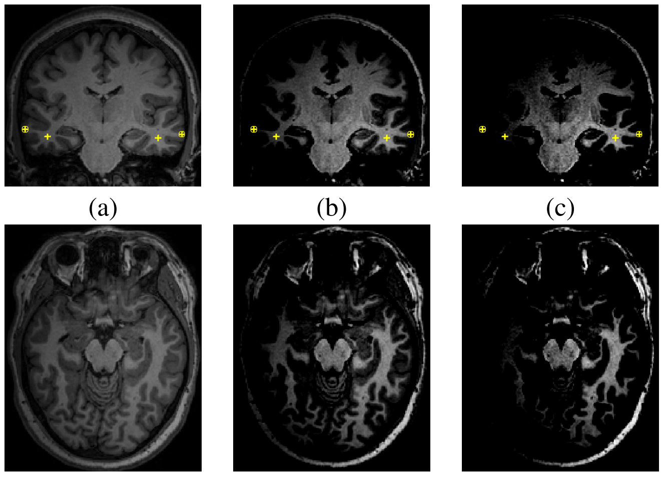


Fig. 3. Example of field inhomogeneity. (a) The yellow markers +/⊕ indicate corresponding regions in different tissues of the hemispheres. On the left, the gray and white matter show intensity values of 57 and 145, respectively, for the selected points. In the right region of the image, we have considerably higher values, 109 and 221. (a-c) By varying the brightness and contrast configurations, it is possible to better observe the problem. (d-f) Variations of brightness and contrast in an axial slice, revealing the inhomogeneity.

In order to give a theoretical foundation to support the usage of NSCF, we theoretically proved that the IFT with any non-smooth function $f \in (\boldsymbol{C_1} \cap \boldsymbol{C_4}) \setminus \boldsymbol{C_2}$ is, in fact, equivalent to the result of a sequence of optimizations, each of them involving a maximal set of elements, in a well-structured way [29]. This proof was supported by the following proposition:

Proposition 1. *Consider a function $f \in (\boldsymbol{C_1} \cap \boldsymbol{C_4}) \setminus \boldsymbol{C_2}$. For a given image graph $G = \langle \boldsymbol{\mathcal{V}}, \boldsymbol{\xi}, \omega \rangle$, and set of seeds $\boldsymbol{\mathcal{S}}$, let $\boldsymbol{O}$ be the set of all pixels $t \in \boldsymbol{\mathcal{V}}$, such that there exists a complete-optimum path π_t for f. In any spanning forest P computed in G by the IFT algorithm for f, all the paths τ_t^P with $t \in \boldsymbol{O}$ are optimum paths.*

Consider the following definitions: Let $\boldsymbol{\xi^{set}}(\boldsymbol{X}) = \{(s,t) \in \boldsymbol{\xi} \mid s \in \boldsymbol{X} \wedge t \in \boldsymbol{X}\}$ denote the set of all arcs interconnecting nodes in the set $\boldsymbol{X}$, $\boldsymbol{\xi^{path}}(\pi)$ denote the set of all arcs in the path π (i.e., $\boldsymbol{\xi^{path}}(\pi) = \{(t_i, t_{i+1})$ for $1 \leq i \leq k-1 | \; \pi = \langle t_1, t_2, \ldots, t_i, t_{i+1}, \ldots, t_k \rangle\})$, $\boldsymbol{\xi^{cut}}(\boldsymbol{X}, \boldsymbol{Y}) = \{(s,t) \in \boldsymbol{\xi} | s \in \boldsymbol{X} \wedge t \in \boldsymbol{Y}\}$, $\boldsymbol{\xi^{pred}}(\boldsymbol{X}) = \bigcup_{\forall t \in \boldsymbol{X}} \boldsymbol{\xi^{path}}(\pi_t^P)$.

In the first optimization step, optimum paths τ_t^P are computed for all $t \in \boldsymbol{O}$ (Proposition 1). Let's denote $\boldsymbol{O}$ as $\boldsymbol{O^1}$ for this first step. In the next optimization step, consider the subgraph $G^2 = \langle \boldsymbol{\mathcal{I}}, \boldsymbol{\xi^{set}}(\boldsymbol{\mathcal{I}} \setminus \boldsymbol{O^1}) \cup \boldsymbol{\xi^{cut}}(\boldsymbol{O^1}, \boldsymbol{\mathcal{I}} \setminus \boldsymbol{O^1}) \cup \boldsymbol{\xi^{pred}}(\boldsymbol{O^1}), \omega \rangle$. A second path optimization is performed, by computing a second IFT, but now in G^2 [3]. Since the arcs interconnecting nodes in $\boldsymbol{O^1}$, are reduced to the arcs in the previous forest P (i.e., $\boldsymbol{\xi^{pred}}(\boldsymbol{O^1})$) in G^2, we have that the optimum paths τ_t^P, computed on the previous step, will remain optimum in the new graph G^2. So the optimum paths τ_t^P with $t \in \boldsymbol{O^1}$ will start a new competition, seeking for their best extensions to the other pixels in $\boldsymbol{\mathcal{I}} \setminus \boldsymbol{O^1}$. By applying the Proposition 1 on this new optimization problem one more time, we have that this second IFT will conquer a new maximal set of pixels $\boldsymbol{O^1} \cup \boldsymbol{O^2}$ that can be reached by optimum paths in G^2. We can then repeat this process over again. The condition $C1$[4] guarantees that at least one new element will be conquered at each step, so that this process will repeat until $\bigcup_{\forall i} \boldsymbol{O^i} = \boldsymbol{\mathcal{I}}$.

Figure 4 shows an example of the ordered process about the sequence of optimizations for a non-smooth function $f^{lex}_{\Sigma \max} \in (\boldsymbol{C_1} \bigcap \boldsymbol{C_4}) \setminus \boldsymbol{C_2}$.

Figure 5 shows an example about the benefits of the non-smooth function $f_{\max|\triangle I|}$ (Equation 6) compared to the smooth connectivity function $f_{\max}$ (Equation 1) in image segmentation with inhomogeneity problem. Note that the function $f_{\max|\triangle I|}$ is more adaptive to cope with problems of inhomogeneity, by offering adaptive weights to the pixels in the image graph.

In our experiments, we used 10 T1-weighted 3D images of male and female adults with normal brains. The image dataset included the head and, at least, a small portion of the neck. Our experimental result, using a robot user [5] for segmenting the brain dataset, indicates that substantial improvements can be obtained by NSCFs in the 3D segmentation of MR images of 3 Tesla, with strong inhomogeneity effects, when compared to smooth connectivity functions. That is because NSCFs permit a more adaptive configuration of the arc weights.

Figure 6 shows the experimental curves, where $IRFC$ [16] and $PW_{q=2}$ [23] represent different algorithms related to the smooth function $f_{\max}$, and we used $\omega(s,t) = G(s) + G(t)$, where $G(s)$ is the magnitude of Sobel gradient at a voxel s. Clearly, $f^{lex}_{\sum|\triangle I|}$ presented the best accuracy. Figures 7 and 8 show examples for user-selected markers. These results emphasize the importance of non-smooth connectivity functions. The non-smooth connectivity function $f^{lex}_{\sum|\triangle I|}$ is a variation of $f_{\sum|\triangle I|}$ (Equation 5), in order to guarantee that $f^{lex}_{\sum|\triangle I|} \in (\boldsymbol{C_1} \cap \boldsymbol{C_4}) \setminus \boldsymbol{C_2}$. The function $f^{lex}_{\sum|\triangle I|}$ gives pairs of values that should be compared according to the lexicographical order. The first component is the non-smooth function $f_{\sum|\triangle I|}$ (Equation 5), and the second is the priority level of the seed/root for that path. The lower its value the higher is its priority. In interactive segmentation, we give lower priority for new inserted seeds, since they are used mainly for corrective actions, so that we can keep their effects more locally. The same process was done for $f^{lex}_{\max|\triangle I|} \in (\boldsymbol{C_1} \cap \boldsymbol{C_4}) \setminus \boldsymbol{C_2}$ and $f^{lex}_{\updownarrow} \in (\boldsymbol{C_1} \cap \boldsymbol{C_4}) \setminus \boldsymbol{C_2}$, in relation to $f_{\max|\triangle I|}$ (Equation 6) and $f_{\updownarrow}$ (Equation 7), respectively [29].

[3]By IFT algorithm, the pixels $s \in \boldsymbol{O^1}$ have a defined status, and their paths can't be changed. Therefore, we consider a new graph G^2, where the arcs interconnecting nodes in $\boldsymbol{O^1}$ which are not in the forest P are disregarded.

[4]By the hypothesis, we know that f satisfies the condition $C1$.

[5]Method introduced in [37], to simulate user interaction of interactive segmentation.

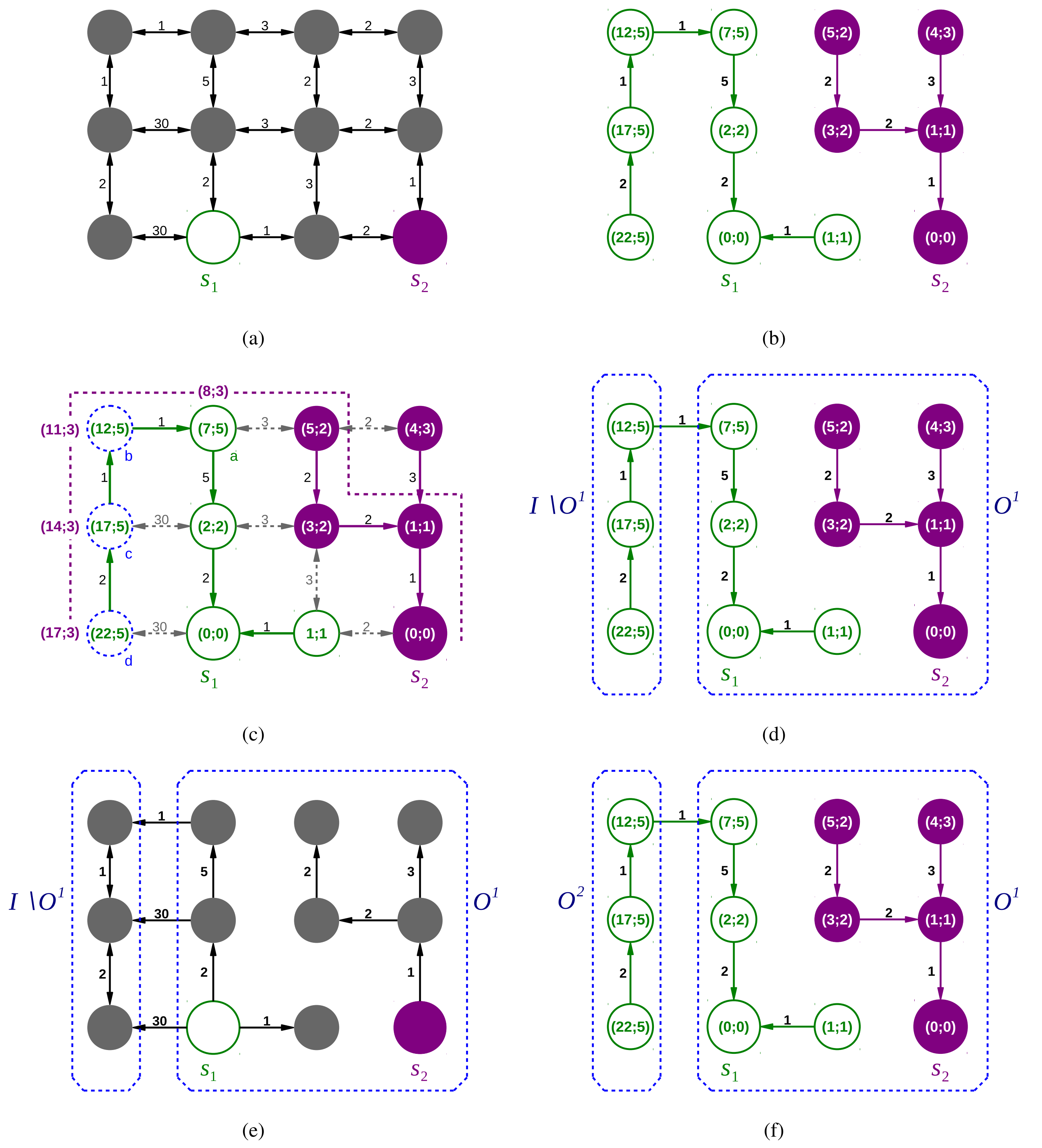
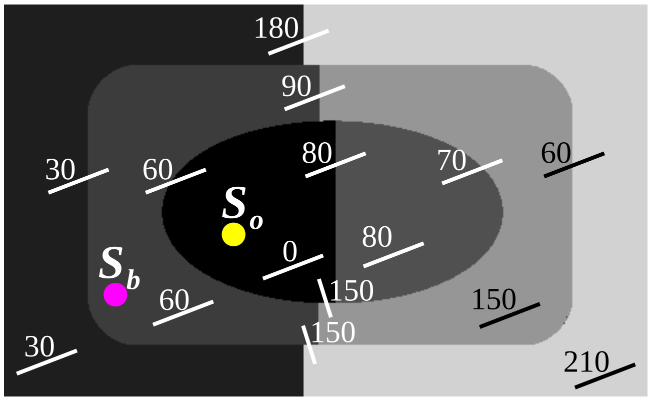
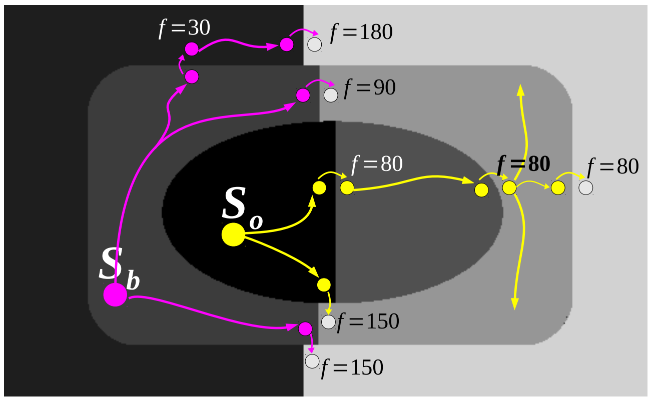


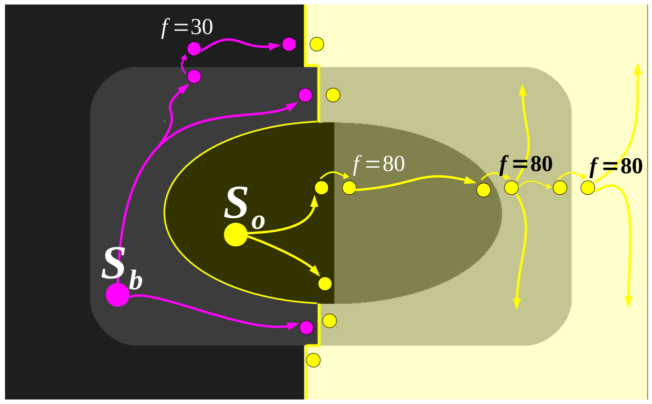
Fig. 4. (a) An image graph G using 4-neighbors for pixel adjacency and seeds $\boldsymbol{\mathcal{S}} = \{s_1, s_2\}$, where the numbers in the arcs represent their weights. (b) Spanning forest computed by the IFT with a non-smooth function $f^{lex}_{\Sigma\max} \in (\boldsymbol{C_1} \bigcap \boldsymbol{C_4}) \setminus \boldsymbol{C_2}$. The numbers inside the nodes indicate values of the path-value map V with two components in lexicographical order. In this figure the arrows indicate the predecessor map. (c) Note that the path $\pi_{s_1 \rightsquigarrow a}$ is a complete-optimum path, but the path $\pi_{s_1 \rightsquigarrow b}$ is not optimum, since that there is another path $\pi'_{s_2 \rightsquigarrow b}$, following the dashed line, offering a better cost (i.e., $f^{lex}_{\Sigma\max}(\pi'_{s_2 \rightsquigarrow b}) = (11;3) < f^{lex}_{\Sigma\max}(\pi_{s_1 \rightsquigarrow b}) = (12;5)$). This shows that the function $f^{lex}_{\Sigma\max}$ is not smooth. (d) By applying the Proposition 1 we have a forest computed for the set $\boldsymbol{O^1}$, composed of optimum paths in the graph G (i.e., first optimization). (e) We have the residual graph G^2, which will be used in a second optimization. (f) Optimum forest computed from the graph G^2, ending the process with $\boldsymbol{O^1} \bigcup \boldsymbol{O^2} = \boldsymbol{\mathcal{I}}$.



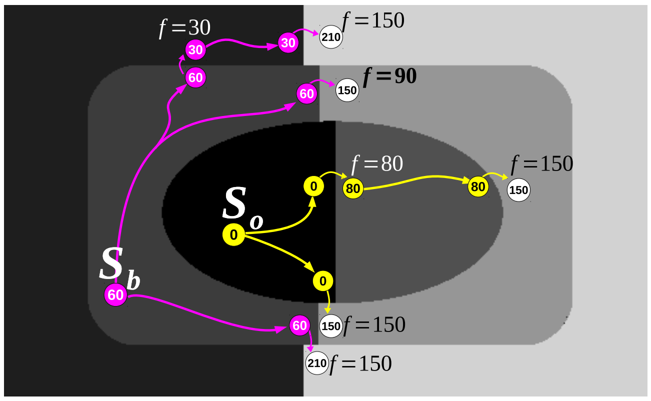
(a)



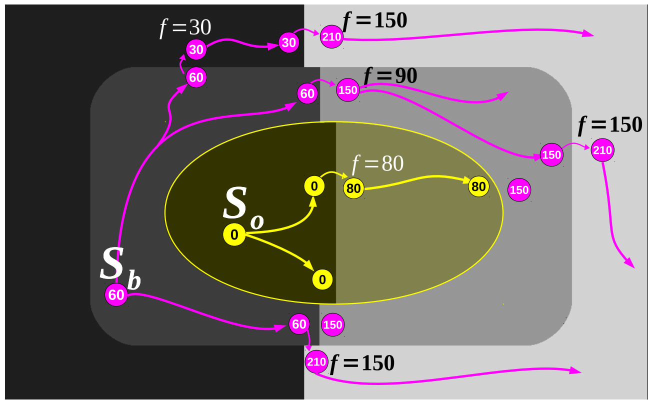
(b)



(c)



(d)



(e)

Fig. 5. (a) Synthetic image with problem of inhomogeneity. The target object for the image segmentation is the central object with elliptical shape. The user-selected markers are the seeds $\mathcal{S}_o$ and $\mathcal{S}_b$. The numbers are some representative arc values $\omega(s,t)$ depicted for each region and border segment. (b-c) Schematic representation of the path costs f offered by the seeds for the pixels in the image using the smooth function $f_{\max}$ and the segmentation result. (d-e) Schematic representation of the path costs f offered by the seeds for the pixels in the image using the non-smooth function $f_{\max|\triangle I|}$ and the segmentation result. The values inside of the nodes represent the intensities of the pixels.

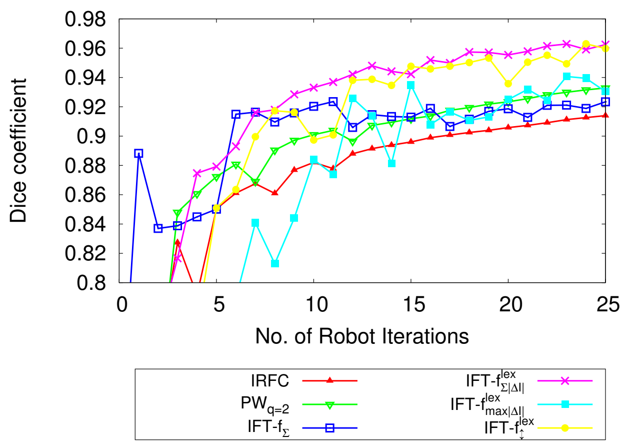


Fig. 6. Results using a robot user for segmenting the 3D brain dataset.

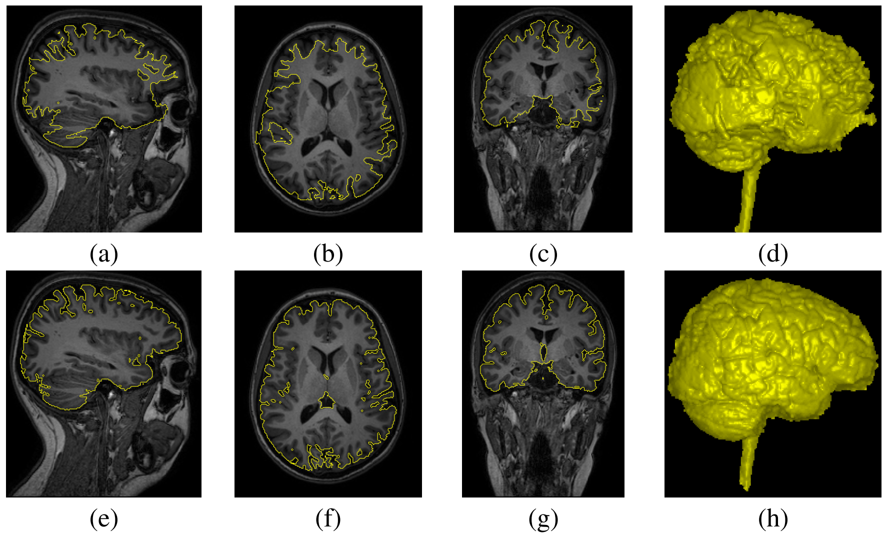


Fig. 7. Brain segmentation results for the same user-selected markers by (a-d) $f_{\max}$, and (e-h) $f^{lex}_{\sum|\triangle I|}$.

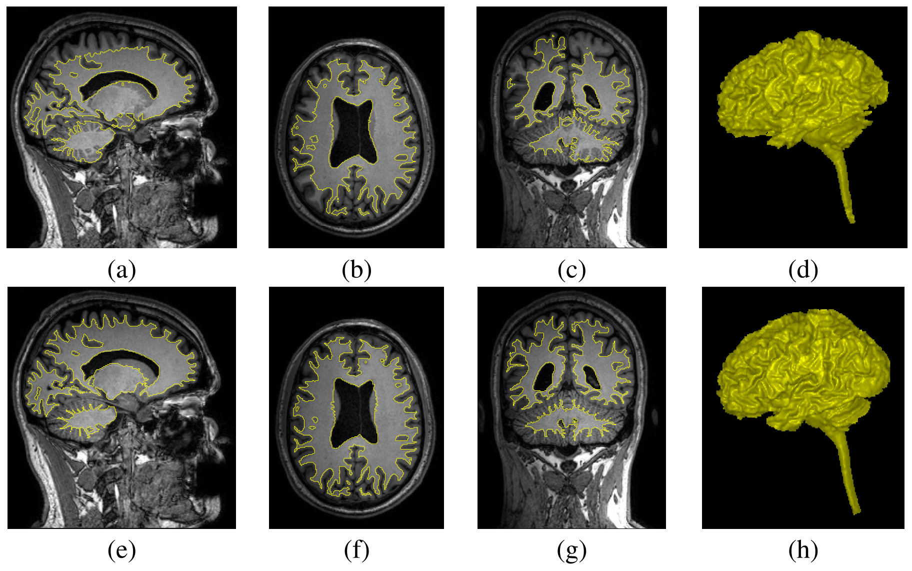


Fig. 8. White matter segmentation in 3D for the same user-selected markers. Results by (a-d) f_{max} over an enhanced gradient, and (e-h) $f^{lex}_{\sum|\triangle I|}$.

V. Boundary Polarity via NSCF

Boundary-based segmentation methods such as like live wire [6], [10], can easily incorporate boundary orientation to resolve between very similar nearby boundary segments (Figure 9), by favoring segmentation on a single orientation (e.g., *counter-clockwise* orientation).

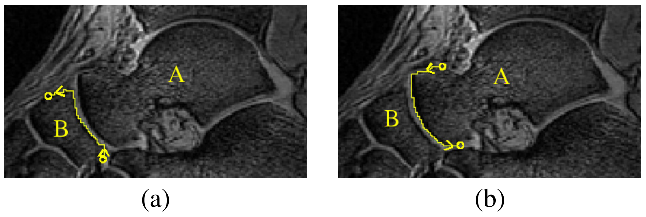


Fig. 9. Example of boundary-based segmentation, where the boundary-orientedness property of live wire helps it to distinguish between two similar boundary segments from the distinct objects A and B [38].

In [25], [26] we successfully incorporated the boundary polarity constraint in the IFT using NSCF in digraphs, resulting in a novel method called ***Oriented Image Foresting Transform*** (OIFT).

In the case of digraphs, there are two different types of cut for each object boundary: an inner-cut boundary composed by arcs that point toward object pixels $\mathcal{C}_i(L)$ (Figure 10a) and an outer-cut boundary with arcs from object to background pixels $\mathcal{C}_o(L)$ (Figure 10b).

$$\mathcal{C}_i(L) = \{(s,t) \in \boldsymbol{\xi} \mid L(s) = 0,\ L(t) = 1\} \tag{12}$$

$$\mathcal{C}_o(L) = \{(s,t) \in \boldsymbol{\xi} \mid L(s) = 1,\ L(t) = 0\} \tag{13}$$

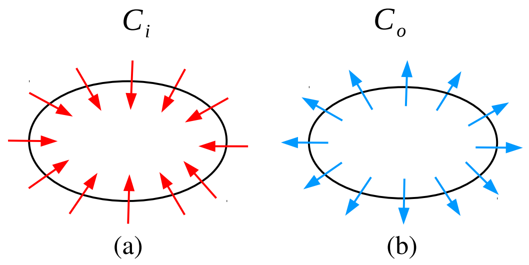


Fig. 10. Schematic representation of (a) inner cuts and (b) outer cuts.

Consequently, we consider two different types of energy, E_i (Equation 14) and E_o (Equation 15).

$$E_i(L,G) = \min_{(s,t)\ \in\ \mathcal{C}_i(L)} \omega(s,t) \tag{14}$$

$$E_o(L,G) = \min_{(s,t)\ \in\ \mathcal{C}_o(L)} \omega(s,t) \tag{15}$$

We use a digraph, where $\omega(s,t)$ is a combination of a regular undirected dissimilarity measure $\psi(s,t)$, multiplied by an orientation factor, as follows:

$$\omega(s,t) = \begin{cases} \psi(s,t) \times (1+\alpha) & \text{if } I(s) > I(t), \\ \psi(s,t) \times (1-\alpha) & \text{otherwise.} \end{cases} \tag{16}$$

Several different procedures can be adopted for $\psi(s,t)$, such as the absolute value of the difference of image intensities (i.e., $\psi(s,t) = |I(t) - I(s)|$). Note that we have $\omega(s,t) \neq \omega(t,s)$ when $\alpha > 0$.

The OIFT is build upon the IFT framework by considering one of the following path functions in a symmetric digraph:

$$\begin{aligned} f^{bkg}_{i,\max}(\langle t \rangle) &= \begin{cases} -1 & \text{if } t \in \mathcal{S}_o \cup \mathcal{S}_b \\ +\infty & \text{otherwise} \end{cases} \\ f^{bkg}_{i,\max}(\pi_{r \rightsquigarrow s} \cdot \langle s,t \rangle) &= \begin{cases} Expr1 & \text{if } r \in \mathcal{S}_o \\ Expr2 & \text{if } r \in \mathcal{S}_b \end{cases} \\ Expr1 &= \max\{f^{bkg}_{i,\max}(\pi_{r \rightsquigarrow s}), 2 \times \omega(t,s) + 1\} \\ Expr2 &= \max\{f^{bkg}_{i,\max}(\pi_{r \rightsquigarrow s}), 2 \times \omega(s,t)\} \end{aligned} \tag{17}$$

$$\begin{aligned} f^{bkg}_{o,\max}(\langle t \rangle) &= \begin{cases} -1 & \text{if } t \in \mathcal{S}_o \cup \mathcal{S}_b \\ +\infty & \text{otherwise} \end{cases} \\ f^{bkg}_{o,\max}(\pi_{r \rightsquigarrow s} \cdot \langle s,t \rangle) &= \begin{cases} Expr1 & \text{if } r \in \mathcal{S}_o \\ Expr2 & \text{if } r \in \mathcal{S}_b \end{cases} \\ Expr1 &= \max\{f^{bkg}_{o,\max}(\pi_{r \rightsquigarrow s}), 2 \times \omega(s,t) + 1\} \\ Expr2 &= \max\{f^{bkg}_{o,\max}(\pi_{r \rightsquigarrow s}), 2 \times \omega(t,s)\} \end{aligned} \tag{18}$$

$$\begin{aligned} f_{i,\omega}(\langle t \rangle) &= \begin{cases} -1 & \text{if } t \in \mathcal{S}_o \cup \mathcal{S}_b \\ +\infty & \text{otherwise} \end{cases} \\ f_{i,\omega}(\pi_{r \rightsquigarrow s} \cdot \langle s,t \rangle) &= \begin{cases} \omega(t,s) & \text{if } r \in \mathcal{S}_o \\ \omega(s,t) & \text{if } r \in \mathcal{S}_b \end{cases} \end{aligned} \tag{19}$$

$$\begin{aligned} f_{o,\omega}(\langle t \rangle) &= \begin{cases} -1 & \text{if } t \in \mathcal{S}_o \cup \mathcal{S}_b \\ +\infty & \text{otherwise} \end{cases} \\ f_{o,\omega}(\pi_{r \rightsquigarrow s} \cdot \langle s,t \rangle) &= \begin{cases} \omega(s,t) & \text{if } r \in \mathcal{S}_o \\ \omega(t,s) & \text{if } r \in \mathcal{S}_b \end{cases} \end{aligned} \tag{20}$$

Figure 11 shows an example demonstrating that the function $f^{bkg}_{o,\max}$ is not smooth.

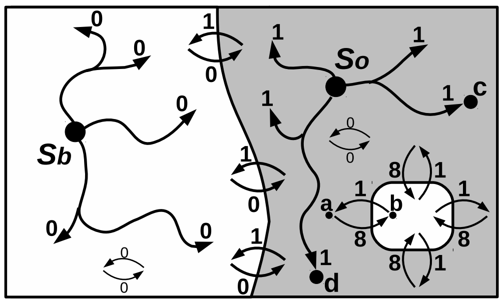


Fig. 11. The function $f^{bkg}_{o,\max}$ process reversed edges for paths from $\mathcal{S}_b$. Suppose that $\mathcal{S}_b$ conquers the left white region with 0 cost, since its pixels are interconnected by 0-weighted arcs. The pixels within the gray region have also 0-weighted arcs, so $\mathcal{S}_o$ provides better costs to them ($2 \times 0 + 1 = 1$ against $2 \times 1 = 2$ offered by $\mathcal{S}_b$). The best cost offered by $\mathcal{S}_o$ to the node b is $2 \times 8 + 1 = 17$, while $\mathcal{S}_b$ could have provided a better cost to b ($2 \times 1 = 2$), but b is assigned to the object. Thus, the function is not smooth.

The segmentation using $f^{bkg}_{i,\max}$ or $f_{i,\omega}$ favors transitions from dark to bright pixels, and $f^{bkg}_{o,\max}$ or $f_{o,\omega}$ favors the opposite orientation (Figure 12), according to Theorem 1. In the case of multiple candidate segmentations with the same energy, $f_{i,\omega}$ and $f_{o,\omega}$ produces a better handling of the *tie zones* than $f^{bkg}_{i,\max}$ and $f^{bkg}_{o,\max}$, respectively (Figure 13) [26].

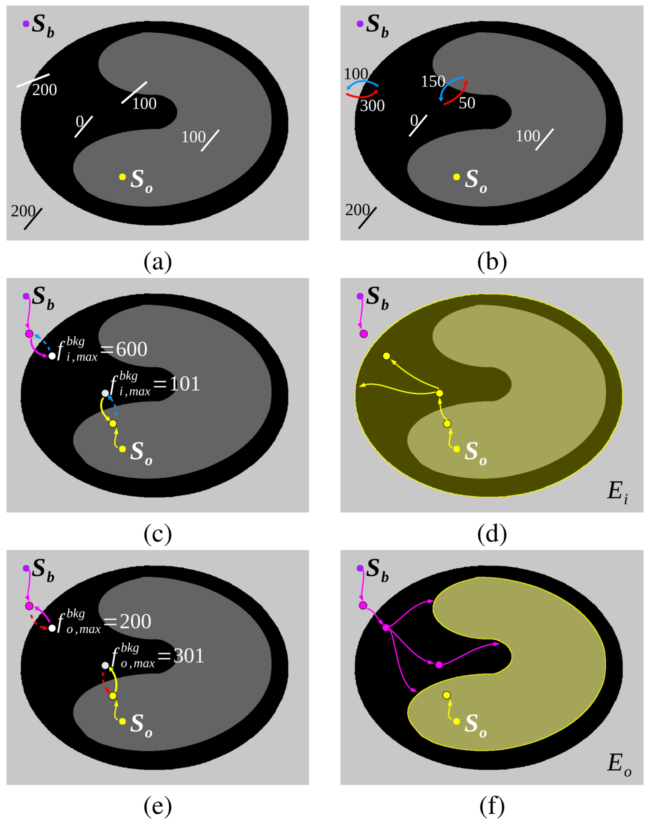


Fig. 12. (a) A synthetic image with internal and external seeds ($\mathcal{S}_o$ and $\mathcal{S}_b$). Some representative arc values $\omega(s,t)$ are depicted for each region and border segment. (b) Arc values $\omega(s,t)$ for border segments according to Equation 16 with $\alpha = 0.5$. (c-d) Transition from dark to bright pixels favors the segmentation using $f^{bkg}_{i,\max}$ and maximizes $E_i(L,G)$. (e-f) Transition from bright to dark pixels favors the segmentation using $f^{bkg}_{o,\max}$ and maximizes $E_o(L,G)$.

Theorem 1 (Inner/outer-cut boundary optimality)**.** *For two given sets of seeds $\mathcal{S}_o$ and $\mathcal{S}_b$, any spanning forest computed by the IFT algorithm for function $f^{bkg}_{o,\max}$ or $f_{o,\omega}$ defines an optimum cut that maximizes $E_o(L,G)$ among all possible segmentation results satisfying the hard constraints. Any spanning forest computed by the IFT algorithm for function $f^{bkg}_{i,\max}$ or $f_{i,\omega}$ defines an optimum cut that maximizes $E_i(L,G)$ among all possible segmentation results satisfying the hard constraints.*

In our experiments, we used four 2D datasets, to segment different objects in MRI and CT slice images; two 3D datasets of MR images and one dataset of colored images. In the first experiment, we used 40 slice images from real MR images of the foot, and we computed the mean performance curve for the functions $f^{bkg}_{\max}$, $f^{bkg}_{o,\max}$ and $f^{bkg}_{i,\max}$, to segment the talus bone, for different seed sets obtained by eroding and dilating the ground truth, as shown in Figure 14a-b. For the second

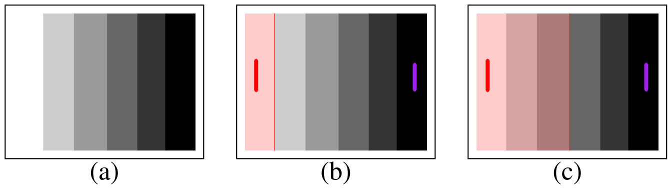


Fig. 13. (a) Input image with equally weighted transitions, having the same orientation. (b) The OIFT result using $f^{bkg}_{o,\max}$ as proposed in [25] assigns the ambiguous regions to the background. (c) OIFT using $f_{o,\omega}$ with a *FIFO* tie-breaking policy gives more equally balanced partitions.

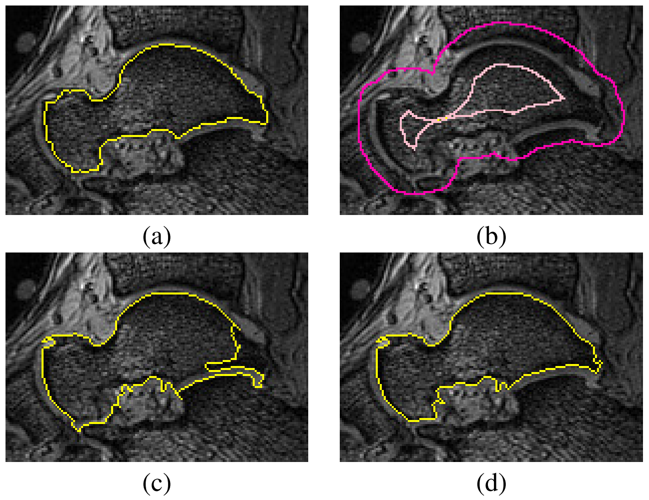


Fig. 14. (a) True segmentation of the talus in an MR image of a foot. (b) Seed sets obtained by eroding and dilating the true segmentation. (c) Segmentation result by IRFC. (d) An improved segmentation result is obtained by exploiting the boundary orientation using $f^{bkg}_{i,\max}$.

dataset, we performed the segmentation of the calcaneus for all the connectivity functions, using 40 slices from MR images of the foot. In the third experiment, 40 slice images were selected from CT cervical spine studies of 10 subjects to segment the spinal-vertebra. Finally, 40 slice images from CT thoracic studies of 10 subjects were used to segment the liver. This gave us a total of 160 2D-images to be processed by each function. The experimental accuracy curves are given in Figures 15, 16, 17 and 18, which show that whenever the object presents transitions from dark to bright pixels $f^{bkg}_{i,\max}$ gives the best accuracy results, and whenever there are transitions from bright to dark pixels $f^{bkg}_{o,\max}$ obtains the top accuracy. Note also that $f^{bkg}_{i,\max}$ gives the best results when $f^{bkg}_{o,\max}$ presents the worst accuracy, and vice versa. This means that by specifying the wrong orientation, it becomes even harder to get the desired boundary as compared to the undirected approach $f^{bkg}_{\max}$, as we expected.

In the 3D experiments, first we used a dataset of 3D MRI images, composed of brain images of 20 normal subjects. We performed the 3D segmentation of the cerebelum for the functions $f^{bkg}_{\max}$, $f^{bkg}_{i,\max}$ and $f^{bkg}_{o,\max}$ in all volumes. Figure 19 shows the obtained results, which indicate that $f^{bkg}_{o,\max}$ improves the accuracy.

Second, we used 20 real volumetric MR images of the foot in 3D. We computed the mean performance curve (Dice

Method	Description	Graph Type
$IRFC$	Iterative Relative Fuzzy Connectedness [16]	undirected graph
$IFT^{\max}_{FIFO}$	Traditional IFT with $f_{\max}$and FIFO tie-breaking policy, as described in [18]	undirected graph
$PW_{q=2}$	The power watershed algorithm [23]	undirected graph
$OIFT^{\max}_{inner}$	OIFT result using $f^{bkg}_{i,\max}$	weighted digraph
$OIFT^{\max}_{outer}$	OIFT result using $f^{bkg}_{o,\max}$	weighted digraph
$OIFT^{\omega}_{inner}$	OIFT using $f_{i,\omega}$	weighted digraph
$OIFT^{\omega}_{outer}$	OIFT using $f_{o,\omega}$	weighted digraph

TABLE I. DESCRIPTION OF THE METHODS USED IN THE EXPERIMENTS.

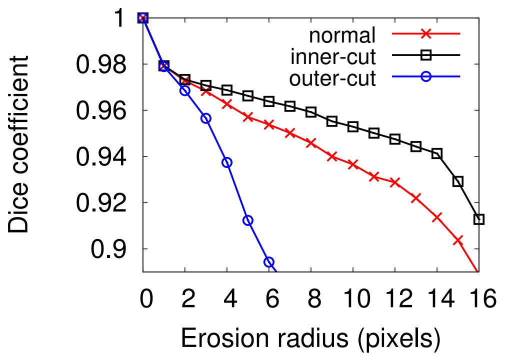


Fig. 15. The mean accuracy curves for $f^{bkg}_{\max}$ (normal), $f^{bkg}_{i,\max}$ (inner), $f^{bkg}_{o,\max}$ (outer) for the segmentation of talus.

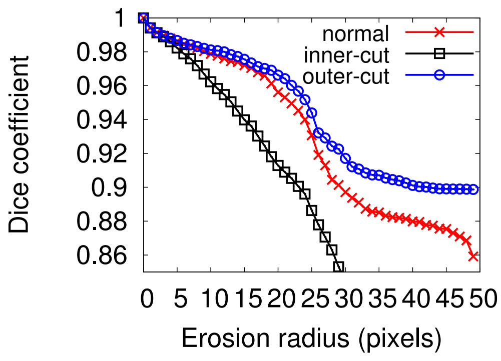


Fig. 18. The mean accuracy curves for $f^{bkg}_{\max}$ (normal), $f^{bkg}_{i,\max}$ (inner), $f^{bkg}_{o,\max}$ (outer) for the segmentation of liver.

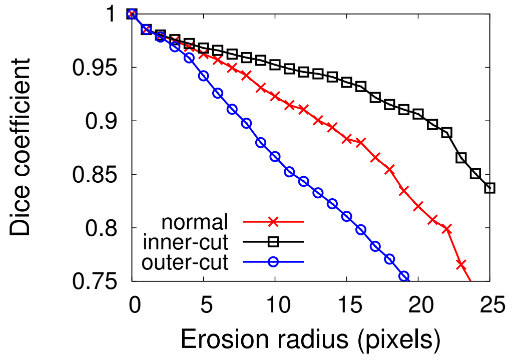


Fig. 16. The mean accuracy curves for $f^{bkg}_{\max}$ (normal), $f^{bkg}_{i,\max}$ (inner), $f^{bkg}_{o,\max}$ (outer) for the segmentation of calcaneus.

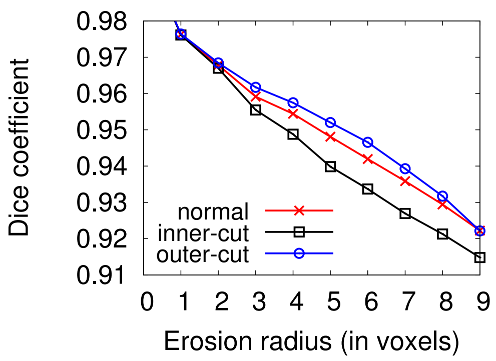


Fig. 19. Mean accuracy curves for $f^{bkg}_{\max}$ (normal), $f^{bkg}_{i,\max}$ (inner), $f^{bkg}_{o,\max}$ (outer) for the 3D segmentation of the cerebellum.

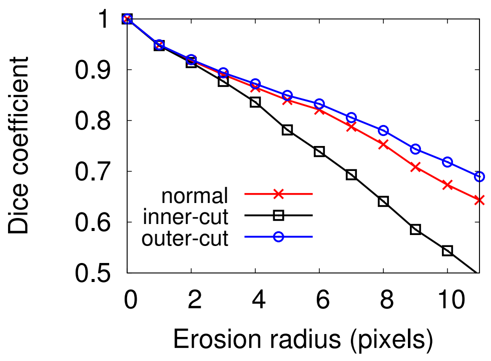


Fig. 17. The mean accuracy curves for $f^{bkg}_{\max}$ (normal), $f^{bkg}_{i,\max}$ (inner), $f^{bkg}_{o,\max}$ (outer) for the segmentation of spinal-vertebra.

coefficient) for all the methods depicted on Table I [26]. The experimental accuracy curves with the Sobel gradient (Figure 20) show that whenever the object presents transitions from dark to bright pixels, as it is the case with the bones talus and calcaneus, $f^{i,\omega}$ and $f^{bkg}_{i,\max}$ give the best accuracy results. Note also that $f^{bkg}_{o,\max}$ and $f_{o,\omega}$ present the worst accuracy values, by specifying the wrong orientation.

In the experiment with colored images, we used 50 natural images with known true segmentations obtained from [39]. The arc weights were computed by the method proposed in [24], using a few manual-selected training markers to estimate the color models. Figure 21 shows the experimental curves, where $f_{o,\omega}$ provided better mean results. Note that $f_{i,\omega}$ presented the worst accuracy values in this case, since it considers the wrong boundary orientation.

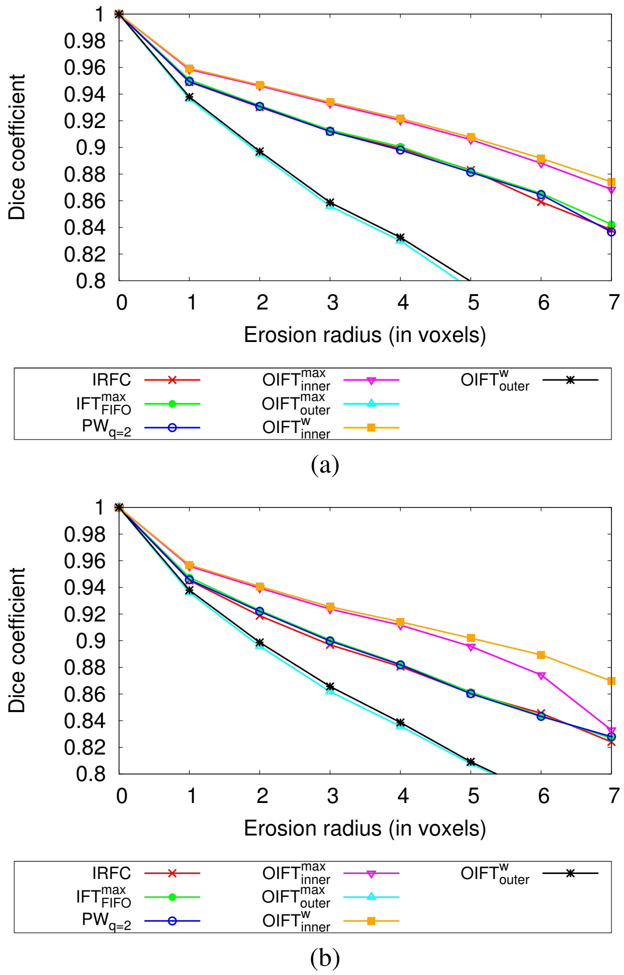


Fig. 20. The mean accuracy curves (Dice) using the Sobel gradient for the 3D segmentation of: (a) talus, and (b) calcaneus.

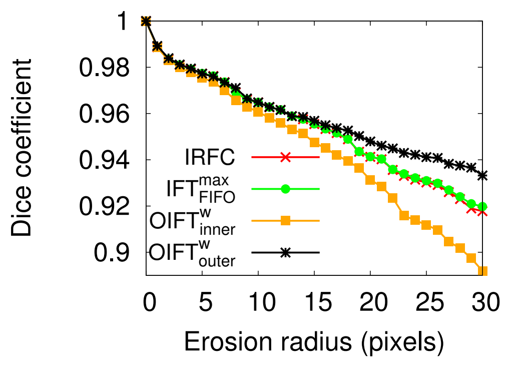


Fig. 21. The mean accuracy curve for different methods using color images.

VI. Shape Constraints via NSCF

Structures poorly defined in medical images, as well as natural images, are often hard to be segmented due to their low contrast in relation to other nearby false boundaries (Figure 22). The usage of shape constraints can help to alleviate part of this problem for objects that have a more regular shape.

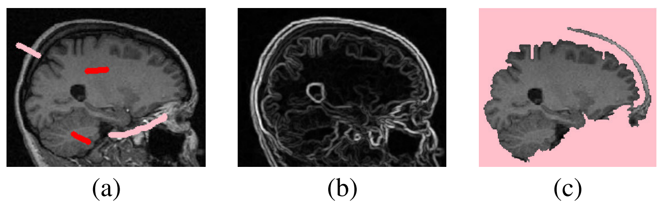


Fig. 22. (a) MR image of the brain (object with regular shape). (b) Image gradient. (c) Segmentation result of the brain by IFT with $f_{\max}$, which outputs a very irregular shape.

Shape constraints, such as the star-convexity prior introduced by Veksler [40], can limit the search space of possible delineations to a smaller subset, thus eliminating false candidate boundaries. In this context, a point p is said to be visible to c via a set $\mathcal{O}$ if the line segment joining p to c lies in the set $\mathcal{O}$. An object $\mathcal{O}$ is star-convex with respect to center c, if every point $p \in \mathcal{O}$ is visible to c via $\mathcal{O}$ (Figure 23). In the case of multiple stars, a computationally tractable definition, was proposed in [37], using a *Geodesic Star Convexity* (GSC) constraint in the segmentation by *min-cut/max-flow*.

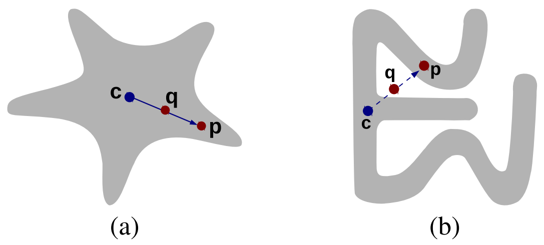


Fig. 23. For any point p within the object and the star center c, we have: (a) a star-convex object and (b) a non-star-convex object.

In [27], we proposed a new algorithm (Algorithm 1) that incorporates the GSC constraint in the energy maximization by IFT [31], favoring the segmentation of objects with more regular shape, resulting in a novel method called ***IFT with Geodesic Star Convexity Constraints*** (GSC–IFT). In this method, the set of star centers is taken as the set of internal seeds ($\mathcal{S}_o$), and the line segments are the paths that form a spanning forest rooted at the internal seeds. The arc weights $\omega_2(s,t)$ in the path-extension rule for f_{sum} (Equation 2) are given by:

$$\omega_2(s,t) \quad = \quad [\omega(s,t)+1]^{\beta} - 1 + \|t-s\| \qquad (21)$$

where $\|t-s\|$ is the Euclidean distance between pixels s and t, and β controls the forest topology.

For lower values of β ($\beta \approx 0.0$), $\omega_2(s,t)$ approaches $\|t-s\|$, and the forest topology becomes similar to the Euclidean shortest-path forest (Figure 24a). For higher values, $[\omega(s,t)+1]^{\beta}$ dominates the expression, and the greater the intensity-based dissimilarity, the greater is its influence over the results (Figure 24b-d).

Figure 25 shows how the parameter β affects the forest for f_{sum}, and its corresponding segmentation with shape constraints. Clearly, for lower values of β the method imposes more star regularization to the object's boundary, while for higher values of β, it allows a better fit to the curved protrusions and indentations of the boundary.

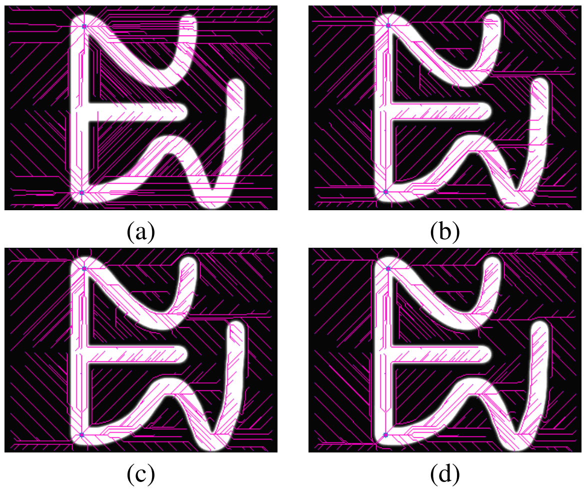


Fig. 24. The effects of the power parameter β over the forest topology: The optimum-path forest of f_{sum} for: (a) $\beta = 0.0$, (b) $\beta = 0.3$, (c) $\beta = 0.4$, and (d) $\beta = 0.5$.

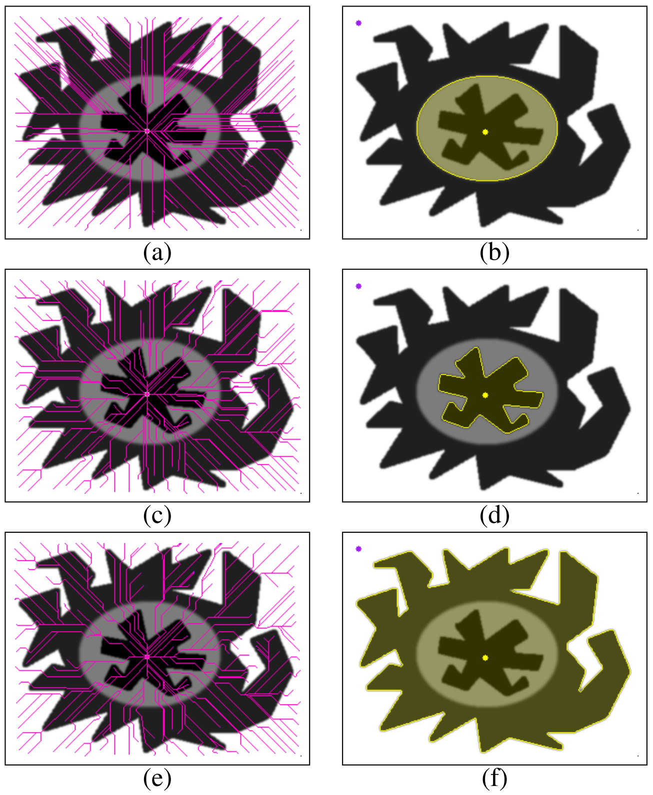


Fig. 25. The effects of the power parameter β over the forest topology: (a) The optimum-path forest of f_{sum} for $\beta = 0$, and (b) its resulting segmentation by the GSC–IFT algorithm (Algorithm 1). Similar results for: (c-d) $\beta = 0.3$, and (e-f) $\beta = 0.5$.

Algorithm 1 obtains a segmentation that maximizes a graph-cut measure among all possible segmentation results satisfying the shape constraints by *Geodesic Star Convexity* (GSC), whose theoretical proof of its optimality has been proved in [27]. In its first step (Line 1), the optimum forest P_{sum} for f_{sum} (Equation 2) is computed by invoking the regular IFT algorithm, using only $\mathcal{S}_o$ (set of star centers) as seeds. The subsequent Lines 2–8 are similar to the original IFT with f_ω (Equation 3). The differences arise from this point (Lines 9–12).

Algorithm 1. – GSC-IFT ALGORITHM

INPUT:	Graph $G = \langle \mathcal{I}, \boldsymbol{\xi}, \omega \rangle$, and seed sets $\mathcal{S}_o$ and $\mathcal{S}_b$.
OUTPUT:	Minimum path-value map V, and label map L.
AUXILIARY:	Optimum-path forest P_{sum}, priority queue Q, variable tmp, and an array of $status$.

```
1.  Compute $P_{sum} \leftarrow IFT(G, \mathcal{S}_o, \emptyset, f_{sum})$.
2.  For each $t \in \mathcal{S}_o$, do $L(t) \leftarrow 1$.
3.  For each $t \in \mathcal{S}_b$, do $L(t) \leftarrow 0$.
4.  For each $t \in \mathcal{I}$, do
5.      Set $V(t) \leftarrow f_\omega(\langle t \rangle)$, and set $status(t) \leftarrow 0$.
6.      If $V(t) \neq +\infty$, then insert $t$ in $Q$.
7.  While $Q \neq \emptyset$, do
8.      Remove $s$ from $Q$ such that $V(s)$ is minimum.
9.      If $L(s) = 1$, then
10.         $Conquer\_Path(s, G, P_{sum}, V, Q, L, status)$.
11.     Else If $L(s) = 0$, then
12.         $Prune\_Tree(s, G, P_{sum}, V, Q, L, status)$.
```

Algorithm 2. – CONQUER_PATH ALGORITHM

INPUT:	Image pixel $s \in \mathcal{I}$, image graph $G = \langle \mathcal{I}, \boldsymbol{\xi}, \omega \rangle$, optimum-path forest P_{sum}, value map V, priority queue Q, label map L, and an array of $status$.

```
1.  $x \leftarrow s$.
2.  do
3.      If $x \in Q$, then Remove $x$ from $Q$
4.      Set $status(x) \leftarrow 1$ , $L(x) \leftarrow 1$ .
5.      For each $y \in \boldsymbol{\xi}(x)$, such that $status(y) \neq 1$, do
6.          Compute $tmp \leftarrow f_\omega(\pi_x \cdot \langle x, y \rangle)$.
7.          If $tmp < V(y)$ and $y \neq Psum(x)$, then
8.              If $y \in Q$, then remove $y$ from $Q$.
9.              Set $V(y) \leftarrow tmp$, $L(y) \leftarrow 1$ .
10.             Insert $y$ in $Q$.
11.     Set $x \leftarrow Psum(x)$.
12. While $(x \neq nil$ and $status(x) \neq 1)$
```

Algorithm 3. – PRUNE_TREE ALGORITHM

INPUT:	The same parameters as in Algorithm 2.
AUXILIARY:	FIFO queue Q_{FIFO}, variable tmp, and x.

```
1.  $x \leftarrow s$.
2.  If $x \in Q$, then Remove $x$ from $Q$
3.  Set $status(x) \leftarrow 1$ , $L(x) \leftarrow 0$, and insert $x$ in $Q_{FIFO}$.
4.  While $Q_{FIFO} \neq \emptyset$
5.      Remove $x$ from $Q_{FIFO}$ .
6.      For each $y \in \boldsymbol{\xi}(x)$, such that $status(y) \neq 1$, do
7.          If $Psum(y) = x$, then
8.              insert $y$ in $Q_{FIFO}$
9.              If $y \in Q$, then remove $y$ from $Q$
10.             Set $status(y) \leftarrow 1$, $L(y) \leftarrow 0$
11.         Else
12.             Compute $tmp \leftarrow f_\omega(\pi_x \cdot \langle x, y \rangle)$.
13.             If $tmp < V(y)$, then
14.                 If $y \in Q$, then remove $y$ from $Q$.
15.                 Set $V(y) \leftarrow tmp$, $L(y) \leftarrow 0$ .
16.                 Insert $y$ in $Q$.
```

Figure 26 shows an example of image segmentation using the GSC–IFT algorithm (Algorithm 1).

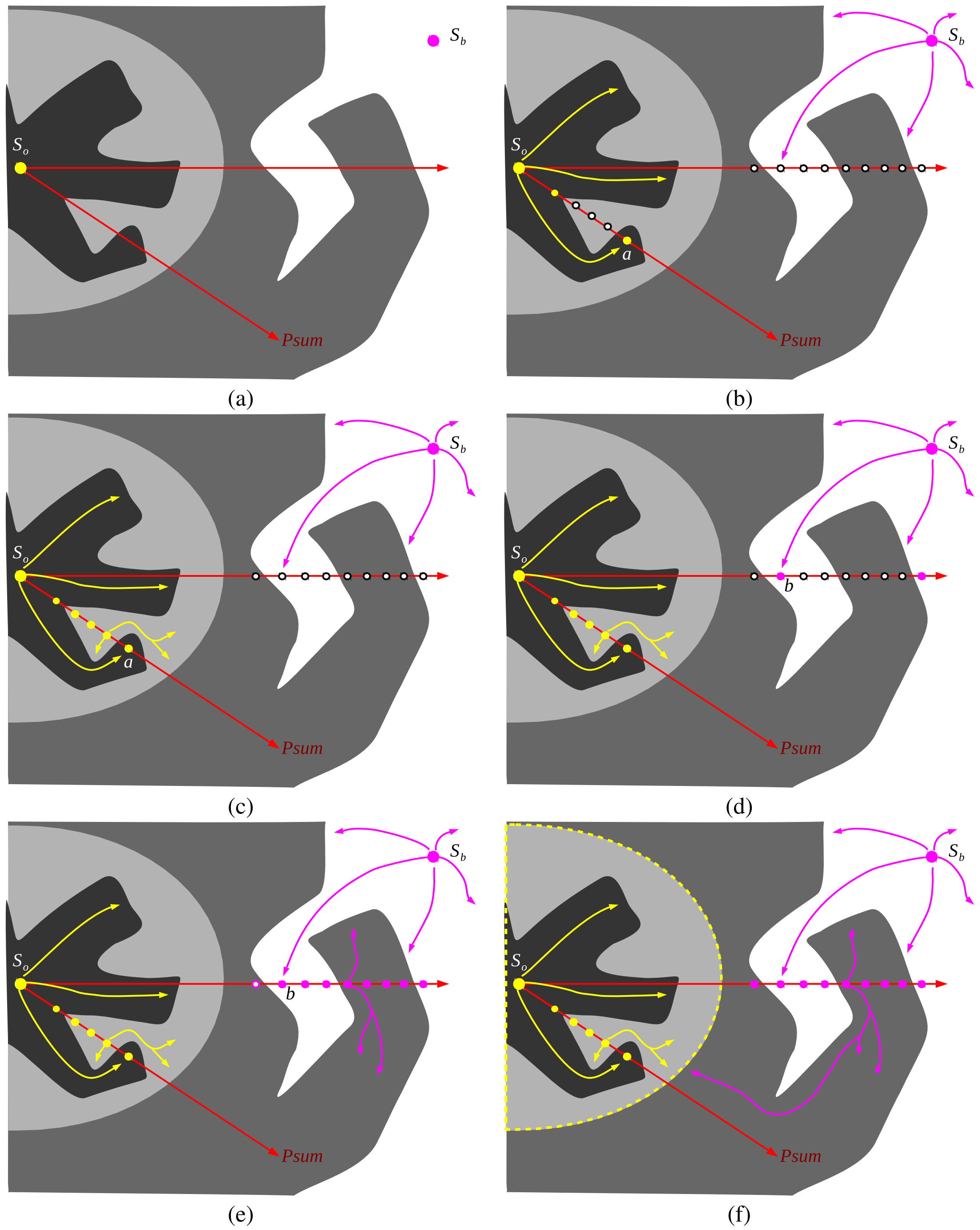


Fig. 26. (a) Synthetic image with three possible boundary cuts, where s_o e s_b are seeds of the object and background, respectively; and the forest P_{sum} permits the visibility of the shape constraints. The target object has a regular shape. (b) After the pixel a leaves of the queue Q with label 1, we have the violation of visibility of the pixel a respect to its closest seed s_o. (c) The subroutine *ConquerPath* (Algorithm 2) assigns label 1 to all his predecessors, allowing the expansion of the object paths in light gray region. (d) The pixel b leaves of the queue Q with label of the background, blocking the visibility of their descendants in P_{sum} respect to s_o. (e) The subroutine *PruneTree* (Algorithm 3) assigns label 0 to their descendants, allowing the expansion of the background paths in the dark gray region. (f) Segmentation result (dashed yellow line) with the GSC–IFT algorithm corresponds with the characteristic of the target object.

In Figures 27 and 28 we present examples of the talus and breast segmentation, respectively. Figure 29 shows an example of 3D segmentation from user-selected markers in MR-T1 images, where the graph's nodes are the voxels, and the arcs are defined between 6-neighbors. Figure 30 shows some results for colored images, with the arc weights computed as in [24]. It is quite clear the advantages of considering the GSC–IFT method.

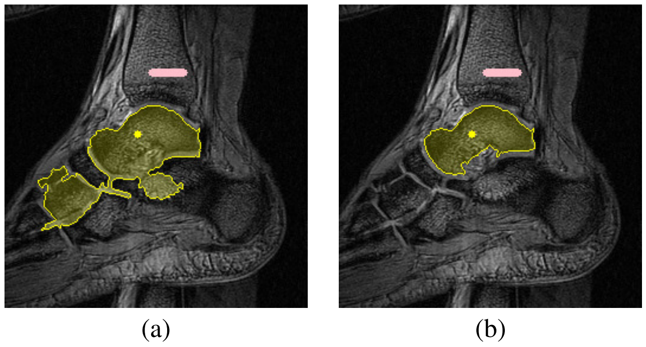


Fig. 27. Segmentation of the talus in a MR image of a foot for the selected markers: (a) IRFC. (b) An improved result by GSC-IFT ($\beta = 0.5$).

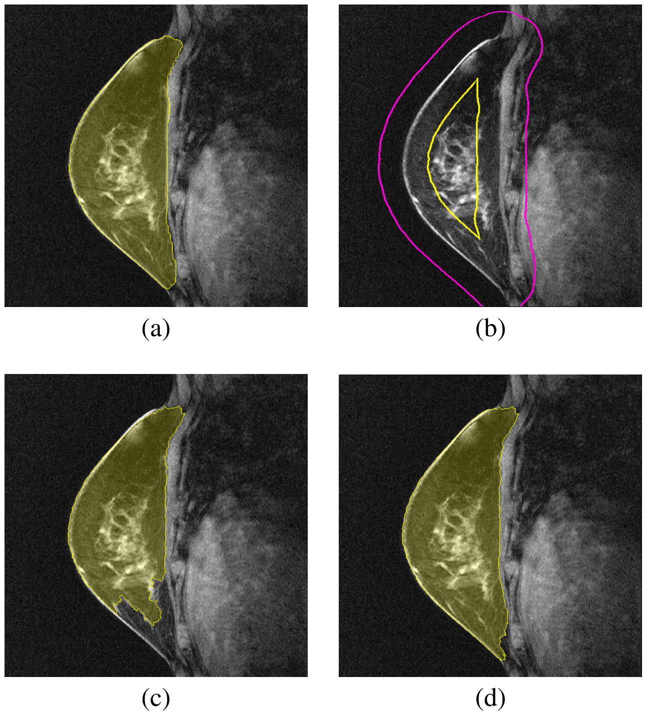


Fig. 28. (a) True segmentation of the breast in MRI. (b) Example of seed sets obtained by eroding and dilating the true segmentation. (c) Segmentation result by IRFC. (d) An improved result is obtained by exploiting the Geodesic Star Convexity (GSC-IFT).

Later, in [28] we proposed the novel method called ***OIFT with Geodesic Star Convexity*** (GSC–OIFT), which incorporate Gulshan's geodesic star convexity prior in the OIFT approach for interactive image segmentation, in order to simultaneously handle boundary polarity and shape constraints (Theorem 2). This method permits the customization of the segmentation by IFT to better match the features of a particular target object (Figure 31). We constrain the search for optimum result, that

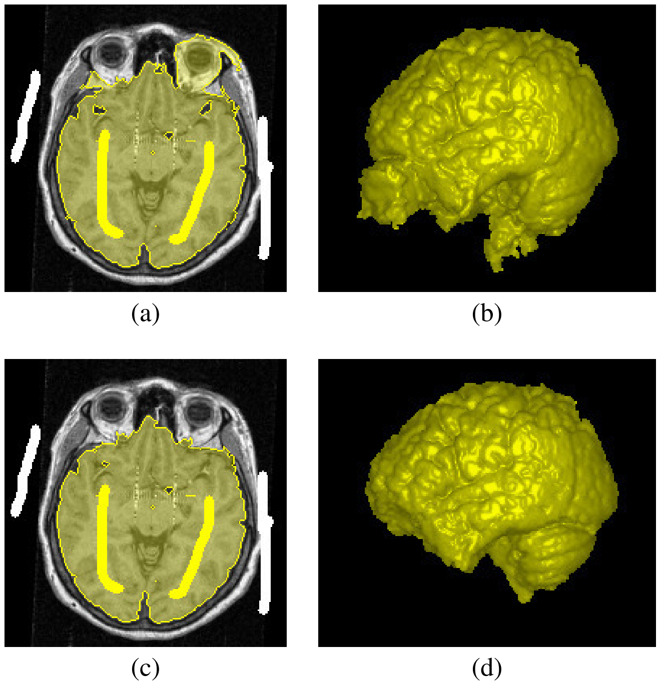


Fig. 29. Example of 3D skull stripping from user-selected markers. (a-b) Segmentation result by IFT with $f_{\max}$. (c-d) An improved result is obtained by exploiting the Geodesic Star Convexity (GSC–IFT with $\beta = 0.1$).

maximize the graph-cut measures E_i (Equation 14) or E_o (Equation 15), only to segmentations that satisfy the geodesic star convexity constraint. We compute a geodesic forest P_{sum} for f_{sum} (Equation 2) by the regular IFT algorithm, using only $\boldsymbol{\mathcal{S}_o}$ as seeds, for the given digraph G, obtaining two sets of arcs $\boldsymbol{\xi^i_{P_{sum}}}$ (Equation 22) and $\boldsymbol{\xi^o_{P_{sum}}}$ (Equation 23). The GSC constraint is violated when $\boldsymbol{\mathcal{C}_i}(L) \cap \boldsymbol{\xi^i_{P_{sum}}} \neq \emptyset$ or $\boldsymbol{\mathcal{C}_o}(L) \cap \boldsymbol{\xi^o_{P_{sum}}} \neq \emptyset$ (Figure 32).

Theorem 2 (Inner/outer-cut boundary optimality)**.** *For a given image graph $G = \langle \boldsymbol{\mathcal{V}}, \boldsymbol{\xi}, \omega \rangle$, consider a modified weighted graph $G' = \langle \boldsymbol{\mathcal{V}}, \boldsymbol{\xi}, \omega' \rangle$, with weights $\omega'(s,t) = -\infty$ for all $(s,t) \in \boldsymbol{\xi^o_{P_{sum}}}$, and $\omega'(s,t) = \omega(s,t)$ otherwise. For two given sets of seeds $\boldsymbol{\mathcal{S}_o}$ and $\boldsymbol{\mathcal{S}_b}$, the segmentation computed over G' by the IFT algorithm for function $f^{bkg}_{o,\max}$ defines an optimum cut in the original graph G, that maximizes $E_o(L,G)$ among all possible segmentation results satisfying the shape constraints by the geodesic star convexity, and the seed constraints. Similarly, the segmentation computed by the IFT algorithm for function $f^{bkg}_{i,\max}$, over a modified graph $G' = \langle \boldsymbol{\mathcal{V}}, \boldsymbol{\xi}, \omega' \rangle$; with weights $\omega'(s,t) = -\infty$ for all $(s,t) \in \boldsymbol{\xi^i_{P_{sum}}}$, and $\omega'(s,t) = \omega(s,t)$ otherwise; defines an optimum cut in the original graph G, that maximizes $E_i(L,G)$ among all possible segmentation results satisfying the shape constraints by the geodesic star convexity.*

$$\boldsymbol{\xi^i_{P_{sum}}} = \{(s,t) \in \boldsymbol{\xi} \mid s = P_{sum}(t)\} \tag{22}$$

$$\boldsymbol{\xi^o_{P_{sum}}} = \{(s,t) \in \boldsymbol{\xi} \mid t = P_{sum}(s)\} \tag{23}$$

Figure 33 shows an example of the GSC–OIFT method with the non-smooth function $f^{bkg}_{o,\max}$.

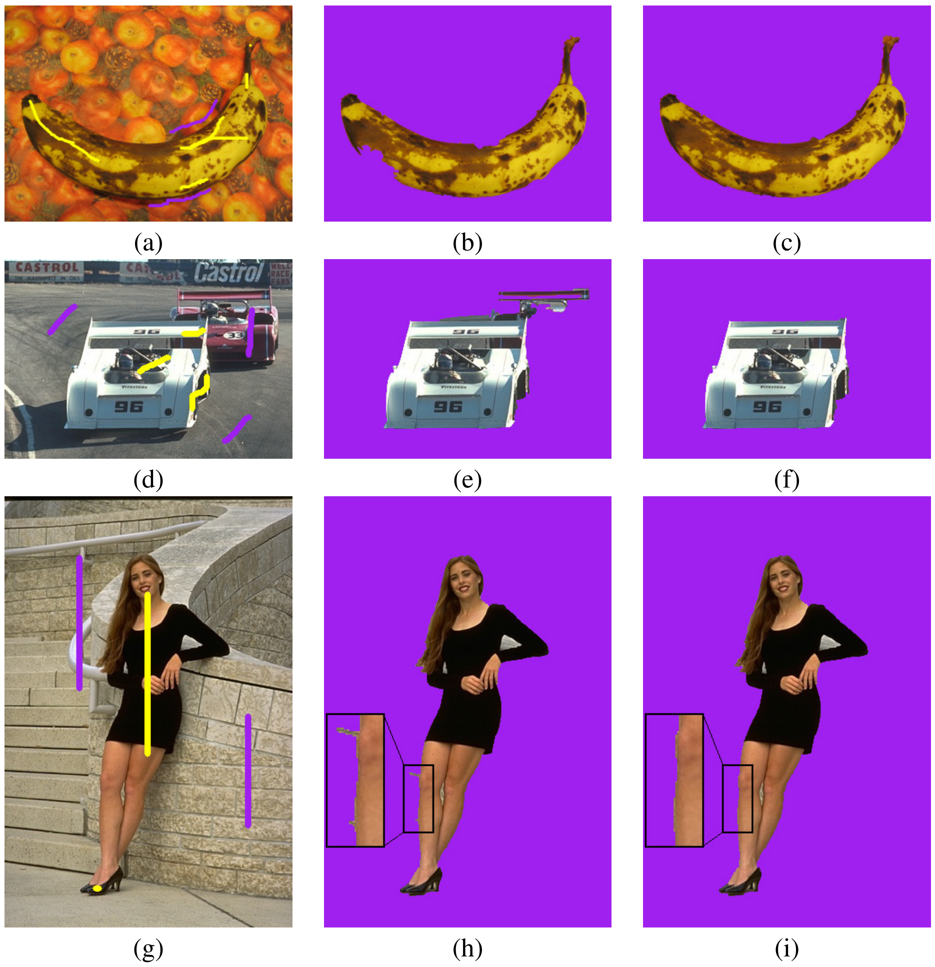


Fig. 30. (a,d,e) Input image with user-selected markers. (b,e,f) Segmentation result by IFT with $f_{\max}$. (c,f,i) Segmentation result by GSC–IFT ($\beta = 0.1$).

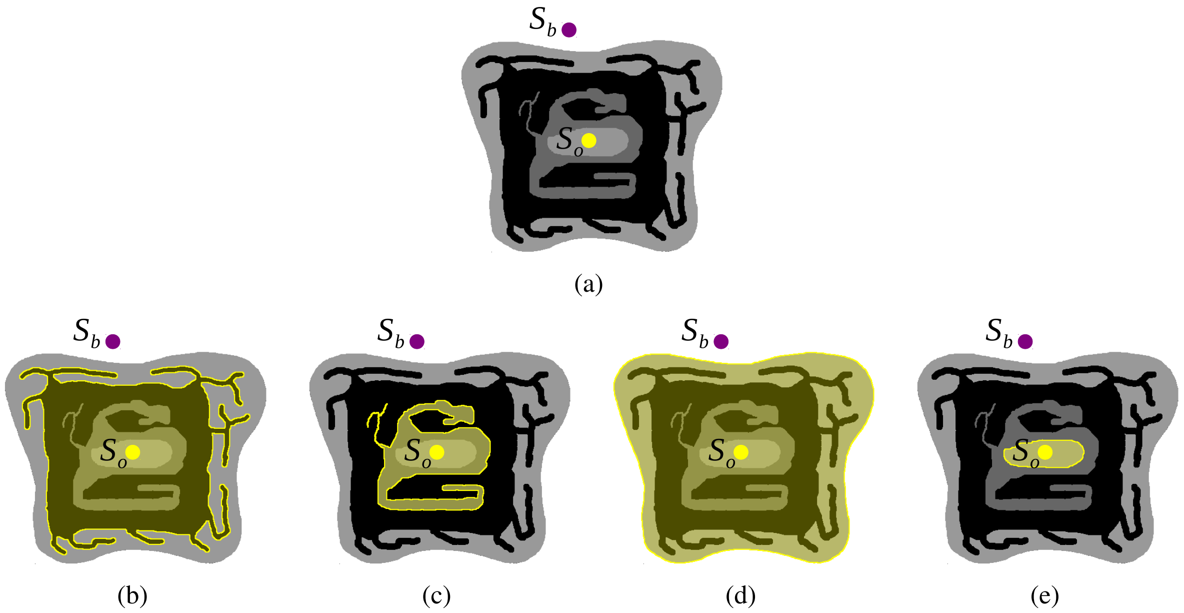
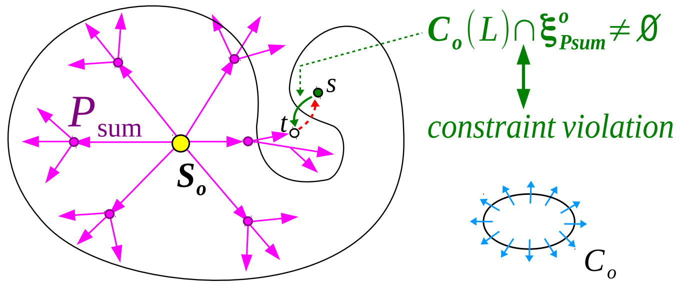
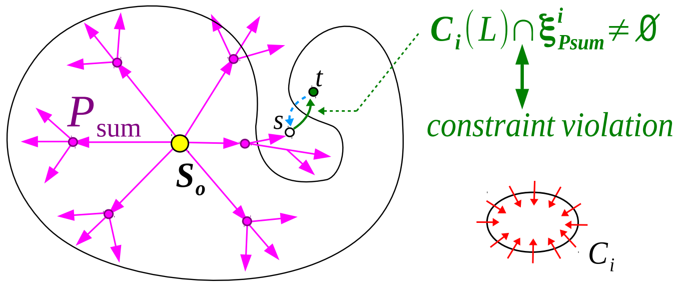


Fig. 31. (a) Synthetic image with selected markers $\boldsymbol{\mathcal{S}_o}$ and $\boldsymbol{\mathcal{S}_b}$. The target object has a regular shape with transitions from bright to dark in its border. Segmentation results by: (b) IFT obtains a non-regular shape and wrong orientation, (c) $OIFT^{\max}_{outer}$ obtains a non-regular shape, (d) GSC–IFT obtains a wrong orientation and (e) GSC–$OIFT^{\max}_{outer}$ (simultaneously considering boundary polarity and shape constraints) obtains a correct matching with the characteristics of the target object.

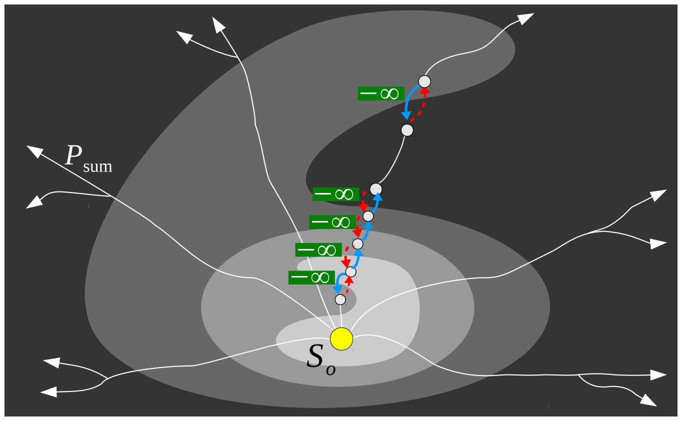


(a)

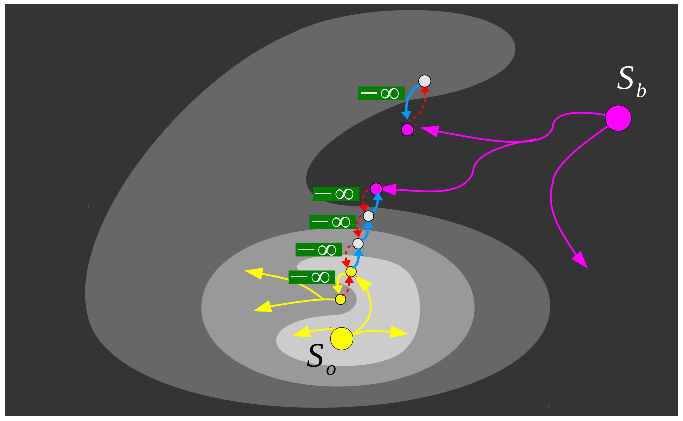


(b)

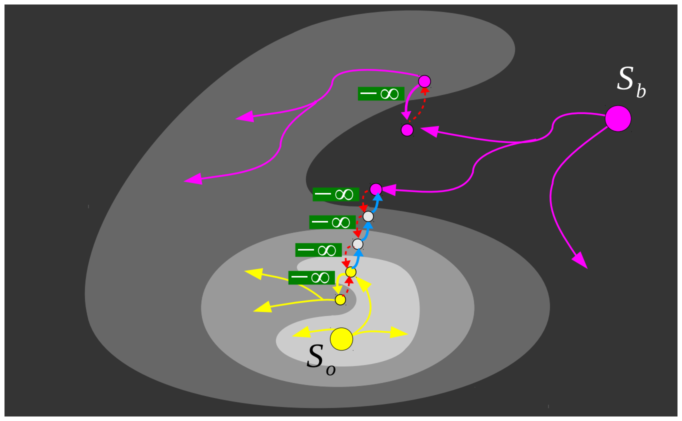
Fig. 32. The GSC constraint is violated when: (a) there is an arc $(s,t)\in\boldsymbol{\mathcal{C}_i}(L)\cap\boldsymbol{\xi}^{\boldsymbol{i}}_{\boldsymbol{P_{sum}}}$, or (b) there is an arc $(s,t)\in\boldsymbol{\mathcal{C}_o}(L)\cap\boldsymbol{\xi}^{\boldsymbol{o}}_{\boldsymbol{P_{sum}}}$.



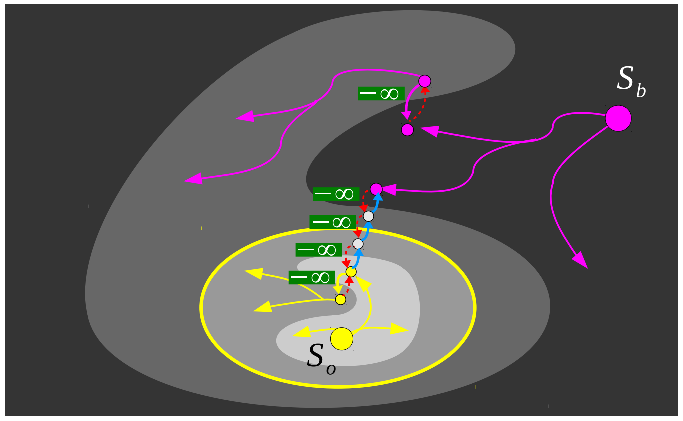
(a)



(b)



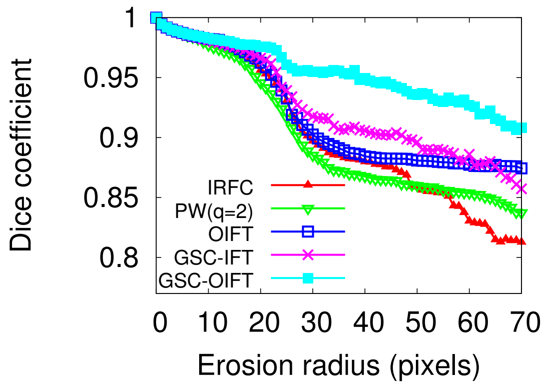
(c)



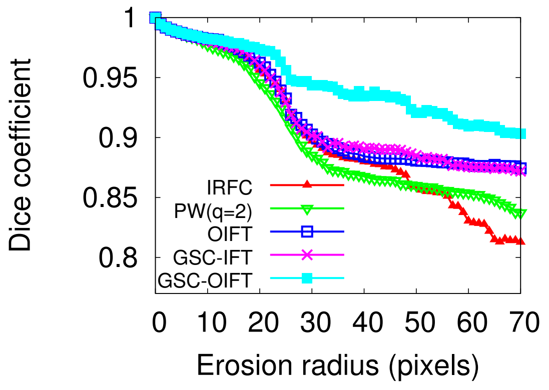
(d)

Fig. 33. (a) Synthetic image with three possible boundary cuts with same orientation (bright to dark), forest P_{sum} and internal seed s_o (star center). The arcs with value $-\infty$ represent arcs $(s,t)\in\boldsymbol{\xi}^{\boldsymbol{o}}_{\boldsymbol{P_{sum}}}$, which violate the shape constraints by the GSC. The blue arrows represent outer cuts in the boundaries and the red (dotted arrows) inner cuts. (b) Pixels are conquered by the internal seed s_o. (c) Pixels are conquered by the external seed s_b. (d) Segmentation result (yellow line) by GSC–OIFT with $f^{bkg}_{o,\max}$.

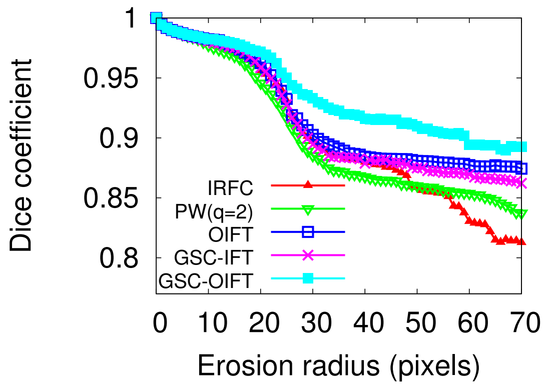
In our experiments, we used 40 image slices of 10 thoracic CT studies to segment the liver. Figure 34a shows the mean accuracy curves for all the images assuming different seed sets obtained by eroding and dilating the ground truth. Note that for higher values of β, GSC–OIFT imposes less shape constraints, so that the accuracy tends to decrease (Figures 34b-d). Figure 35 shows some results in the case of user-selected markers for the liver, and Figure 36 shows one example in 3D.



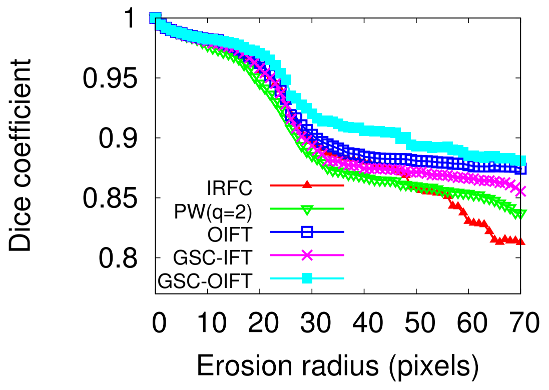
(a)



(b)

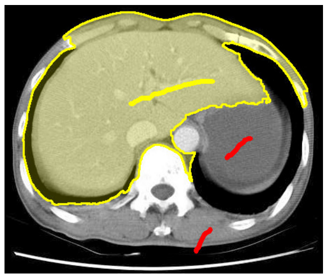


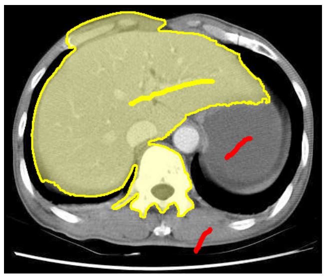
(c)

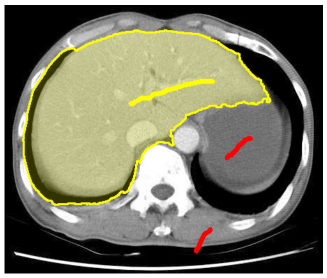


(d)

Fig. 34. The mean accuracy curves of all methods for the liver segmentation for various values of β: (a) $\beta = 0.0$, (b) $\beta = 0.2$, (c) $\beta = 0.5$, and (d) $\beta = 0.7$.


(a)


(b)


(c)

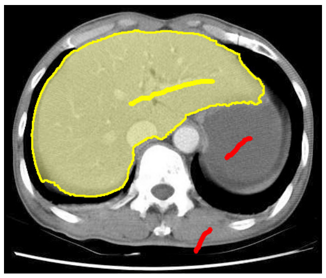
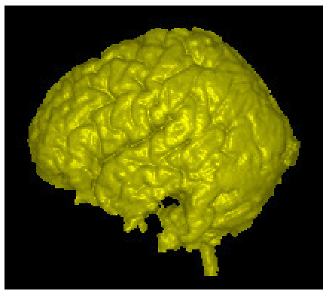
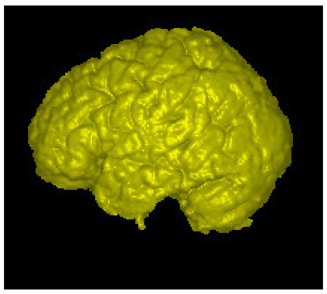

(d)

Fig. 35. Results for user-selected markers: (a) IRFC, (b) OIFT ($f^{bkg}_{o,\max}$ with $\alpha = 0.5$), (c) GSC–IFT ($\beta = 0.7$), and (d) GSC–OIFT ($\beta = 0.7$, $\alpha = 0.5$).


(a)


(b)

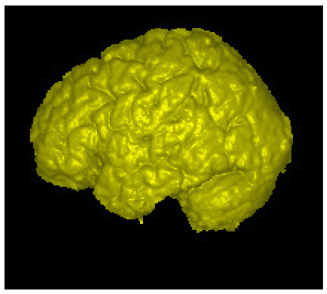

(c)

Fig. 36. Example of 3D skull stripping in MRI: (a) IRFC (IFT with $f_{\max}$), (b) GSC-IFT ($\beta = 0.3$, $\alpha = 0.0$), and (c) GSC-OIFT ($\beta = 0.3$, $\alpha = 0.5$), for the same user-selected markers.

VII. Conclusion

The proposed extension GSC–OIFT includes the IFT with $f_{\max}$, OIFT and GSC–IFT as particular cases, depending on the configuration of its parameters α and β. Note that the adaptive functions presented in Section IV can't be reduced to a GSC–OIFT computation. Table II presents an useful classification of the proposed methods in the master's dissertation [30], according to the specific image characteristics.

The theoretical foundation proposed in this work has also allowed new achievements that were recently published, such as [41] and [42]. The project also contributed in a FINEP project (1266/13) in biomedical engineering, CNPq project (486083/2013-6), and FAPESP project (2012/06911-2).

As future work, we intend to combine the proposed methods with statistical models to automatically define seeds for automatic segmentation.

Acknowledgment

The author thank Dr. Paulo A. V. Miranda (IME–USP) for his supervision, advice and guidance during the whole master's study, Dr. J.K. Uduapa (MIPG–UPENN) for the images, FAPESP grant #2011/50761-2 #2012/06911-2, CNPq (305381/2012-1, 486083/2013-6), CAPES, NAP eScience - PRP - USP.

References

[1] M. Sonka, R. Boyle, and V. Hlavac, *Image Processing, Analysis and Machine Vision*. ITP, 1999.

[2] P. Suetens, *Fundamentals of Medical Imaging*. Cambridge University Press, 2009.

[3] A. Madabhushi and J. Udupa, "Interplay between intensity standardization and inhomogeneity correction in mr image processing," *IEEE TMI*, vol. 24, no. 5, pp. 561–576, 2005.

[4] X. Bai and G. Sapiro, "Distance cut: interactive segmentation and matting of images and videos," in *IEEE International Conference on Image Processing (ICIP)*, vol. 2, San Antonio, Texas, 2007, pp. II – 249–II – 252.

[5] J. Wang and M. Cohen, "An iterative optimization approach for unified image segmentation and matting," in *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 936–943.

[6] A. Falcão, J. Udupa, S. Samarasekera, S. Sharma, B. Hirsch, and R. Lotufo, "User-steered image segmentation paradigms: Live-wire and live-lane," *Graphical Models and Image Processing*, vol. 60, no. 4, pp. 233–260, Jul 1998.

[7] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, 2006.

[8] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.

[9] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Thinnings, shortest path forests, and topological watersheds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 925–939, 2010.

[10] E. Mortensen and W. Barrett, "Interactive segmentation with intelligent scissors," *Graphical Models and Image Processing*, vol. 60, pp. 349–384, 1998.

[11] P. Saha and J. Udupa, "Relative fuzzy connectedness among multiple objects: theory, algorithms, and applications in image segmentation," *Computer Vision and Image Understanding*, vol. 82, no. 1, pp. 42–56, 2001.

[12] W.Yang, J.Cai, J. Zheng, and J. Luo, "User-friendly interactive image segmentation through unified combinatorial user inputs," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2470–2479, Sep 2010.

[13] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, Jun 1991.

[14] J. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundamenta Informaticae*, vol. 41, pp. 187–228, 2000.

[15] L. Grady, "Random walks for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[16] K. Ciesielski, J. Udupa, P. Saha, and Y. Zhuge, "Iterative relative fuzzy connectedness for multiple objects with multiple seeds," *Computer Vision and Image Understanding*, vol. 107, no. 3, pp. 160–182, 2007.

[17] Y. Boykov and M. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *International Conference on Computer Vision (ICCV)*, vol. 1, 2001, pp. 105–112.

[18] A. Falcão, J. Stolfi, and R. Lotufo, "The image foresting transform: Theory, algorithms, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 19–29, 2004.

[19] V. Vezhnevets and V. Konouchine, ""growcut" - interactive multi-label N-D image segmentation by cellular automata," in *Proc. Graphicon.*, 2005, pp. 150–156.

[20] A. Sinop and L. Grady, "A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm," in *11th International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2007.

[21] P. Miranda and A. Falcão, "Elucidating the relations among seeded image segmentation methods and their possible extensions," in *XXIV Conference on Graphics, Patterns and Images (SIBGRAPI)*. Maceió, AL: Los Alamitos: IEEE Computer Society, Aug 2011.

[22] K. Ciesielski, J. Udupa, A. Falcão, and P. Miranda, "A unifying graph-cut image segmentation framework: algorithms it encompasses and equivalences among them," in *Proceedings of SPIE on Medical Imaging*, vol. 8314, San Diego, California, USA, 2012.

[23] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watersheds: A unifying graph-based optimization framework," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, 2010.

[24] P. Miranda, A. Falcão, and J. Udupa, "Synergistic arc-weight estimation for interactive image segmentation using graphs," *Computer Vision and Image Understanding*, vol. 114, no. 1, pp. 85–99, 2010.

[25] P. Miranda and L. Mansilla, "Oriented image foresting transform segmentation by seed competition," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 389–398, Jan 2014.

[26] L. Mansilla and P. Miranda, "Image segmentation by oriented image foresting transform: Handling ties and colored images," in *18th International Conference on Digital Signal Processing (DSP)*. Santorini, Greece: IEEE, Jul 2013, pp. 1–6.

[27] L. Mansilla, M. Jackowski, and P. Miranda, "Image foresting transform with geodesic star convexity for interactive image segmentation," in *IEEE International Conference on Image Processing (ICIP)*, Melbourne, Australia, Sep 2013, pp. 4054–4058.

[28] L. Mansilla and P. Miranda, "Image segmentation by oriented image foresting transform with geodesic star convexity," in *Computer Analysis of Images and Patterns (CAIP)*, vol. 8047, York, UK, Aug 2013, pp. 572–579.

[29] L. Mansilla, F. Cappabianco, and P. Miranda, "Image segmentation by image foresting transform with non-smooth connectivity functions," in *XXVI Conference on Graphics, Patterns and Images (SIBGRAPI)*. Arequipa, Perú: IEEE, Aug 2013, pp. 147–154.

[30] L. Mansilla, "Image foresting transform with non-smooth connectivity functions: Adaptive weights, boundary polarity, and shape constraints," Master's thesis, Institute of Mathematics and Statistics, University of São Paulo, Brazil, Feb 2014. [Online]. Available: http://www.teses.usp.br/teses/disponiveis/45/45134/tde-17032014-121734/pt-br.php

[31] P. Miranda and A. Falcão, "Links between image segmentation based on optimum-path forest and minimum cut in graph," *Journal of Math-*

Image characteristics	**Proposed method**
Presence of inhomogeneity effects.	IFT with $f^{lex}_{\Sigma\|\triangle\mathcal{I}\|}$ or $f^{lex}_{\updownarrow}$.
Transitions of pixels in the object boundary from dark to bright without shape constraint.	OIFT with $f^{bkg}_{i,\max}$ ($\alpha >= 0.5$), OIFT with $f_{i,\omega}$ ($\alpha >= 0.5$) or GSC–OIFT with $f_{i,\omega}$ ($\beta >= 1.0$ and $\alpha >= 0.5$).
Transitions of pixels in the object boundary from bright to dark without shape constraint.	OIFT with $f^{bkg}_{o,\max}$ ($\alpha >= 0.5$), OIFT with $f_{o,\omega}$ ($\alpha >= 0.5$) or GSC–OIFT with $f_{o,\omega}$ ($\beta >= 1.0$ and $\alpha >= 0.5$).
Object with strict regular shape (star convex) and transition of pixels in its boundary from dark to bright.	GSC–OIFT with $f^{bkg}_{i,\max}$ ($\beta = 0$ and $\alpha >= 0.5$).
Object with strict regular shape (star convex) and transition of pixels in its boundary from bright to dark.	GSC–OIFT with $f^{bkg}_{o,\max}$ ($\beta = 0$ and $\alpha >= 0.5$).
Object with regular shape (star convex) allowing a certain degree of variability in its shape, and transition of pixels in its boundary from dark to bright.	GSC–OIFT with $f^{bkg}_{i,\max}$ ($\beta > 0$ and $\alpha >= 0.5$).
Object with regular shape (star convex) allowing a certain degree of variability in its shape, and transition of pixels in its boundary from bright to dark.	GSC–OIFT with $f^{bkg}_{o,\max}$ ($\beta > 0$ and $\alpha >= 0.5$).

TABLE II. RELATION OF THE PROPOSED METHODS AND THE IMAGE TYPE TO SEGMENT.

ematical Imaging and Vision, vol. 35, no. 2, pp. 128–142, 2009, doi: 10.1007/s10851-009-0159-9.

[32] K. Ciesielski, J. Udupa, A. Falcão, and P. Miranda, "Comparison of fuzzy connectedness and graph cut segmentation algorithms," in *Proceedings of SPIE on Medical Imaging: Image Processing*, vol. 7962, Orlando, 2011.

[33] P. Miranda, A. Falcão, and T. Spina, "Riverbed: A novel user-steered image segmentation method based on optimum boundary tracking," *IEEE Transactions on Image Processing*, vol. 21, no. 6, pp. 3042–3052, 2012, doi:10.1109/TIP.2012.2188034.

[34] A. Falcão, F. Bergo, F. Favretto, G. Ruppert, P. Miranda, and F. Cappabianco, *Neurociências e epilepsia. Capítulo: Processamento, Visualização e Análise de Imagens Anatômicas do Cérebro Humano*. Plêiade, 2008, vol. 1, série CInAPCe.

[35] P. Miranda, A. Falcão, and J. Udupa, "Cloud bank: A multiple clouds model and its use in MR brain image segmentation," in *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI)*, Boston, MA, 2009, pp. 506–509.

[36] F. Cappabianco, P. Miranda, J. Ide, C. Yasuda, and A. Falcão, "Unraveling the compromise between skull stripping and inhomogeneity correction in 3T MR images," in *Conference on Graphics, Patterns and Images (SIBGRAPI 2012)*, Aug 2012, pp. 1–8.

[37] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman, "Geodesic star convexity for interactive image segmentation," in *Proceedings of Computer Vision and Pattern Recognition*, 2010, pp. 3129–3136.

[38] A. Falcão, J. Udupa, and F. Miyazawa, "An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly," *IEEE Transactions on Medical Imaging*, vol. 19, no. 1, pp. 55–62, Jan 2000.

[39] C. Rother, V. Kolmogorov, A. Blake, and M. Brown, "Image and video editing: Grabcut," http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm.

[40] O. Veksler, "Star shape prior for graph-cut image segmentation," in *European Conference on Computer Vision (ECCV)*, vol. 5304, 2008, pp. 454–467.

[41] C. Braz and P. Miranda, "Image segmentation by image foresting transform with geodesic band constraints," in *IEEE International Conference on Image Processing (ICIP)*, Paris, France, Oct 2014, pp. 4333 – 4337.

[42] H. Bejar and P. Miranda, "Oriented relative fuzzy connectedness: Theory, algorithms, and applications in image segmentation," in *XXVII Conference on Graphics, Patterns and Images (SIBGRAPI)*. Rio de Janeiro, Brazil: IEEE, Aug 2014, pp. 304–311.

Solving the Art Gallery Problem: A Practical and Robust Method for Optimal Point Guard Positioning

Davi Colli Tozoni
Institute of Computing
University of Campinas (Unicamp)
Campinas, SP, Brazil
Email: davi.tozoni@gmail.com

***Abstract*—This Master's thesis focused on studying and developing techniques for optimally solving the Art Gallery Problem (AGP), one of the most investigated problems in Computational Geometry. The AGP, which is a known ℕℙ-hard problem, consists in finding the minimum number of guards sufficient to ensure the visibility coverage of an art gallery represented as a polygon.**

We studied how to apply Computational Geometry concepts and algorithms as well as Integer Programming techniques in order to solve the AGP to optimality. This work culminated in the creation of a new algorithm for the AGP, whose idea is to iteratively generate upper and lower bounds for the problem through the resolution of discretized versions of the AGP.

The algorithm was implemented and tested on more than 2800 instances of different sizes and classes of polygons. The technique was able to solve in minutes more than 90% of all instances considered, including polygons with thousands of vertices, greatly increasing the set of instances for which exact solutions are known. To the best of our knowledge, in spite of the extensive study of the AGP in the last four decades, no other algorithm has shown the ability to solve the AGP as effectively as the one described here. For illustration, in a direct comparison with the algorithm by Kröller et al., considered one of the most promising techniques for the AGP, our method solved almost 32% more instances than its competitor.

In addition, we provide a free version of our code and of our benchmark for download, which is unprecedented in the literature.

***Index Terms*—Art Gallery Problem, Exact Algorithm, Computational Geometry, Combinatorial Optimization, Visibility.**

I. Introduction

During this Master's Degree at the Institute of Computing (Unicamp), a new method was developed for optimally solving the Art Gallery Problem (AGP), one of the most famous and studied problems in Computational Geometry. The original version of the Art Gallery Problem, also known as the AGP with Point Guards, was proposed in 1973 by Klee and consists in answering the following question: how many guards are sufficient to ensure that an art gallery (represented by a polygon) is fully guarded, assuming that a guard's field of view covers 360 degrees as well as an unbounded distance limited only by the walls of the gallery? As an example of an AGP instance, consider the floor plan of the second level of the Brazilian National Museum of natural history and anthropology, in Rio de Janeiro, displayed in Fig. 1[1]. In this example, we see that 25 guards are sufficient to watch over the entire area.

The main motivation in working with the AGP lies in its interdisciplinarity and its complexity. The AGP can be defined as a Computational Geometry problem, but it is at the same time an Optimization problem. In addition, despite being a theoretical problem, the endless possibilities of relaxing and restricting the initial AGP constraints allow us to create a number of variations, some of which have important practical applications. For example, by limiting the visibility range of the guards and forcing connectivity between them, we have a problem similar to what is found when placing nodes in a sensor network.

Another interesting fact in working with the AGP is the challenge in pursuing exact solutions. Until the first decade of 2000, the main achievements in the study of the AGP were on the theoretical side of the problem. As early as 1975, Chvátal shown that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary for covering a simple polygon of n vertices [1]. Roughly a decade later, Lee and Lin [2] proved the ℕℙ-hardness of the *Art Gallery Problem with Vertex Guards* (AGPV), which is the variant where the guards are restricted to the vertices of the polygon. The case of point guards (the original AGP) is also known to be ℕℙ-hard, as proved by Aggarwal et al. in 1988 [3].

On the other hand, before this thesis, despite several decades of extensive investigation on the AGP, including contributions from renowned researchers as O'Rourke, Mitchell, Urrutia, Ghosh and Fekete, all previously published algorithms for the AGP with Point Guards were unable to handle instances with hundreds of vertices and often failed to prove optimality for a significant fraction of the instances tested. As a matter of fact, some experts have claimed at that time that "practical algorithms to find optimal solutions or almost-optimal bounds are not known" for this problem [4].

In this context, our objective was to find a practical and robust method for the AGP that would make it possible to solve complex instances in a reasonable amount of time. To achieve this goal, we decided to approach the problem with Integer Linear Programming (ILP) techniques and to employ the exact method for the AGP with Vertex Guards presented in [5] as a tool. In their work, Couto et al. solved instances of the AGPV with thousands of vertices.

[1]Obtained from `www.museunacional.ufrj.br`.

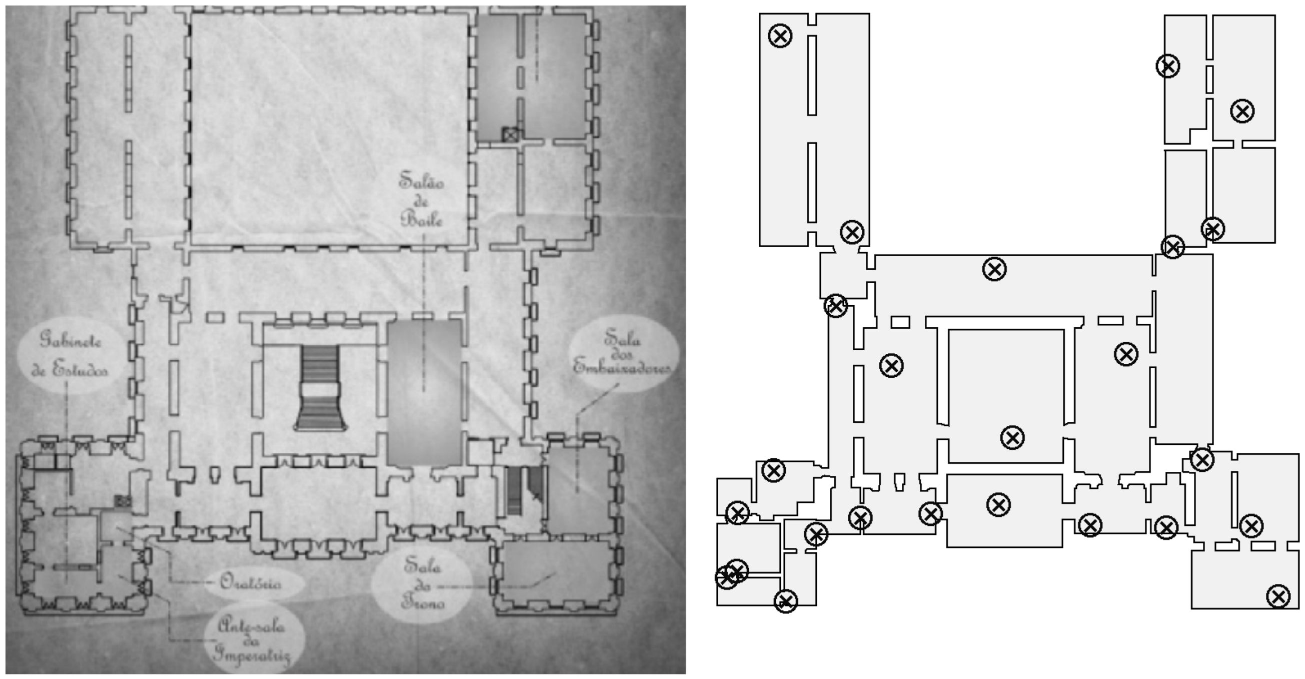


Fig. 1. National Museum floor plan (left); an optimal guard positioning (right).

A. Our contribution

In the thesis, we detailed a new robust method for solving the Art Gallery Problem with Point Guards. The new algorithm, which was presented in papers [6], [7], iteratively solves discretized versions of the AGP making use of ILP solvers to quickly obtain new lower and upper bounds for the problem, until an optimal solution is found or a timeout is reached.

The technique was tested in different occasions and, as seen in Section IV, it led to good results, being considered today the state-of-the-art technique for optimally solving the AGP. In total, 2880 publicly available instances with sizes reaching 5000 vertices were tested and the method achieved an optimality rate of more than 90%, which means a significant improvement over all previously published techniques.

Furthermore, due to the success of our implementation, we have also recently released a free version of our source code in the project's website [8]. This action is a milestone in the quest for practical solvers for the AGP, since, to our knowledge, it is the first time that an implementation with verified efficiency is made publicly available. Hopefully, this will be a catalyst in the search for new techniques for the problem and an incentive for other experimentation projects.

B. Text Organization

This text is organized as follows. The concepts, definitions and theorems necessary to understand the algorithm to be described are presented in the next section. In Section III, the technique we developed is explained. Section IV focuses on how the algorithm was implemented, on the environment and instances used for testing and also on the most significant results obtained, including a comparison with other state-of-the-art techniques. Finally, some conclusions are presented in Section V.

II. Preliminaries

Before digging into the algorithm and its particulars, it is necessary to fully understand some important concepts, being they in the computational geometry field or part of combinatorial optimization. In the next sections, the background related to the Art Gallery Problem and our solver is explained.

A. Computational Geometry

The objective of the AGP is to watch over an art gallery, which may be represented as a simple polygon or as a polygon with holes. A *simple polygon* consists of straight, non-intersecting line segments that are joined pair-wise to form a closed path. The set of vertices V contains all points where consecutive segments are joined. Figures 2 and 3 display examples of simple and non-simple polygons, respectively.

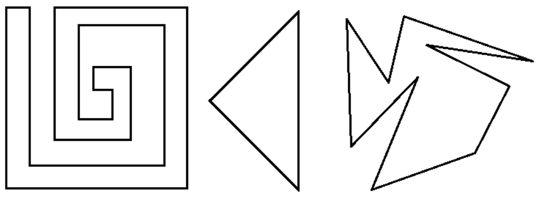


Fig. 2. Simple polygons.

On the other hand, a *polygon with holes* P_h can be described using a simple polygon P_b as the outer boundary and a set of m disjoint simple polygons $H_1, H_2, ..., H_m$ totally contained inside P_b as the holes. In this case, P_h consists in the set $P_b -$

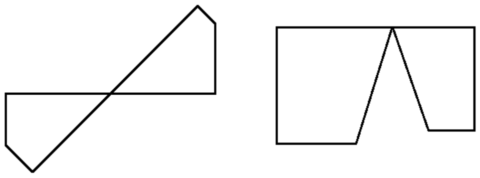


Fig. 3. Non-simple polygons.

$\bigcup_{i=1}^{m} H_i$. Two examples of polygons with holes are displayed in Fig. 4.

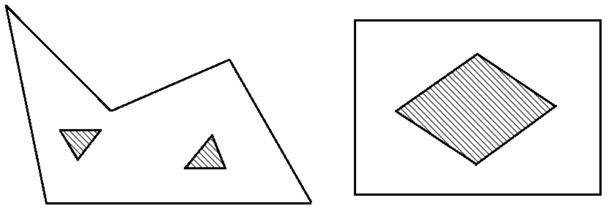


Fig. 4. Polygons with holes.

A vertex in a polygon can also receive a special name depending on the angle between its two incident edges with the interior of the polygon. If this angle is less than 180°, than the vertex is called *convex*. Otherwise, it is a *reflex* vertex. Fig. 5 presents some examples. Note that, in the case of a hole H_i, the convex vertices of H_i are actually reflex in relation to the entire polygon P_h. Obviously, the reverse is also true for reflex vertices.

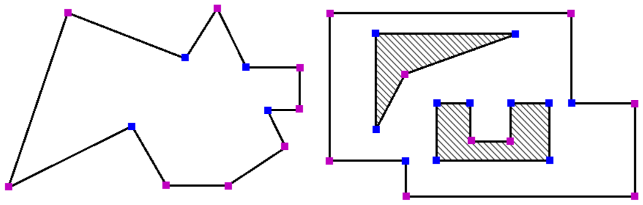


Fig. 5. Convex (purple) and reflex (blue) vertices.

In the AGP, we say that a position is *surveilled* (*watched*, *guarded* or *covered*) by a guard g if this position is visible to g. The concept of visibility can be described as follows: two points in a polygon P are *visible* to each other if the line segment that joins them does not intersect the exterior of P. Thereafter, the *visibility polygon* of a point $p \in P$, denoted by Vis(p), is the set of all points in P that are visible from p. The edges of Vis(p) are called *visibility edges* and one visibility edge is said to be *proper* for p if it is not contained in any of the edges of P. Fig. 6 illustrates the concept of visibility.

After defining visibility, the next step is to define the coverage of a given region. Given a finite set S of points in P, a *covered* (respectively, *uncovered*) region induced by S in P is a maximal connected region in $\bigcup_{p \in S}$ Vis(p) (respectively, $P - \bigcup_{p \in S}$ Vis(p)). Knowing this, we say that S *fully covers* the polygon P if the covered region induced by S in P equals

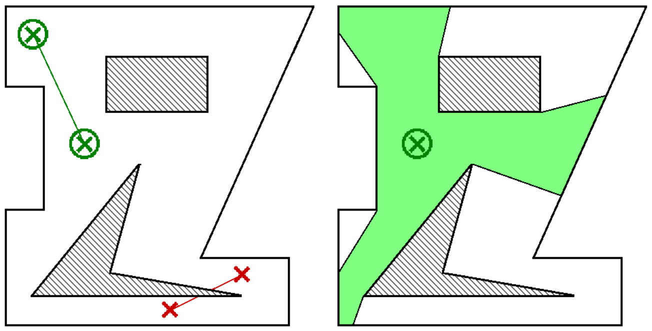


Fig. 6. Two points visible (green) and two other that are not visible (red) to each other (left); the visibility polygon of a point (right).

P. In addition, we can also define $C_{\mathcal{U}}(S)$ as a set containing exactly one interior point of each uncovered region induced by S.

Moreover, the geometric arrangement defined by the visibility edges of the points in S, denoted Arr(S), partitions P into a collection of closed polygonal faces called *Atomic Visibility Polygons* or simply *AVP*s. Clearly, the edges of an AVP are either portions of edges of P or portions of proper visibility edges of points of S. Denote by $C_f \subset S$ the set of points in S that cover an AVP f. Define a partial order $\prec$ on the faces of Arr(S) as follows. Given two AVPs f and f', we say that $f \prec f'$ if and only if $C_f \subset C_{f'}$. We call f a *shadow AVP* (*light AVP*) if f is minimal (maximal) with respect to $\prec$. Applying these concepts, it is possible to define $C_{\mathcal{L}}(S)$ as a set containing exactly one point within each light face of Arr(S) and $V_{\mathcal{L}}(S)$ as the set of all vertices of light AVPs. Figures 7 and 8 illustrate these concepts.

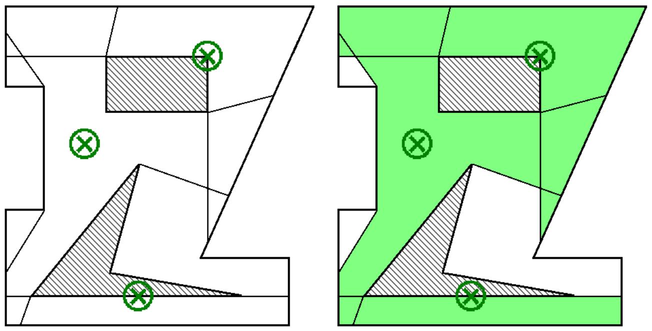


Fig. 7. The arrangement induced by a finite set S of points (left); the set S and its covered (light green) and uncovered (white) regions (right).

In our work, the complexity of the resulting arrangement is of great importance for the solver's performance. In [9], Bose et al. showed that, for the case where the set of vertices V induces the arrangement in a simple polygon, the arrangement complexity is $\Theta(|P|^3)$. This result can be easily adapted to prove that, for a generic set S in a hole-free polygon P, the complexity of constructing Arr(S), as well as the number of AVPs in the arrangement, is $\Theta(|S|^2 \cdot |P|)$. In the general situation, where P can have holes, we did not find

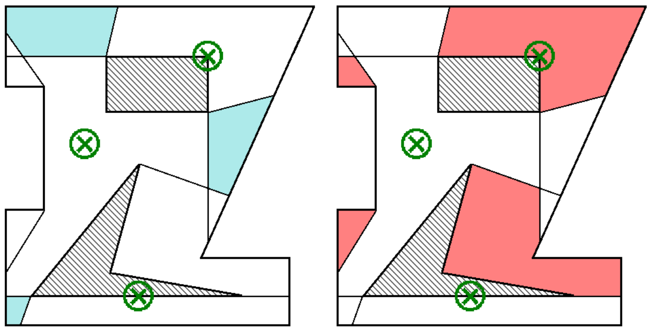


Fig. 8. The arrangement induced by S with the *light* (blue) (left); and *shadow* (red) AVPs (right).

any complexity results in literature. However, as we have $O(|S| \cdot |P|)$ visibility edges in the polygon, in the worst case, considering that all of them intersect, we will have a final complexity of $O(|S|^2 \cdot |P|^2)$.

B. Integer Programming

In 1987, Ghosh presented an approximation technique [10] for the AGP with Vertex Guards in which a reduction of the AGPV to the Set Cover Problem (SCP) is employed. A similar reduction is also used in the method developed in this Master's project.

The SCP is one of the most famous combinatorial problems. Given a set of elements U, called Universe, and a set A containing subsets of U, the SCP asks for the minimum number of sets from A whose union equals U. As proved by Karp in 1972, the Set Cover Problem is $\mathbb{NP}$-complete [11], which means that obtaining an algorithm with polynomial worst-case complexity is not possible, unless $\mathbb{P} = \mathbb{NP}$. Nevertheless, in practice, a good option for solving an SCP instance is to model the problem as an Integer Linear Program (ILP), even though the cost of solving a general ILP is theoretically exponential. Today, several ILP solvers are capable of finding optimal solutions for large SCP instances, with thousands of variables and constraints, in just a few seconds or minutes. Below the classic ILP model for the SCP is given.

$$\begin{aligned} \min \quad & \sum_{s \in A} x_s \\ \text{s.t.} \quad & \sum_{\substack{s \in A \\ e \in s}} x_s \geq 1, \ \forall e \in U \\ & x_s \in \{0,1\}, \ \forall s \in A \end{aligned}$$

In the model, the variable x_s has value 1 if the set s is chosen to be part of the resulting collection of subsets and 0 otherwise. The objective is to minimize the sum of variables x_s, which actually means to minimize the number of sets selected from A. Finally, the set of constraints presented in the model ensures that, for every element $e \in U$, at least one of the subsets containing e is chosen, which gives rise to a viable solution.

Although ILP solvers are normally a good choice for solving SCP instances, there are cases where their use may not be so efficient. In these situations, a technique that can take advantage of particular characteristics of the problem can behave better and be used to improve the ILP solver's performance. In our AGP solver, we implemented some techniques with this purpose. One well tested method to find good viable solutions for SCP instances and that was implemented in this project is a Lagrangian Heuristic.

Apart from the techniques employed to find viable solutions for the SCP, others can be used to simplify the problem, before an actual solver is used. For example, after a problem is reduced to an ILP instance, it is possible to search for redundant variables or constraints and remove them from the original matrix. This type of operation can normally be done quickly and may greatly minimize the size of the initial instance, probably improving the performance of the ILP solver.

Details about the Lagrangian Heuristic and the matrix reduction methods that were implemented in our solver can be seen in the full text of the thesis [12].

C. The Art Gallery Problem

After discussing important geometric and combinatorial concepts, it is now possible to discuss the AGP in a more formal way.

In a geometric setting, the AGP can be restated as the problem of determining a smallest set of points $G \subset P$ such that $\bigcup_{g \in G} \text{Vis}(g)$ equals P. This leads to a reduction from the AGP to the SCP, in which the points in P are the elements to be covered (set U) and the visibility polygons of the points in P are the sets used for covering (which compose the collection A). Accordingly, we can use this reduction to construct an ILP formulation for the AGP:

$$\begin{aligned} \min \quad & \sum_{c \in P} x_c \\ \text{s.t.} \quad & \sum_{\substack{c \in P \\ w \in \text{Vis}(c)}} x_c \geq 1, \ \forall w \in P \\ & x_c \in \{0,1\}, \ \forall c \in P \end{aligned}$$

However, for non-trivial cases, this formulation has an infinite number of constraints and an infinite number of variables, rendering it of no practical benefit. A natural idea that arises is to make at least one of these quantities finite. By fixing only the guard candidates to be a finite set, we obtain the so-called *Art Gallery Problem With Fixed Guard Candidates* (AGPFC). On the other hand, by restricting solely the points that need to be covered (here called witness set) to a finite set, we end up with the special AGP variant known as the *Art Gallery Problem With Witnesses* (AGPW). In principle, in the first case, we are still left with an infinite number of constraints while, in the second case, we still have an infinite number of variables. However, in order to have a tractable SCP instance, one should have both the witness and the guard candidate sets of finite size. The AGP variant that fulfills this property is named

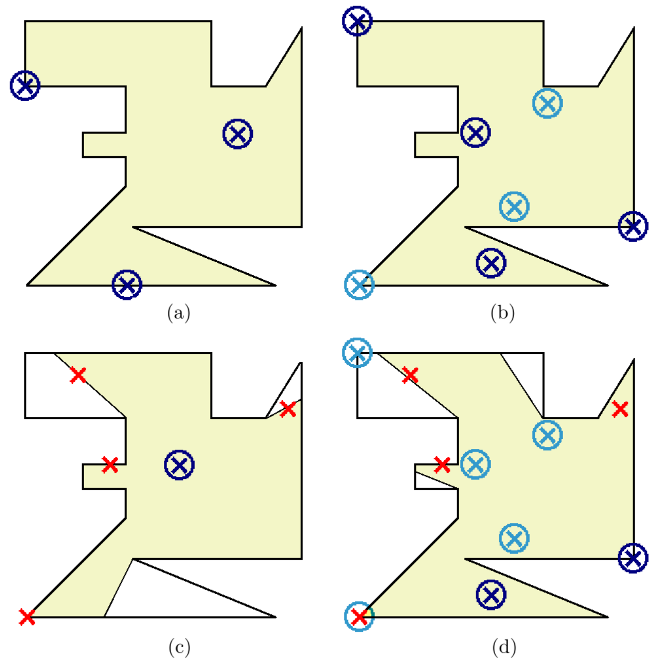


Fig. 9. An illustration of the four different variants of the Art Gallery Problem: (a) AGP; (b) AGPFC; (c) AGPW; (d) AGPWFC.

the *Art Gallery Problem with Witnesses and Fixed Guard Candidates* (AGPWFC). Examples of these three versions of the AGP are shown in Fig. 9. Therein, the witnesses and the guard candidates are identified by the symbols "$\times$" and "$\otimes$", respectively. Darker guard candidates refer to guards present in an optimal solution of the corresponding problem and, when appropriate, have their visibility polygons also depicted.

To assist in the description of the algorithm in the next section, we introduce here some other useful notations. Let D and C be finite witness and guard candidate sets, respectively. We denote the AGPW, AGPFC and AGPWFC problems defined for the sets C and D by AGPW(D), AGPFC(C) and AGPWFC(D, C), respectively. The SCP model associated to AGPWFC(D, C) is then

$$\begin{aligned} \min \quad & \sum_{c \in C} x_c, \\ \text{s.t.} \quad & \sum_{\substack{c \in C \\ w \in \mathrm{Vis}(c)}} x_c \geq 1, \ \forall w \in D, \\ & x_c \in \{0, 1\}, \ \forall c \in C. \end{aligned}$$

D. Basic Theorems

To close this section, we briefly introduce the theorems that form the basis of our algorithmic solution, which will be fully explained in Section III. The following theoretical results allow us to apply reductions of the AGPFC and the AGPW to the AGPWFC (SCP) and also guarantee its usage to find correct bounds for the original AGP. It is noteworthy that Theorems 2, 3 and 4 are actually adaptations of results presented in the work of Couto et al. [5], where the specific problem called AGPV (AGPFC(V)) was treated.

Theorem 1. *Let D be a finite subset of points in P. Then, there exists an optimal solution for AGPW(D) in which every guard belongs to a light AVP of* Arr(D).

Proof: Let G be an optimal (cardinality-wise) set of guards that covers all points in D. Suppose there is a guard g in G whose containing face f in Arr(D) is not a light AVP. This means that f is not maximal with respect to the order relation $\prec$ (see Section II-A). In other words, there exists a face f' of Arr(D) that shares an edge with f such that $f \prec f'$, i.e., a point in f' sees more points of D than one in f does. An inductive argument suffices to show that this process eventually reaches a light AVP (maximal w.r.t. $\prec$) wherein lies a point that sees at least as much of D as g does, i.e., g may be replaced by a guard that lies on a light AVP. The Theorem then follows, by induction, on the number of guards of G. ■

Theorem 2. *Let C be a finite subset of points in P. Consider the set D composed of a point of each AVP of* Arr(C). *Then, $G \subseteq C$ guards D if and only if G is a guard set for P.*

Proof: The necessity part is trivial since $D \subset P$, therefore, we focus on the proof of sufficiency. By the construction process of Arr(C), all interior points of a given AVP A_i are visible to the same set $S_i \in C$. Otherwise, there would be an edge of Arr(C) separating two different points in A_i, which is obviously not possible. Consequently, if a set G guards one interior point of A_i, G directly covers the entire AVP. Thus, since the union of all faces of Arr(C) equals P, G can watch over the whole polygon by simply covering one interior point within each AVP. ■

Theorem 3. *Let C be a finite subset of points in P. Consider the set D composed of a point of each shadow AVP of* Arr(C). *Then, $G \subseteq C$ guards D if and only if G is a guard set for P.*

Proof: The necessity part is trivial since $D \subset P$, therefore, we focus on the proof of sufficiency. Suppose $G \subset C$ guards D, but not P. Thus, there exist regions of P that are not covered by any of the points of G. Let R be a maximal connected region not covered by G. Note that R is the union of (disjoint) AVPs. To prove that at least one of those AVPs is of type shadow, notice that the entire region R is not seen by any point in G whose proper visibility edges spawn R. If R is an AVP, it is by definition a shadow AVP. Otherwise, there is a candidate $c_i \in C$ which has a proper visibility edge e_{ci} that intersects and partitions R in two other regions. One of these regions matches the side of e_{ci} visible from c_i while the opposite one does not. Hence, through an inductive argument, by successively partitioning R, at least one shadow AVP is bound to be contained in R and therefore uncovered. This contradicts the hypothesis since G guards D, which is comprised of interior points of all shadow AVPs. ■

Theorem 4. *Let D and C be two finite subsets of P, so that C fully covers P. Assume that $G(D, C)$ is an optimal solution for AGPWFC(D, C). If $G(D, C)$ also covers P, then $G(D, C)$ is an optimal solution for AGPFC(C).*

Proof: Assume that $G(D, C)$ covers P, but it is not an optimal solution for AGPFC(C). Then, there exists $G' \subseteq C$ with $|G'| < |G(D, C)|$ such that G' is a feasible solution for AGPFC(C), i.e., G' covers P. This implies that G' is also a feasible solution for AGPWFC(D, C), contradicting the fact that $G(D, C)$ is optimal for this problem. ■

III. The Algorithm

The core idea of our algorithm consists in computing increasing lower bounds and decreasing upper bounds for the AGP until a proven optimal solution is found or a pre-established maximum time limit is exceeded. The procedure for obtaining these bounds involves the resolution of discretized versions of the AGP. To find a new lower bound, an AGPW instance is solved, while in the upper bound case, an AGPFC instance in which the guard candidate set covers the polygon is worked out. Observe that an optimal solution for such an instance of the AGPFC is also viable for the AGP, since the AGPFC asks for a solution that guards the entire polygon. Consequently, reducing the gap between bounds to zero means reaching an optimal solution for the original AGP.

In Algorithm 1, we summarize how our technique works. After initializing the witness and guard candidate sets, the algorithm enters its main loop (lines 4 to 10). At each iteration, new lower and upper bounds are computed and, if optimality has not been proved, the witness and guard candidate sets are updated in line 8.

In the following two subsections, the procedures for solving AGPW (line 5) and AGPFC (6) instances are described. After this, Section III-C briefly describes the resolution method for AGPWFC instances through ILP techniques. Both the AGPW and the AGPFC can be reduced to an AGPWFC instance, justifying why we focus on the latter. In Section III-D, we present how the management of the witness set is done and, in the following section, we discuss the selection of guard candidates. Section III-F gathers all the algorithmic information presented previously and describes the complete algorithm for the AGP.

A. Solving the AGPW

The resolution of an AGPW on D allows for the discovery of a new lower bound for the AGP, since fully covering P requires at least as many guards as the minimum sufficient to cover the points in $D \subset P$. However, despite being a simplification of the original AGP problem, we are still dealing with an infinite number of potential guard positions, which does not allow for a direct reduction to the set cover problem. Thus, our approach is based on discretizing the guard candidate set, creating an AGPWFC instance from our original AGPW. To do this, we apply Theorem 1, presented in Section II-D.

From Theorem 1, one concludes that there exists an optimal solution for AGPW(D) in which all the guards are in Light AVPs of the arrangement induced by D. Besides, as every vertex of an AVP can see at least the same set of witnesses observed by the points inside it, we can state that there is

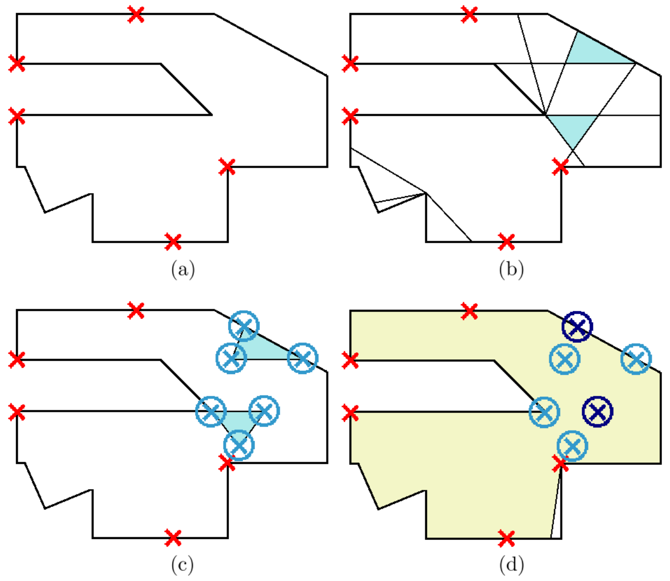


Fig. 10. **Algorithm 2:** (a) AGPW(D); (b) The light AVPs of Arr(D); (c) The guard candidate set; (d) An optimal solution $G \subseteq V_{\mathcal{L}}(D)$ for AGPW(D).

an optimal solution G where each point in G belongs to the set $V_{\mathcal{L}}(D)$ of all vertices from the light AVPs of Arr(D). Therefore, an optimal solution for AGPW(D) can be obtained simply by solving AGPWFC($D, V_{\mathcal{L}}(D)$), as illustrated in the example of Fig. 10. As seen before, the latter problem can be modeled as an ILP, where the numbers of constraints and of variables are polynomial in $|D|$. This follows, as mentioned in Section II-A, from the fact that the number of AVPs (and vertices) in Arr(D) is bounded by a polynomial in $|D|$ and $|P|$. Algorithm 2 shows a pseudo-code of the AGPW resolution method.

As will be shown in Section III-E, the guard candidate set used in the implemented algorithm for the AGP is not actually equal to $V_{\mathcal{L}}(D)$. The final set C includes additional points and, depending on the discretization strategy used, may employ points from $C_{\mathcal{L}}(D)$, which are located in the interior of light faces, instead of the ones in $V_{\mathcal{L}}(D)$.

B. Solving the AGPFC

As we now know how to generate lower bounds for the AGP, the next task is to compute good upper bounds for the problem. A possible way to achieve this is through the resolution of an AGPFC instance in which the guard candidate set is known to cover the polygon. Under this condition, an AGPFC solution is always viable for the original problem.

In contrast to what was discussed regarding the AGPW, we now have a finite number of guard candidates and an infinite number of points to be covered. Therefore, a direct resolution method using a reduction to an SCP is not possible. To circumvent this, our algorithm discretizes the original AGPFC instance, relying on what Theorem 4 establishes. Theorem 4 shows that an optimal solution for the AGPFC may be obtained through the resolution of an AGPWFC instance, provided it fully covers P. Whenever an optimal solution for

Algorithm 1 AGP (Sketch)

```
1: Set UB $\leftarrow |V|$ and LB $\leftarrow 0$
2: Select the initial witness set $D$
3: Select the initial guard candidate set $C \supseteq V$
4: while (UB $\neq$ LB) and (MAXTIME not reached) do
5:     Solve AGPW($D$), set $G_w \leftarrow$ optimal solution of AGPW($D$) and LB $\leftarrow \max\{$LB$, |G_w|\}$
6:     Solve AGPFC($C$), set $G_f \leftarrow$ optimal solution of AGPFC($C$) and UB $\leftarrow \min\{$UB$, |G_f|\}$
7:     if (UB $\neq$ LB) then
8:         Update $D$ and $C$
9:     end if
10: end while
11: return $G_f$
```

Algorithm 2 AGPW(D)

```
1: Arr($D$) $\leftarrow$ construct the arrangement from the visibility polygons of the points in $D$
2: $V_{\mathcal{L}}(D) \leftarrow$ identify the vertices of the light AVPs of Arr($D$)
3: $C \leftarrow V_{\mathcal{L}}(D)$
4: Solve AGPWFC($D$, $C$): set $G_w \leftarrow$ optimal solution of AGPWFC($D$, $C$)
5: return $G_w$
```

the simplified version (AGPWFC) leaves uncovered regions in P, additional work is required. To guarantee that we attain an optimal solution for the AGPFC, we will employ here a technique designed by Couto et al. [5] to solve the AGPV, a special case of the AGPFC where $C = V$, but which may be used to handle the general case without significant changes.

Initially, consider that we have a finite witness set D. Using the guard candidate set C, we can create and solve the AGPWFC(D,C) instance. If the solution fully covers the polygon, we have satisfied the hypothesis of Theorem 4 and, consequently, we have an optimal solution for the AGPFC. Otherwise, there are regions of the polygon that remain uncovered. We now update the witness set, adding new points within the uncovered regions, denoted $C_{\mathcal{U}}(G)$, and repeat the process.

As demonstrated in [5] by Couto et al., the iterative method for solving the AGPFC converges in polynomial time. To clarify this point, consider Theorem 2 and its proof. This theorem states that constructing D by choosing only one point within each AVP of Arr(C) is enough to ensure the whole coverage of P. As the number of AVPs is polynomial (see Section II-A) and we iteratively construct D by choosing witnesses in the interior of uncovered AVPs of Arr(C), it is straightforward that the number of iterations is bounded by the polynomial complexity of Arr(C). Although the convergence time for AGPFC is theoretically guided by this complexity, Couto et al. showed throw extensive experiments that, in practice, the number of necessary iterations is much lower. Moreover, it can even be argued that it suffices to select one point from each shadow AVP of the arrangement induced by C (see Theorem 3). Fig. 11 shows how the algorithm for the AGPFC iteratively adds new witnesses in different AVPs until the polygon is fully covered.

A pseudo-code for the algorithm employed to solve the AGPFC is shown in Algorithm 3.

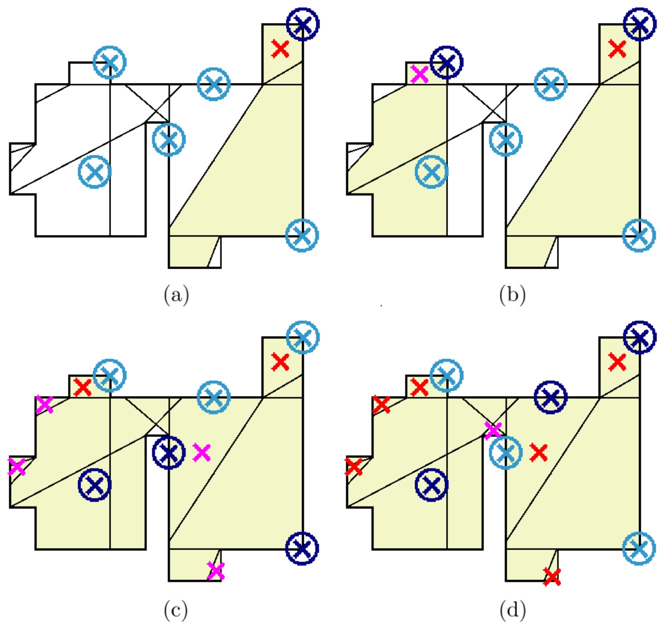


Fig. 11. **Algorithm 3:** A sequence of AGPWFC(D,C) instances is generated until a viable solution for the AGP is obtained.

C. Solving the AGPWFC (SCP)

Having reduced the AGPW and the AGPFC into AGPWFC instances in order to obtain the desired bounds, the objective becomes solving the latter efficiently. Since AGPWFC(D, C) can be easily reduced to an SCP, where the witnesses in D are the elements to be covered and the visibility polygons of the guard candidates in C are the subsets considered, we will make use of the ILP formulation for SCP presented in Section II-B.

Algorithm 3 AGPFC(C)

```
1: $D_f \leftarrow$ initial witness set
2: **repeat**
3:   Solve AGPWFC($D_f, C$): set $G_f \leftarrow$ optimal solution
4:   **if** $G_f$ does not fully cover $P$ **then**
5:     $D_f \leftarrow D_f \cup C_{\mathcal{U}}(G_f)$
6:   **end if**
7: **until** $G_f$ fully covers $P$
8: **return** $G_f$
```

A simple and straightforward approach would be to directly use an ILP solver, such as XPRESS [13], CPLEX [14] or GLPK [15], since even large instances of the ($\mathbb{NP}$-hard) SCP can be solved quite efficiently by many modern integer programming solvers. However, as observed in our experiments, some AGP instances can generate significantly complex and very large SCP instances, rendering the solvers less efficient. For this reason, some known techniques were implemented to improve the solvers' running time. Among these, the most effective consisted in the reduction on the number of constraints and variables in the initial model. Moreover, we also implemented a Lagrangian Heuristic to help the solver obtaining initial viable solutions. Details of these implementations can be seen in the full text of the dissertation [12].

D. Witness Management

The witnesses selected during the execution of our algorithm play an important role in the search for optimal solutions for the AGP. The witness set D is not only decisive for producing good lower bounds through the resolution of AGPW(D), but it is crucial to find tight upper bounds, since the candidate set C of AGPFC(C) is constructed from Arr(D). In this context, choosing D wisely may lead to a lower gap between the bounds and, consequently, to a lower number of iterations of Algorithm 1.

Recall that, before the first iteration of Algorithm 1, an initial witness set is chosen and, in the following ones, it gets suitably updated (line 5). In this section, we present all the initialization alternatives that were considered since our first work [6] and, after this, we discuss how the set D is updated.

The first initialization choice, called *All-Vertices* (AV), consists in using all vertices of the polygon as witnesses, i.e., $D = V$. Besides the easy construction of this set, it was confirmed in tests that the coverage of such points is usually a good start for covering the whole polygon.

In an attempt to begin with a smaller number of witnesses, we also considered initializing D with only the convex vertices of P. This strategy is referred to as the *Convex-Vertices* (CV) initialization. The reason for reducing the initial witness set lies on the fact that a smaller set D is likely to lead to a lower number of visibility polygon calculations, to a less complex visibility arrangement and, consequently, to a simpler SCP model. In addition, we decided to choose only convex vertices because the reflex ones are usually more exposed due to its wider visibility angle.

The third alternative is based on a work by Chwa et al. [16]. In this paper, a polygon P is defined to be *witnessable* when there exists a finite witness set $W \subset P$ with the property that any set of guards that covers W must also cover the entire polygon. The authors also present an algorithm that computes a minimum witness set for a polygon whenever it is witnessable. The method for constructing this minimum witness set consists in placing a witness in the interior of every *reflex-reflex* edge of P and on the convex vertices of every *convex-reflex* edge. The terms *convex* and *reflex* here refer to the interior angles at a vertex or at the endpoints of an edge. Based on these selection criteria, we devised our third discretization method, called *Chwa-Points* (CP), which assembles the initial witness set for our algorithm from the midpoints of all reflex-reflex edges and all convex vertices from convex-reflex edges.

It follows from the results in [16] that, when the Chwa-Points discretization is used for a witnessable input polygon, our AGP algorithm will find an optimal solution in a single iteration. However, as observed in our experiments, non-witnessable polygons are the norm. In fact, among our random benchmark instances, they constitute the vast majority.

Finally, an additional discretization was created based on CP, in an attempt to improve the results previously obtained. In this strategy, called *Chwa-Extended* (CE), besides all points from CP, we also include all reflex vertices of P in the initial discretization.

An example of each one of the four discretization strategies implemented is presented in Fig. 12. Notice that, when characterizing vertices of a hole, the terms *convex* and *reflex* have their meaning inverted.

Let us now focus on the updating process of the witness set throughout the algorithm. This procedure takes place in two different occasions: while solving an AGPFC instance (in line 5 of Algorithm 3) and when jumping to the next iteration of the AGP algorithm (line 8 of Algorithm 1). In the first case, as presented in Section III-B, new witnesses are positioned considering the current solution of AGPWFC(D_f,C). Here, we add to D_f one point placed in the interior of each uncovered region, which (as explained in Section III-B) is enough to guarantee the convergence of the AGPFC resolution method. See Fig. 13 for an example.

On the other hand, a deeper analysis is necessary when dealing with the update procedure of the main algorithm, because, as discussed at the beginning of this section, the

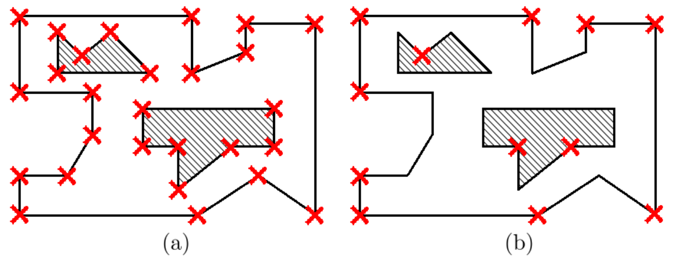
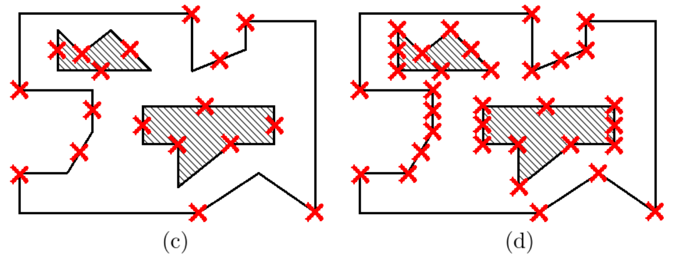



Fig. 12. Examples of the initial set D when using each one of the following discretization strategies: (a) All-Vertices (AV); (b) Convex-Vertices (CV); (c) Chwa-Points (CP); (d) Chwa-Extended (CE).

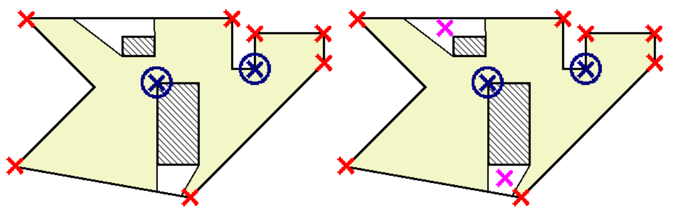


Fig. 13. Solution of an AGPWFC instance (left); New witnesses chosen (violet) for the next iteration of the AGPFC algorithm (right).

selection of D affects all significant parts of the technique. In a summarized form, the better the choice of a new set of points for inclusion into the witness set, the better the lower and the upper bounds obtained would be and, consequently, the faster the convergence.

In essence, the process consists of adding points from the uncovered regions arising from the solution of the previous AGPW instance. Our first attempt was to imitate the strategy adopted by the AGPFC algorithm and to position one new witness inside each uncovered region. However, our experiments later showed that the inclusion of only these points were not sufficient to lead the algorithm to good performance and convergence. The shortfall was traced to the absence of new points on the boundary of the polygon, which proved to be useful to the evolution of the bounds obtained. Therefore, whenever an edge of an uncovered region is found to be contained on the boundary of the polygon, its vertices and its midpoint are also selected to increment the witness set throughout the iterations. These points along with an interior point of each uncovered region form the whole set M of new witnesses. Note that the selection of midpoints in this procedure is arbitrary and, in general, can have a better or worse effect than choosing any other non-extreme point of the segment. Fig. 14 displays an example of how this updated procedure works.

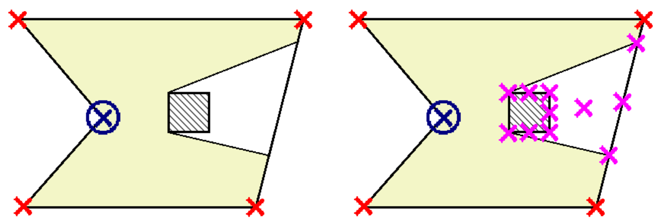


Fig. 14. Solution of an AGPW instance (left); New witnesses chosen (violet) for the next iteration of the AGP algorithm (right).

E. Guard Candidate Management

After constructing the arrangement induced by the set of witnesses D and identifying the corresponding light AVPs, our algorithm is able to build the guard candidate set C. This set must be built in such a way that guarantees that an optimal solution G for AGPW(D) is contained in C. After solving AGPW(D), C is maintained and also used in AGPFC(C), contributing to the discovery of a new upper bound. In this section, we present, in details, how these candidates are selected.

Since our first published work [6], two different discretization strategies for C have been experimented. Both of them follow the idea presented in Theorem 1 and construct C using at least one point from each existing light AVP, thereby ensuring that AGPW(D) is optimally solved. In addition, as using only points from Light AVPs may not guarantee the existence of a solution that covers the entire polygon (a necessary requirement for the AGPFC solvability), both strategies also include all vertices of P ($V \subset C$).

Our first strategy, named *Boundary-Guards* (BG), contains, besides the vertices of P, all points from $V_{\mathcal{L}}(D)$ ($C = V \cup V_{\mathcal{L}}(D)$). This discretization was originally the only method used for choosing C in the two papers that describe our algorithm [6], [7].

However, recall that the arrangement does not grow linearly with the number of witnesses in D. This way, in our experiments with large polygons, the tasks which depend on C, such as the computation of visibility between pairs of points, become increasingly time consuming. For example, in a simple polygon with 5000 vertices, we may have to compute more than 100 million visibility tests between candidates and witnesses in a single iteration.

In this context, a new discretization with a lower number of guard candidates was experimented in [17]. This time, the points from $V_{\mathcal{L}}(D)$ were not included in C. The idea, named *Center-Guards* (CG), was to replace the vertices of a given light AVP by an internal point of it, reducing the number of these candidates by a factor of at least 3. For an example of BG and CG strategies, see Fig. 15.

F. Resulting Algorithm

Once each of the main steps of the algorithm is understood, we are able to describe how these parts fit together to comprise

Algorithm 4 AGP(P)

```
1:  $D \leftarrow$ initial witness set   {see Section III-D}
2:  Set LB $\leftarrow 0$, UB $\leftarrow n$ and $G^* \leftarrow V$
3:  loop
4:    Solve AGPW($D$): set $G_w \leftarrow$ optimal solution and $z_w \leftarrow |G_w|$   {see Section III-A}
5:    if $G_w$ is a coverage of $P$ then
6:      return $G_w$
7:    end if
8:    LB $\leftarrow \max\{\text{LB}, z_w\}$
9:    if LB $=$ UB then
10:     return $G^*$
11:   end if
12:   $C \leftarrow V_{\mathcal{L}}(D) \cup V$ (or $C_{\mathcal{L}}(D) \cup V$)   {see Section III-E}
13:   $U \leftarrow C_{\mathcal{U}}(G_w)$   {one additional point per uncovered region}
14:   $D_f \leftarrow D \cup U$
15:   Solve AGPFC($C$), using $D_f$: set $G_f \leftarrow$ optimal solution and
      $z_f \leftarrow |G_f|$   {see Section III-B}
16:   UB $\leftarrow \min\{\text{UB}, z_f\}$ and, if UB $= z_f$ then set $G^* \leftarrow G_f$
17:   if LB $=$ UB then
18:     return $G_f$
19:   end if
20:   $D \leftarrow D \cup U \cup M$   {see Section III-D}
21: end loop
```

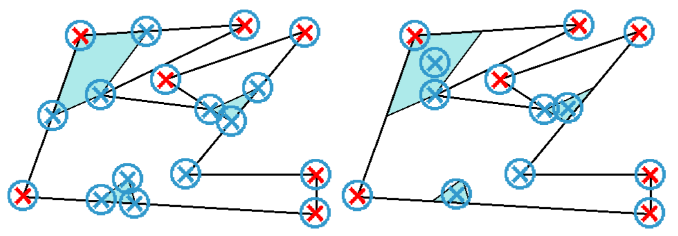


Fig. 15. Examples of the guard candidate set C when using each of the following discretizations: Boundary-Guards (BG) (left); Center-Guards (CG) (right);

the complete algorithm. Algorithm 4 sums up how the process as a whole works.

It is important to notice that the set of guard candidates used in the AGPW resolution is actually the set C from the AGPFC(C) instance solved on Line 15. This means that the AGPW resolution is actually the first iteration of the AGPFC algorithm (Algorithm 3). Thus, all information obtained during the solution of AGPW(D) can be reused for AGPFC(C), which improves the performance of the implementation.

Another relevant aspect of this algorithm is that information on bounds may be used throughout the iterations in order to skip unnecessary steps. For instance, if an upper bound UB was found in a previous iteration and a new AGPFC instance is being solved, whose current solution is not lower than UB, then we may stop the AGPFC resolution before obtaining an optimal solution since the upper bound can not be improved.

IV. Implementation and Computational Results

To verify the quality of the new algorithm described, we coded it in the C++ programming language and used the Computational Geometry Algorithms Library (CGAL) [18] to benefit from visibility operations, arrangement constructions and other geometric tasks. In addition, to exactly solve the AGP discretizations, we employed ILP solvers like XPRESS [13], CPLEX [14] and the free package GLPK [15].

A. Instances

We experimented with AGP instances from 4 sources: [5], [19], [20] and [7]. The instances, whose sizes vary from 20 to 5000 vertices, can be divided into 6 classes of polygons: simple, orthogonal, von Kock, simple-simple (simple with holes), ortho-ortho (orthogonal with holes) and spike. Testing with different polygon classes was crucial to verify the robustness of the technique. In total, we tested our algorithm on 2880 instances and obtained an overall optimality rate of more than 90%. Fig. 16 presents examples of each class of polygons used in our experiments.

B. Evolution of the Implementation

Our implementation went through several versions since its first release. Such changes, which included the employment of new routines, data structures and decisions, substantially improved the performance of the software. Our first version (I1) was completed in late 2012 and was reported in a conference paper [6]. Shortly after that, in the first half of 2013, a second version (I2) was produced and described in a full paper [7]. The latest version (I3) was developed during a two-month internship at the Technische Universität

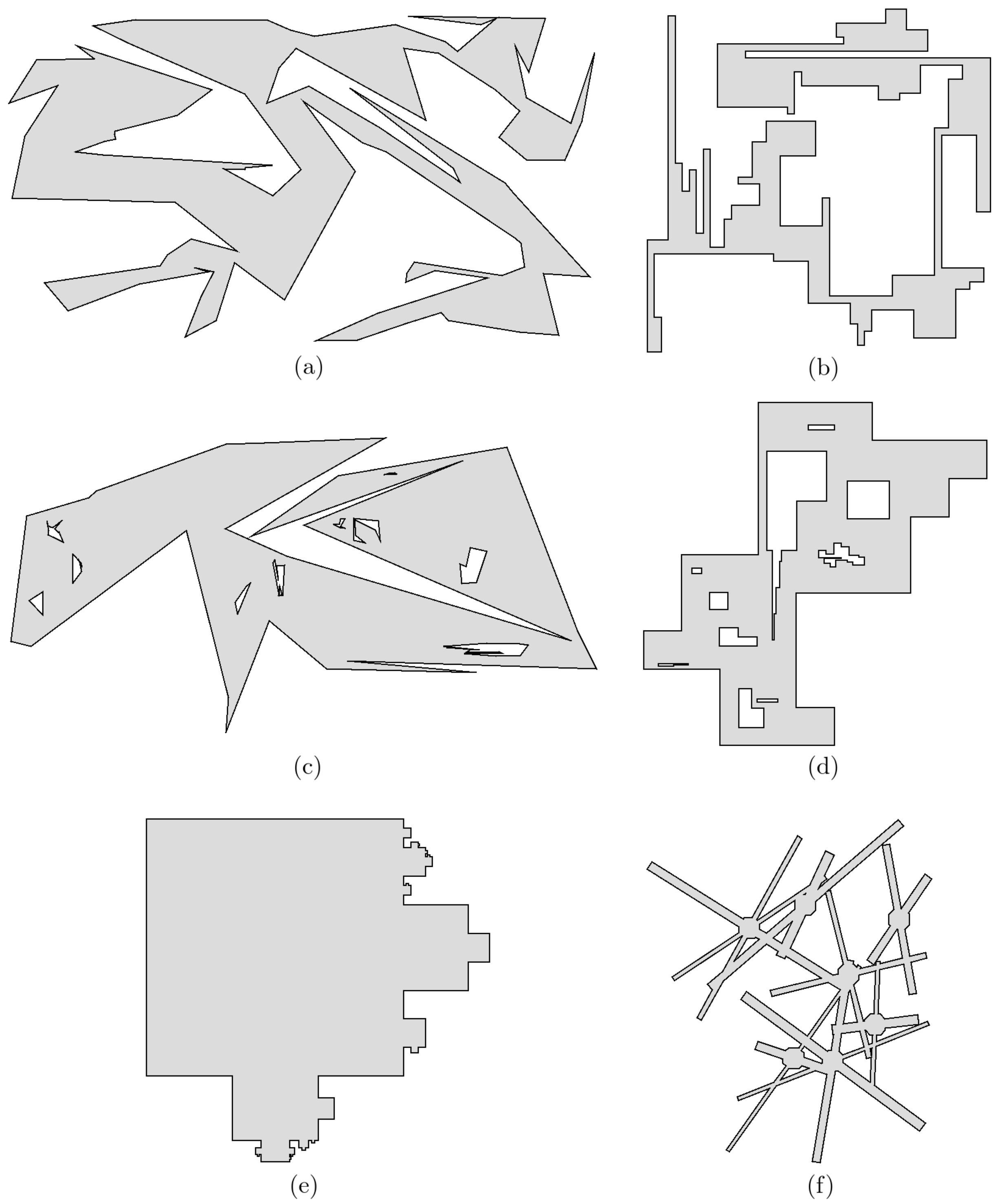


Fig. 16. Examples of instances from different classes. (a) Simple; (b) Orthogonal; (c) Simple-simple; (d) Ortho-ortho; (e) von Koch; (f) Spike.

Braunschweig (TUBS), in Germany, under the supervision of Prof. Dr. A. Kröller. During this visit, the student worked in straight collaboration with researchers from the Algorithms Group headed by Prof. Dr. S. Fekete. Their group has also developed methods for the AGP, including a solution for the original problem [20]. Our new implementation is described in a survey on algorithms for Art Gallery Problems [17], in a joint work by researchers from UNICAMP and TUBS.

During the writing of the AGP Survey [17], we had the chance of experimenting I1, I2 and the new version I3 on 900 instances in a single environment in the laboratory of the Algorithms Group at TUBS. The joint experimentation provided a direct comparison and verification of the improvement occurred during the Master's project. In this occasion, I3 showed to be very successful, being able to optimally solve 768 polygons, including instances with 5000 vertices, in runs of less than 20 minutes. Table I displays the optimality rates achieved by I1, I2 and I3.

TABLE I
OPTIMALITY RATES OF I1, I2 AND I3.

Class	Source	n	Optimality Rate (%)		
			I1	**I2**	**I3**
Simple	From [5]	200	100.00	100.00	100.00
		500	100.00	100.00	100.00
		1000	96.67	100.00	100.00
		2000	6.67	50.00	100.00
		5000	0.00	0.00	100.00
Orthogonal	From [5]	200	100.00	100.00	96.67
		500	100.00	96.67	93.33
		1000	100.00	100.00	100.00
		2000	70.00	90.00	100.00
		5000	0.00	0.00	93.33
Simple-simple	From [17]	200	-	100.00	100.00
		500	-	83.33	100.00
		1000	-	0.00	100.00
		2000	-	0.00	46.67
		5000	-	0.00	0.00
Ortho-ortho	From [7]	200	-	96.67	100.00
		500	-	83.33	100.00
		1000	-	3.33	96.67
		2000	-	0.00	33.33
		5000	-	0.00	0.00
von Koch	From [5]	200	100.00	100.00	100.00
		500	96.67	100.00	100.00
		1000	46.67	100.00	100.00
		2000	0.00	0.00	100.00
		5000	0.00	0.00	0.00
Spike	From [20]	200	-	100.00	100.00
		500	-	100.00	100.00
		1000	-	96.67	100.00
		2000	-	96.67	100.00
		5000	-	0.00	100.00

The results in Table I evince two big steps in our headway. From I1 to I2, besides a considerable improvement in the optimality rate, we became able to solve polygons with holes, greatly increasing the range of treatable classes. Subsequently, from I2 to I3, a remarkable performance improvement was conquered, as evidenced by the resolution of polygons of 5000 vertices. These polygons have twice the size of the previous largest instances already treated by AGP solvers, fact achieved by I2 in [7].

TABLE II
AVERAGE TIME OF I1, I2 AND I3.

Class	Source	n	Average Time (sec)		
			I1	**I2**	**I3**
Simple	From [5]	200	7.31	3.63	0.75
		500	67.81	32.82	2.96
		1000	358.97	158.73	9.18
Orthogonal	From [5]	200	4.10	2.72	0.37
		500	30.06	19.61	1.49
		1000	189.41	111.40	5.22
von Koch	From [5]	200	11.12	3.54	1.20
		500	158.53	31.88	7.80
		1000	767.01	186.49	52.81

In order to confirm this analysis, we collected information about the time necessary to find optimal solutions. Table II shows the average time needed to solve simple, orthogonal and von Koch polygons, considering only instances where optimal solutions were found by all three implementations. From this table, one can see that the average time of I2 can be about 5 times smaller than I1, as verified in results of von Koch polygons with 500 vertices. The difference is even greater when analyzing I2 against I3, which is capable of solving, on average, orthogonal polygons of size 1000 almost 22 times faster than I2.

C. Comparison With Other Techniques

In recent years, other algorithms were proposed for the AGP. In this Master's thesis, we compared our achievements with the two of them that excelled the most. First, we analyzed the differences between our method and the work by Bottino and Laurentini in 2011 [19]. In [19], Bottino and Laurentini proposed a heuristic for the original AGP, aiming to produce good viable solutions with an efficient method. The technique was experimented and obtained promising results, including some optimal solutions. In the paper, the authors compared their technique with the one by Amit et al. [21] and claimed that their method was able to achieve better results.

Upon learning about this work, we decided to try our I2 version with exactly the same instances used by Bottino and Laurentini and compare our findings. The experiments were done using all simple and orthogonal instances from Bottino et al., which vary between 30 and 60 vertices. Table III summarizes the results, showing two types of information: average number of guards and average run time.

In our tests, I2 was able to find proven optimal solutions for all instances, meaning that the column with average number of guards found by our method actually contains optimal values. Knowing this, we can conclude that the heuristic by Bottino and Laurentini was able to find good solutions, but not always

TABLE III
COMPARISON BETWEEN THE METHOD OF BOTTINO ET AL. [19] AND I2.

Class	n	Number of Guards		Time (sec)	
		Method [19]	I2	Method [19]	I2
Simple	30	4.20	4.20	1.57	0.14
	40	5.60	5.55	2.97	0.10
	50	6.70	6.60	221.92	0.24
	60	8.60	8.35	271.50	0.27
Orthogonal	30	4.60	4.52	1.08	0.04
	40	6.10	6.00	9.30	0.07
	50	7.80	7.70	6.41	0.12
	60	9.30	9.10	81.95	0.16

optimal. Except for simple polygons with 30 vertices, the heuristic did not manage to find the best possible solutions for all polygons of a subgroup. Looking more carefully, we can notice a growing gap between the average number of guards from both techniques as the size of the instances increases.

Besides comparing the quality of the solutions, it is also important to evaluate the time needed to find them. To this end, Table III exhibits the computing times for the two methods. It is important to notice though that the experiments were done in different environments, which invalidates a direct comparison of performance between the techniques. While our tests were conducted in machines featuring an Intel® Core™ i7-2600 at 3.40 GHz and 8 GB of RAM, the researchers of [19] performed their experiments on an Intel® Core2™ processor at 2.66 GHz and with 2 GB of RAM. Despite this, Table III shows that the average run time of our technique to compute proven optimal solutions for the AGP is orders of magnitude smaller than the time used by the heuristic. At least it seems safe to say that this large disparity in computing times can not be entirely attributed to hardware and software differences.

Finally, we also performed a complete comparison with the most recent implementation of the algorithm presented in 2012 by Kröller et al. [20], based on results obtained from the survey on algorithms for the AGP [17]. All experiments were performed during the internship at TUBS on the same computing environment, enabling a fair comparison between the two techniques. Considering all experiments, where only polygons of size 200, 500, 1000, 2000 and 5000 were considered, our current implementation (I3) was very successful. I3 was able to optimally solve 768 of 900 polygons, including instances with 5000 vertices, in runs of less than 20 minutes. Meanwhile, Kröller et al. technique (here called BS3) could solve 583 instances. In addition, while our version was able to obtain 100% optimality in 21 out of 30 subgroups of instances considered, the version from TUBS only achieved this for 4 subgroups. Table IV shows the optimality results.

To get a deeper insight into the differences in behavior of the techniques, we also developed a running time comparison between them, using results of all polygon classes. This comparison is shown in Fig. 17. For a fairer analysis, the average times in the charts only considered values of instances resolved by both I3 and TUBS implementation.

TABLE IV
OPTIMALITY RATE OF OUR IMPLEMENTATIONS AND BS3.

Class	Source	n	Optimality Rate (%)	
			BS3	I3
Simple	From [5]	200	96.67	**100.00**
		500	96.67	**100.00**
		1000	90.00	**100.00**
		2000	60.00	**100.00**
		5000	26.67	**100.00**
Orthogonal	From [5]	200	96.67	96.67
		500	93.33	93.33
		1000	86.67	**100.00**
		2000	70.00	**100.00**
		5000	40.00	93.33
Simple-simple	From [7]	200	86.67	**100.00**
		500	60.00	**100.00**
		1000	13.33	**100.00**
		2000	0.00	46.67
		5000	0.00	0.00
Ortho-ortho	From [7]	200	86.67	**100.00**
		500	53.33	**100.00**
		1000	16.67	96.67
		2000	0.00	33.33
		5000	0.00	0.00
von Koch	From [5]	200	**100.00**	**100.00**
		500	93.33	**100.00**
		1000	96.67	**100.00**
		2000	86.67	**100.00**
		5000	0.00	0.00
Spike	From [20]	200	96.67	**100.00**
		500	**100.00**	**100.00**
		1000	**100.00**	**100.00**
		2000	**100.00**	**100.00**
		5000	96.67	**100.00**

In Fig. 17, it is easy to see that BS3 was faster in solving simple-simple, ortho-ortho and von Koch polygons. On the other hand, I3 was more efficient with simple polygons and meaningly better when dealing with orthogonal and spike instances. In the specific case of the spike class, I3 was about 20 times faster than Kröller et al. implementation to solve the instances with 5000 vertices.

Through all results presented, one can conclude that the method from TUBS have a natural difficulty in converging to a proven optimal solution. While some positive results where observed in run time, the optimality rate of BS3 was not able to follow it. For illustration, BS3 needed an average time of 164.65 seconds to solve simple-simple polygons with 1000 vertices (26% percent less than I3), but the optimality rate for this subgroup was only 13.33%, far below the results using I3, when **all** 30 instances were solved within the imposed time limit. In the case of our method, it seems that our technique, since its first release, tends to find the optimal solution in almost all cases and the low optimality observed in larger instances is directly related to the maximum run time imposed in the testing environment.

D. More Results

In the thesis [12], a complete analysis is presented as well as results of the entire set of experiments, which include other interesting data, as, for example, the effect of employing different techniques for choosing the witness and the guard

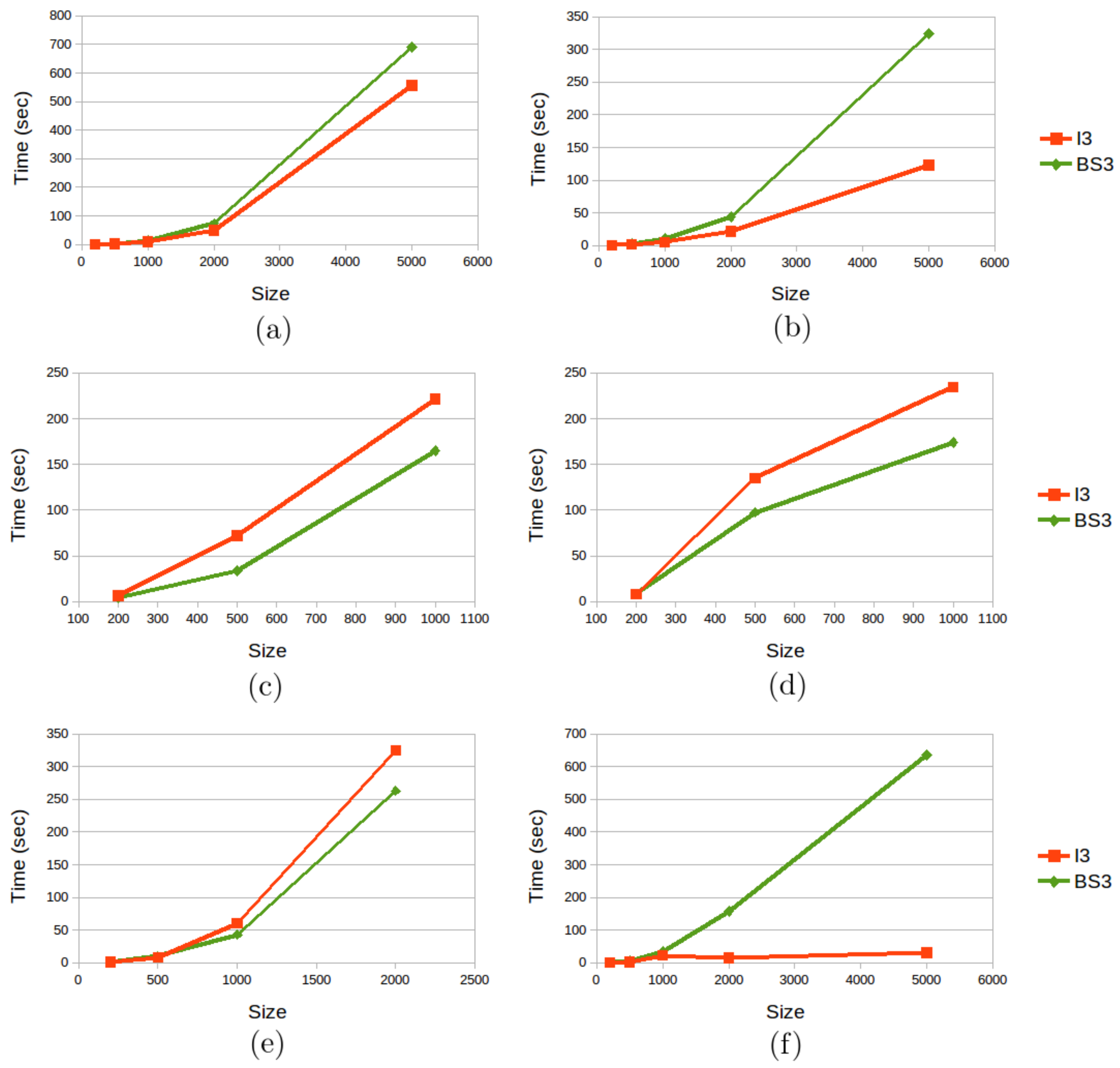


Fig. 17. Performance comparison between I3 and TUBS' technique (BS3) when solving the following classes: (a) Simple; (b) Orthogonal; (c) Simple-simple; (d) Ortho-ortho; (e) von Koch; (f) Spike. Here, only fully solved instances are considered.

candidate sets. In addition, we also reported the results of using different ILP solvers in our implementation.

V. Conclusion

In this work, we designed an algorithm to optimally solve the AGP. The method iteratively discretizes the original problem to find lower and upper bounds while seeking an optimal solution for this $\mathbb{NP}$-hard problem. To allow its correct evaluation, our algorithm was coded and had its implementation modified and optimized over time. In total, we tested our technique on more than 2800 instances from different sources and classes of polygons. Our methodology proved capable of optimally solving polygons with up to 5000 vertices in less than 20 minutes each, something not possible a few years ago.

Moreover, we also compared our results with those produced by other state-of-the-art techniques. These comparisons revealed a significant advantage when using our technique, which proved to be far more effective, faster and more robust than all the others. These results encouraged us to release a free source implementation of our algorithm on the web page of the project [8]. By doing so, we expect to contribute to future research on the topic, since it is now possible for new techniques to be directly tested and compared to our software package.

Lastly, this research generated four papers on the subject, two of which have already been published [22], [6] and two recently submitted [7], [17]. These studies provided a strong interaction with other researchers on the topic, as was the case of the survey for the AGP [17], produced in partnership with a group from TUBS.

References

[1] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory Series B*, vol. 18, pp. 39–41, 1975.

[2] D. T. Lee and A. Lin, "Computational complexity of art gallery problems," *Information Theory, IEEE Transactions on*, vol. 32, no. 2, pp. 276–282, March 1986.

[3] A. Aggarwal, S. K. Ghosh, and R. K. Shyamasundar, "Computational complexity of restricted polygon decompositions," in *Computational Morphology*, G. T. Toussaint, Ed. North-Holland, 1988, pp. 1–11.

[4] A. Kröller, T. Baumgartner, S. P. Fekete, M. Moeini, and C. Schmidt, "Practical solutions and bounds for art gallery problems," August 2012. [Online]. Available: http://ismp2012.mathopt.org/show-abs?abs=1046

[5] M. C. Couto, P. J. de Rezende, and C. C. de Souza, "An exact algorithm for minimizing vertex guards on art galleries," *International Transactions in Operational Research*, vol. 18, no. 4, pp. 425–448, 2011. [Online]. Available: http://dx.doi.org/10.1111/j.1475-3995.2011.00804.x

[6] D. C. Tozoni, P. J. de Rezende, and C. C. de Souza, "The quest for optimal solutions for the art gallery problem: A practical iterative algorithm," in *Proceedings of the 12th International Symposium on Experimental Algorithms, SEA 2013*, ser. Lecture Notes in Computer Science, V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela, Eds., vol. 7933. Rome, Italy: Springer, 2013, pp. 320–336.

[7] ——, "A practical iterative algorithm for the art gallery problem using integer linear programming," *Optimization Online*, Oct. 2013, www.optimization-online.org/DB_HTML/2013/11/4106.html.

[8] P. J. de Rezende, C. C. de Souza, M. C. Couto, and D. C. Tozoni, "The Art Gallery Problem Project (AGPPROJ)," 2013, *www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery*.

[9] P. Bose, A. Lubiw, and J. I. Munro, "Efficient visibility queries in simple polygons," *Computational Geometry*, vol. 23, no. 3, pp. 313–335, 2002.

[10] S. K. Ghosh, "Approximation algorithms for art gallery problems," in *Proc. Canadian Inform. Process. Soc. Congress*. Mississauga, Ontario, Canada: Canadian Information Processing Society, 1987, pp. 429–434.

[11] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, ser. The IBM Research Symposia Series, R. Miller, J. Thatcher, and J. Bohlinger, Eds. Springer US, 1972, pp. 85–103. [Online]. Available: http://dx.doi.org/10.1007/978-1-4684-2001-2_9

[12] D. C. Tozoni, "Solving the Art Gallery Problem: A Practical and Robust Method for Optimal Point Guard Positioning," Master's thesis, University of Campinas, Campinas, Sao Paulo, Brazil, 2014.

[13] XPRESS, *Xpress Optimization Suite*, FICO Solutions, 2013, http://www.fico.com/en/products/fico-xpress-optimization-suite/ (access November 2013).

[14] CPLEX, *IBM CPLEX Optimizer*, IBM, 2015, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/ (access June 2015).

[15] GLPK, *GNU Linear Programming Kit*, GNU, 2013, http://www.gnu.org/software/glpk/ (access December 2013).

[16] K.-Y. Chwa, B.-C. Jo, C. Knauer, E. Moet, R. van Oostrum, and C.-S. Shin, "Guarding art galleries by guarding witnesses," *Intern. Journal of Computational Geometry And Applications*, vol. 16, no. 02n03, pp. 205–226, 2006. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218195906002002

[17] P. J. de Rezende, C. C. de Souza, S. Friedrichs, M. Hemmer, A. Kröller, and D. C. Tozoni, "Engineering art galleries," 2014, submitted.

[18] CGAL, "Computational Geometry Algorithms Library," 2012, www.cgal.org (access January 2012).

[19] A. Bottino and A. Laurentini, "A nearly optimal algorithm for covering the interior of an art gallery," *Pattern Recognition*, vol. 44, no. 5, pp. 1048–1056, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/B6V14-51JF80D-1/2/4dd0f7df085fe27ab888c7c017f60512

[20] A. Kröller, T. Baumgartner, S. P. Fekete, and C. Schmidt, "Exact solutions and bounds for general art gallery problems," *J. Exp. Algorithmics*, vol. 17, no. 1, pp. 2.3:2.1–2.3:2.23, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2133803.2184449

[21] Y. Amit, J. S. B. Mitchell, and E. Packer, "Locating guards for visibility coverage of polygons," in *ALENEX*. New Orleans, Lousiana: SIAM, January 2007, pp. 1–15.

[22] D. Borrmann, P. J. de Rezende, C. C. de Souza, S. P. Fekete, S. Friedrichs, A. Kröller, A. Nüchter, C. Schmidt, and D. C. Tozoni, "Point guards and point clouds: solving general art gallery problems," in *Proceedings of the twenty-ninth annual symposium on Computational geometry*, ser. SoCG '13. New York, NY, USA: ACM, 2013, pp. 347–348. [Online]. Available: http://doi.acm.org/10.1145/2462356.2462361

Multivariate Investigation of NP-Hard Problems: Boundaries Between Parameterized Tractability and Intractability

Uéverton dos Santos Souza
Instituto de Computação - Universidade Federal Fluminense
Av. Gal. Milton Tavares de Souza, s/n^o
São Domingos - Niterói - RJ, CEP: 24210-346, Brazil
Email: usouza@ic.uff.br

Advisor: Fábio Protti
Instituto de Computação - Universidade Federal Fluminense
Av. Gal. Milton Tavares de Souza, s/n^o, São Domingos
Niterói - RJ, CEP: 24210-346, Brazil
Email: fabio@ic.uff.br

Co-advisor: Maise Dantas da Silva
Dep. Ciência e Tecnologia - Universidade Federal Fluminense
Rua Recife, s/n, Jardim Bela Vista,
Rio das Ostras - RJ, CEP: 28895-532, Brazil
Email: maise@vm.uff.br

Co-advisor: Dieter Rautenbach
Inst. Opt. und Operations Research - Universität Ulm
Helmholtzstraße 18 / Raum 1.68
89081 Ulm, Germany
Email: dieter.rautenbach@uni-ulm.de

***Abstract*—The main goal when using computing to solve a problem is to develop a mechanism to solve it efficiently. In general, this efficiency is associated with solvability in polynomial time. The theory of NP-completeness was developed to show which problems probably do not have polynomial time algorithms. However, many NP-hard and NP-complete problems must still be solved in practice; therefore it is natural to ask if each of these problems admits an algorithm whose non-polynomial time complexity is purely a function of some subset of its aspects. Questions about the existence of such algorithms are addressed within the theory of parameterized computational complexity developed by Downey and Fellows.**

In this thesis we present a multivariate investigation of the complexity of some NP-hard problems, i.e., we first develop a systematic complexity analysis of these problems, defining its subproblems and mapping which one belongs to each side of an "imaginary boundary" between polynomial time solvability and intractability. After that, we analyze which sets of aspects of these problems are *sources* of their intractability, that is, subsets of aspects for which there exists an algorithm to solve the associated problem, whose non-polynomial time complexity is purely a function of those sets. Thus, we use classical and parameterized computational complexity in an alternate and complementary approach, to show which subproblems of the given problems are NP-hard and latter to diagnose for which sets of parameters the problems are fixed-parameter tractable, or in FPT.

This thesis exhibits a classical and parameterized complexity analysis of different groups of NP-hard problems. The addressed problems are divided into four groups of distinct nature, in the context of data structures, combinatorial games, and graph theory: (I) *and/or* graph solution and its variants; (II) flooding-filling games; (III) problems on P_3-convexity; (IV) problems on induced matchings.

Keywords*—*Parameterized Complexity, And/Or Graph Solution, Flood-filling Games on Graphs, P_3-Convexity, Induced Matching.

I. Introduction

The question "$P = NP$?" is the most important open question in computer science, and the theory of NP-completeness was developed to show which problems probably do not have polynomial-time algorithms. Though it is nice to know that some problems do not have polynomial time algorithms unless $P = NP$, many NP-hard and NP-complete problems must still be solved in practice (especially those with real-world applications); therefore it is natural to ask if each of these problems admits an algorithm whose non-polynomial time complexity is purely a function of some subset of its aspects (in pratice many aspects of the problem often has bounded size or value). Questions about the existence of such algorithms are addressed within the theory of parameterized computational complexity developed by Downey and Fellows [1], [3], [4].

In the first part of the thesis, we present a detailed review of the theory of parameterized complexity, where the concepts and techniques involving fixed-parameter tractability and intractability are discussed.

The parameterized complexity theory became very popular in recent years and has become an important research topic at many universities around the world. In this sense, the Latin-American community can be somewhat outdated, with few Latin-American researchers working on this area. A major contribution of this thesis is to collaborate in popularizing the parameterized complexity nationwide. In this regard, as a consequence of our work, we can mention a narrowing of the relationship between researchers and the professors Frances Rosamond and Michael R. Fellows, one of the authors of the parameterized complexity. This relationship resulted in some studies developed with their co-authorship, and in particular in their visit to Latin-American in November 2014 and their subsequent participation as invited speakers in the 6th Latin American Workshop on Cliques in Graphs.

In addition, one of the goals of this thesis it is to make an analysis on the sources of polynomial time intractability of some interesting problems. We develop a systematic complexity analysis of these problems, defining its subproblems and mapping which one belongs to each side of an "imaginary boundary" between polynomial-time solvability and intractability. After that, we analyze which sets of aspects of these problems are *sources* of their intractability, that is, subsets of aspects for which there exists an algorithm to solve the associated problem, whose non-polynomial time complexity is purely a function of these sets. Thus, we use classical and parameterized computational complexity in an alternate and complementary approach, to show which subproblems of the given problems are NP-hard and latter to diagnose for which sets of parameters the problems are fixed-parameter tractable, or in FPT.

This thesis exhibits a classical and parameterized complexity analysis of different groups of NP-hard problems. The problems studied are of distinct nature, and the concepts discussed have applications in many areas such as software engineering, distributed systems, artificial intelligence, bioinformatics, operational research, social networks, automation, game theory, among others. Morever, the proofs presented are of several types such as NP-hardness proofs, polynomial algorithms, structural characterizations, W[1]-hardness and W[2]-hardness proofs, FPT algorithms, polynomial kernel results, and infeasibility of polynomial kernels.

The first group of studied problems involve two important data structures used for modeling many real-word applications, *and/or graphs* and *x-y graphs*. An and/or graph is an acyclic digraph containing a source, such that every vertex $v \in V(G)$ has a label $f(v) \in \{\texttt{and}, \texttt{or}\}$. X-y graphs are a generalization of and/or graphs: every vertex v_i of an x-y graph has a label x_i-y_i meaning that v_i depends on x_i of its y_i out-neighbors. We investigate the complexity of finding a solution subgraph H of such digraphs, which must contain the source and obey the following rule: if a vertex is included in H then x_i of its out-edges must also be included in H, where an `and`-vertex has $x_i = y_i$, and an `or`-vertex has $x_i = 1$.

The second group of problems consists of variants of a one-player combinatorial game known as the Flood-Filling Game, which is played on a colored board and whose objective is to make the board monochromatic ("flood the board") with the minimum number of flood moves. A flood move consists of assigning a new color c_i to the a pivot tile p and also to all the tiles connected to p by a monochromatic path immediately before the move. The flood-filling game where all moves use the same pivot p is denoted by Flood-It. When the player can freely choose which tile will be the pivot of each move the game is denoted by Free-Flood-It.

The third group comprises some problems on P_3-convexity. More specifically we are interested in identifying either the minimum P_3-geodetic set or the minimum P_3-hull set S of a graph, from which the whole vertex set of G is obtained either after one or eventual iterations, respectively. Each iteration adds to a set S' of vertices all the vertices of $V(G) \setminus S'$ with two neighbors in S'.

The last group of problems studied in this thesis focus on a classical topic in graph theory. These problems are related to maximum matchings, maximum induced matchings, and the distance between them in a graph. The matching number $\nu(G)$ of G is the maximum cardinality of a matching in G, and a matching with $\nu(G)$ edges is a maximum matching of G. An induced matching is a set M' of edges of G at pairwise distance at least 2. The induced matching number $\nu_2(G)$ of G is the maximum cardinality of an induced matching in G, and an induced matching with $\nu_2(G)$ edges is a maximum induced matching. The distance between a maximum matching of a graph G and its maximum induced matching is the difference between the cardinality of these sets $(\nu(G) - \nu_2(G))$.

For each group of problems above, there is a chapter in this thesis devoted to it. A brief abstract of our work and the obtained results it is presented at the beginning of each chapter.

Below we present the list of papers developed and published throughout my PhD, related to this thesis.

1) Tractability and hardness of flood-filling games on trees. *Theoretical Computer Science*, v. 576, p. 102-116, 2015.
2) Maximum Induced Matchings close to Maximum Matchings. *Theoretical Computer Science*, 2015. (to appear)
3) Complexity Properties of Complementary Prism. *Journal of Combinatorial Optimization*, 2015. (to appear)
4) An algorithmic analysis of Flood-it and Free-Flood-it on graph powers. *Discrete Mathematics and Theoretical Computer Science*, v. 16, p. 279-290, 2014.
5) Revisiting the complexity of and/or graph solution. *Journal of Computer and System Sciences*, v. 79, p. 1156-1163, 2013.
6) Complexidade Parametrizada para Problemas em Grafos E/OU. *Pesquisa Operacional para o Desenvolvimento*, v. 4, p. 160-174, 2012.
7) The Flood-It game parameterized by the vertex cover number. *Electronic Notes in Discrete Mathematics*, LAGOS 2015.
8) On Graphs with Induced Matching Number Almost Equal to Matching Number. *Electronic Notes in Discrete Mathematics*, LAGOS 2015.
9) On P3-convexity of Graphs with Bounded Degree. *Lecture Notes in Computer Science*, v. 8546, p. 263-274, 2014. *10th Conference on Algorithmic Aspects of Information and Management (AAIM).*
10) Parameterized Complexity of Flood-Filling Games on Trees. *Lecture Notes in Computer Science*, v. 7936, p. 531-542, 2013. *19th International Computing and Combinatorics Conference (COCOON).*
11) Complexity of Geodetic Number Problem in Graphs with Maximum Degree 4. *13th Cologne-Twente Workshop on Graphs and Comb. Optimization (CTW)*, 2015.
12) The P3-Convexity in the Complementary Prism of a Graph *13th Cologne-Twente Workshop on Graphs and Comb. Optimization (CTW)*, 2015.
13) Parameterized And/Or Graph Solution. *12th Cologne-Twente Workshop on Graphs and Comb. Optimization (CTW)*, 2013.
14) Optimal Variability Selection in Product Line En-

gineering.[1] 24th Conference on Software Eng. and Knowledge Engineering (SEKE), 2012.

15) Inundação em Grafos. *16th Congreso Latino Iberoamericano de Investigación Operativa (CLAIO)*, 2012.

Other projects were developed and submitted along my PhD, such as:

16) P_3-Convexity Problems on Bounded-Degree and Planar Graphs. *Theoretical Computer Science*.
17) Tractability and Kernelization Lower Bound for And/Or Graph Solution. *Discrete Applied Mathematics*.
18) Complexity of Geodetic Number Problem in Graphs with Bounded Degree. *Journal of Computer and System Sciences*.
19) A Multivariate Investigation of Flood-It Game. *Discrete Applied Mathematics*.
20) Parameterized Problems on Complementary Prisms. *Discrete Mathematics*.

The main results of this work are briefly presented as follows.

II. Background

A *computational problem* is a question to be answered, typically containing several variables whose values are unspecified. An *instance* of a problem is created by specifying particular values for its variables. A problem is described by specifying both its instances and the nature of solutions for those instances.

A *decision problem* Π consists of a set D_Π of instances and a set $Y_\Pi \subseteq D_\Pi$ of yes-instances. A decision problem is described informally by specifying: (i) a generic instance in terms of its variables; (ii) a yes-no question stated in terms of the generic instance.

An *optimization problem* Π consists of a set D_Π of instances and a set $S_\Pi \subseteq D_\Pi$ of solutions such that for each $I \in D_\Pi$, there is an associated set $S_\Pi[I] \subseteq S_\Pi$ of solutions for I. An optimization problem is described informally by specifying: (i) a generic instance in terms of its variables; (ii) an objective function g to be calculated, and the properties that must be satisfied by any solution associated with an instance created from the generic instance. An optimal solution $S_\Pi[I]$ is a solution which maximizes/minimizes the value $g(S_\Pi[I])$.

An *algorithm A* for a problem Π is a finite sequence of instructions for some computer which solves Π. A *polynomial time algorithm* is defined to be one whose time complexity function is $O(p(n))$ for some polynomial function p, where n is used to denote the input length [2].

Definition 1: A problem Π belongs to class *P* if and only if Π can be solved in polynomial time by a deterministic algorithm.

Definition 2: A problem Π belongs to class *NP* if and only if for a given certificate (a string that certifies the answer to a computation), there is a deterministic algorithm which verifies its validity in polynomial time.

[1]In this paper the authors apply `and/or` graphs in software engineering problems.

Definition 3: Given two problems Π and Π', $\Pi \propto \Pi'$ (Π is reducible to Π' in polynomial time) if there exists an algorithm that, given an instance I of Π, constructs an instance I' of Π' in polynomial time in $|I|$ such that from a subroutine to I', a correct answer for I is output.

Definition 4: A problem Π' is *NP-hard* if for all problems $\Pi \in NP$, $\Pi \propto \Pi'$; if Π' is also in NP, then Π' is NP-complete.

It is easy to see that $\Pi \in P$ implies $\Pi \in NP$. If any single NP-hard problem can be solved in polynomial time, then all problems in NP can also be solved in polynomial time. If any problem in NP cannot be solved in polynomial time, then so neither can all NP-complete problems. An NP-complete problem Π, therefore, has the following property: $\Pi \in P$ if and only if $P = NP$. The question "$P = NP$?" is the most important open question in computer science.

An algorithm is *efficient* if its complexity function satisfies some criterion, e.g., the complexity function is a polynomial in the instance size. A problem is *tractable* if it has an efficient algorithm; otherwise, the problem is said to be *intractable*. As there are many possible criteria which can be used to define efficiency, there are many possible types of tractability and intractability [5].

A. Parameterized Tractability in Pratice

The theory of NP-completeness was developed to show which problems do not probably have polynomial time algorithms. Since the beginning of this theory in 1971, thousands of other problems have been shown to be NP-hard and NP-complete. Though it is nice to know that such problems do not have polynomial time algorithms unless $P = NP$, a inconvenient fact remains: these problems (especially those with real-world applications) must still be solved. Thus, the following question emerges:

"How does one solve NP-complete problems efficiently in practice?"

Firstly, we have two possibilities:

× - Try to construct a polynomial time algorithm (implies $P = NP$).

√ - Invoke some type of polynomial-time heuristic algorithm.

However, in pratice some set of aspects of the problem has bounded size or value. There are another approaches:

× - Invoke some type of "brute force" optimal-solution technique, that in effect runs in polynomial time because its time complexity function is $O(n^{f(k)})$, where n is used to denote the input length and k is some aspect with bounded size or value. However, when the instances to be solved are large, this approach may not be feasible.

√ - Invoke a non-polynomial time algorithm such that its non-polynomial time complexity is *purely* a function of some subset of aspects of the problem that are of bounded size or value in practice.

This last approach immediately suggests the following questions:

1) Given a problem and a subset of aspects of that problem, is there an algorithm for that problem whose non-polynomial time complexity is purely a function of those aspects?
2) Relatively to which aspects of that problem do such algorithms exist?

If a problem Π for a set K of its aspects admits such algorithms described in (1), i.e. solvable in $f(K).n^{O(1)}$ time, then $\Pi \in FPT$ with respect to K (the class of *fixed-parameter tractable* problems). Alternatively, one can show that such algorithm probably does not exist by establishing a intractability of this version of the problem.

B. Multivariate Investigation

According to Garey and Johnson [2], whenever we are confronted with a new problem, a natural first question to ask is: Can it be solved via a polynomial time algorithm? We can concentrate our efforts on trying to find a polynomial time algorithm as efficient as possible. However, if no polynomial time algorithm is apparent, an appropriate second question to ask is: “Is the problem NP-complete?”. Suppose now we have just succeeded in demonstrating that our initial problem is NP-complete. Even though this answers the two questions which began our analysis, there are still many appropriate follow-up questions that could be asked. The problem we have been analyzing is often distilled from a less elegant applied problem, and some of the details that were dropped in the distillation process might alter the problem enough to make it polynomially solvable. If not, there still might be significant special cases that can be solved in polynomial time. Such possibilities can be investigated by analyzing subproblems of our original problem.

It should be apparent that, even though a problem Π is NP-complete, each of the subproblems of Π might independently be either NP-complete or polynomially solvable. Assuming that $P \neq NP$, we can view the subproblems of any NP-complete problem Π as lying on different sides of an imaginary “boundary” between polynomial time solvability and intractability [2].

Figure 1 [2] gives a schematic representation for one possible “current state of knowledge” about a collection of subproblems of a problem Π.

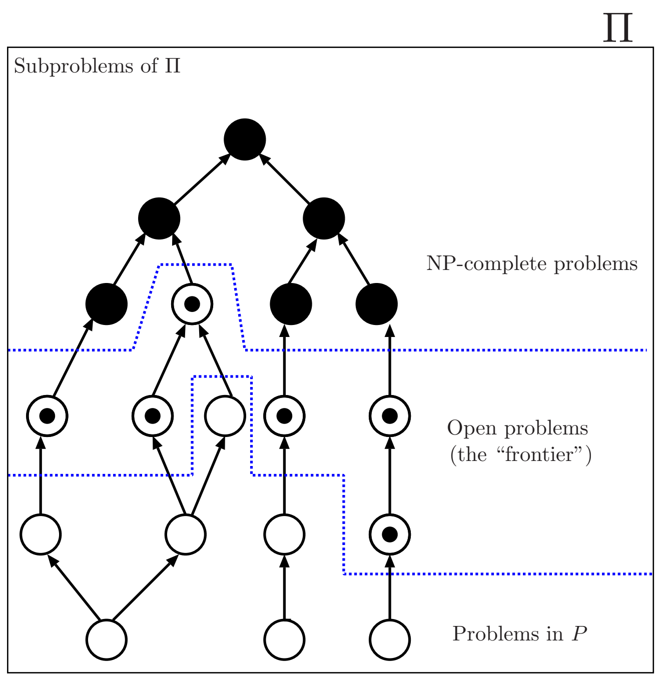


Fig. 1. One possible state of knowledge about subproblems of an NP-complete problem Π. An arrow from Π^1 to Π^2 signifies that Π^1 is a subproblem of Π^2.

In this thesis, our first goal when analyzing a problem is to determine which subproblems lie on each side.

1) Intractability Mapping: Any problem Π contains a domain D that is the set of all instances of Π. A problem Π' is a subproblem of Π if it asks the same question as Π, but only over a subset of the domain of Π.

Definition 5: Let Π be a problem with domain D and let $C = \{a_1, a_2, \ldots, a_\ell\}$ be a subset of aspects of Π. We denote by:

- *[a_1=c'_1,a_2=c'_2,...,a_ℓ=c'_ℓ]*-Π the subproblem of Π with domain D' such that each instance in D' has aspects $a_1, a_2, \ldots, a_\ell$ bounded by the constants $c'_1, c'_2, \ldots, c'_\ell$ respectively.
- *[a_1,a_2,...,a_ℓ]*-Π, or *[C]*-Π, the family of variants of Π such that every aspect in C is bounded by some constant.

Given an NP-hard problem Π and a subset C of its aspects, a systematic complexity analysis starts from the following steps.

1) Verify if [C]-Π is in P, or NP-hard.
2) If [C]-Π is in P, determine each minimal subset C' of C such that [C']-Π is in P.
3) If [C]-Π is NP-hard, determine for which values of the aspects in C the problem is solvable in polynomial time or remains NP-hard.

In a systematic complexity analysis of a problem Π, it is very common to identify subproblems of Π which can be solved in polynomial time. In general, it can be shown by some exhaustive algorithm in time $O(n^{f(k)})$, where n is used to denote the input length and k is some aspect with bounded size or value. Note that, when the instances to be solved are large, this approach may not be feasible in practice.

A parameter is a function which extracts a particular aspect or set of aspects of a problem from instances of that problem; it can also be considered as that set of aspects. As such, a parameter is both a mechanism for isolating an aspect of a problem and the “container” in which these aspects are packaged for subsequent manipulation [5].

Definition 6: A parameterized problem Π is described informally by specifying:

- A generic instance in terms of its variables.
- The aspects of an instance that comprise the parameter.

- A question stated in terms of the generic instance.

Definition 7: Let Π be a NP-hard problem and let $S = \{a_1, a_2, \ldots, a_\ell\}$ be a subset of aspects of Π. We denote by:

- $\Pi(a_1, a_2, \ldots, a_\ell)$, or $\Pi(S)$, the parameterized version of Π where the aspects in S are fixed as parameters.

Definition 8: [1] A parameterized problem $\Pi(S)$ belongs to the class XP if there exists an algorithm to solve $\Pi(S)$ in time $f(S).n^{g(S)}$, where n is used to denote the input length and f and g are arbitrary functions.

Observation 1: [S]-Π and $\Pi(S)$ are different problems. The instances of *[S]-*Π has the aspects in S with size bounded by constants, while in $\Pi(S)$ the parameters are just a mechanism for isolating aspects for subsequent manipulation (in this case, the aspects not necessarily have bounded size).

Lemma 1: Given an NP-hard problem Π and a subset S of its aspects, if *[S]-*Π remains NP-hard, then the parameterized problem $\Pi(S)$ is not in XP, unless $P = NP$.

Proof. If Π is in XP then by definition this problem is solved by an algorithm that runs in time $f(S)n^{g(S)}$ for some functions f and g. When the value of every aspect in S is fixed, the values of $f(S)$ and $g(S)$ become constants and this running time becomes polynomial in n. As this algorithm also solves $[S]$-Π and $[S]$-Π is NP-hard then $P = NP$.

Corollary 1: If $P \neq NP$, then $\Pi(S)$ is in XP if and only if *[S]-*Π is solvable in polynomial time.

Given a problem Π and some subset $S = \{s_1, \ldots, s_n\}$ of the aspects of Π, there are 3^n different basic families of variants of the problem, based on which of the aspects is declared as either:

1) an unrestricted part of the input,
2) part of the aggregate parameterization, or
3) a fixed constant (yielding part of the indexing of the family of parameterized problems).

Let Π be problem and let $S = \{s_1, \ldots, s_n\}$ be a subset of the aspects of Π. $[S_1]$-$\Pi(S_2)$ is the family of parameterized problems where the aspects in $S_1 \subseteq S$ are fixed constants and the aspects in $S_2 \subseteq S$ are aggregate parameters.

> A parameterized problem $\Pi(S)$ belongs to the class FPT, or *fixed-parameter tractable*, if there exists an algorithm to solve $\Pi(S)$ in time $f(S).n^{O(1)}$, where n is used to denote the input length and f is an arbitrary function.

Individual parameterized results are very good at establishing whether or not a given problem has an FPT-algorithm for a particular set of aspects of that problem. However, if one is interested in fully characterizing the set of FPT-algorithms for parameterized versions of a NP-hard problem, individual results are not sufficient because a fixed-parameter tractability (intractability) result says nothing about which subsets (supersets) of its aspects also render fixed-parameter tractability (intractability) [5]. In this case, it is necessary to make a systematic parameterized complexity analysis of the problem, determining the parameterized complexity relative to all non-empty subset of aspects of the problem.

A list of parameterized results produced by a systematic parameterized complexity analysis relative to some set of aspects S for a problem Π can be visualized as a *polynomial time intractability map* that shows which sets of aspects of the problem can be said to be responsible for (and hence are sources of) that problem's polynomial time intractability [5].

Definition 9: [5] Given a NP-hard problem Π and some subset S of aspects of Π, S is a source of polynomial-time intractability for Π, if $\Pi(S)$ is in FPT.

> " In parameterized complexity, the focus is not on *whether* a problem is hard, the theory starts from the assumption that most interesting problems are intractable when considered classically. The focus is on the question: *What makes the problem computationally difficult?* ". [1]

In this thesis, one of the goals it is to make an analysis on the sources of polynomial time intractability of some problems, that are "minimal" in the sense that their associated FPT-algorithms are not trivial extensions of other FPT-algorithms.

C. Parameterized Complexity

Classical complexity views a problem as an instance and a question, where the running time is specified by the input's size. However, when a problem comes from "real life" we always know more about the problem. The problem is planar, the problem has small width, the problem only concerns small values of the parameters. Thus, why not have a complexity theory which exploits these structural parameters? Why not have a complexity theory more fine-tuned to actual applications? [1]

The Parameterized Complexity Theory was proposed by Downey and Fellows [1] as a promising alternative to deal with NP-hard problems described by the following general form: given an object x and a nonnegative integer k, does x have some property that depends only on k (and not on the size of x)? In parameterized complexity theory, k is set as the *parameter*, considered to be *small* in comparison with the size $|x|$ of object x. It may be of high interest for some problems to ask whether they admit deterministic algorithms whose running times are exponential with respect to k but polynomial with respect to $|x|$.

As is common in complexity theory, we describe problems as languages over finite alphabets Σ. To distinguish them from parameterized problems, we refer to sets $\Pi \subseteq \Sigma^*$ of strings over Σ (nonempty) as classical problems.

Definition 10: Let Σ be a finite alphabet.

1) A parametrization of Σ^* is a mapping $k : \Sigma^* \to \mathbb{N}$ that is polynomial time computable.
2) A parameterized problem (over Σ) is a pair $(\Pi, k) = \Pi(k)$, consisting of a set $\Pi \subseteq \Sigma^*$ and a parametrization k of Σ^*.

Example. Let SAT denote the set of all satisfiable propositional formulas, where propositional formulas are encoded as strings over some finite alphabet Σ. Let $k : \Sigma^* \to \mathbb{N}$ be the parameterization defined by:

$$k = \begin{cases} number\ of\ variables\ of\ x, & (at\ least\ one\ variable), \\ 1, & otherwise. \end{cases} \quad (1)$$

If $\Pi(k)$ is a parameterized problem over the alphabet Σ, then we call strings $x \in \Sigma^*$ *instances* of $\Pi(k)$ and k, the the corresponding parameter. Usually, we represent a parameterized problem $\Pi(k)$ in the form:

Instance: $x \in \Sigma$.
Parameter: k
Problem: Decide whether $x \in \Pi(k)$.

In the same way that the notion of *polynomial time* is central to the classical formulation of computational complexity, a central notion to parameterized complexity is *fixed-parameter tractability*.

Definition 11: A parameterized problem $\Pi(k)$ is fixed-parameter tractable, or *FPT*, if the question "$x \in \Pi(k)$?" can be decided in running time $f(k).|x|^{O(1)}$, where f is an arbitrary function on nonnegative integers. The corresponding complexity class is called *FPT*.

III. Complexity of And/Or Graph Solution

The first group of studied problems involve two important data structures used for modeling many real-word applications, *and/or graphs* and *x-y graphs*. An and/or graph is an acyclic, edge-weighted directed graph containing a single source vertex such that every vertex v has a label $f(v) \in \{\texttt{and}, \texttt{or}\}$. A solution subgraph H of an and/or-graph must contain the source and obey the following rule: if an and-vertex (resp. or-vertex) is included in H then all (resp. one) of its out-edges must also be included in H. X-y graphs are defined as a natural generalization of and/or graphs. In this section we first present the results published in the paper [U. S. Souza, F. Protti, M. Dantas da Silva, Revisiting the complexity of and/or graph solution, J. Comput. Syst. Sci. 79:7 (2013) 1156-1163] where we have investigated the complexity of such problems under various aspects, including parameterized versions of it. However, this article kept the main open question still open: Is the problem of finding a solution subgraph of cost at most k (where k is a fixed parameter) in FPT? We finish this work finally presenting a positive answer to this question, via kernelization techniques. Also, using a framework developed by Bodlaender *et al.* (2009) and Fortnow and Santhanam (2011), based upon the notion of compositionality, we show that the above parameterized problem does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

An and/or graph is an acyclic digraph containing a source (a vertex that reaches all other vertices by directed paths), such that every vertex $v \in V(G)$ has a label $f(v) \in \{\texttt{and}, \texttt{or}\}$. In such digraphs, edges represent dependency relations between vertices: a vertex labeled and depends on all of its out-neighbors (conjunctive dependency), while a vertex labeled or depends on only one of its out-neighbors (disjunctive dependency).

We define x-y graphs as a generalization of and/or graphs: every vertex v_i of an x-y graph has a label x_i-y_i to mean that v_i depends on x_i of its y_i out-neighbors. Given an and/or graph G, an equivalent x-y graph G' is easily constructed as follows: sinks of G are vertices with $x_i = y_i = 0$; and-vertices satisfy $x_i = y_i$; and or-vertices satisfy $x_i = 1$.

And/or graphs were used for modeling problems originated in the 60's within the domain of Artificial Intelligence. Since then, they have successfully been applied to other fields, such as Operations Research, Automation, Robotics, Game Theory, and Software Engineering, to model cutting problems, interference tests, failure dependencies, robotic task plans, assembly/disassembly sequences, game trees, software versioning, and evaluation of boolean formulas. With respect to x-y graphs, they correspond to the *x-out-of-y model* of resource sharing in distributed systems.

In addition to the above applications, special directed hypergraphs named *F-graphs* are equivalent to and/or graphs. An F-graph is a directed hypergraph where hyperarcs are called *F-arcs* (for *forward arcs*), which are of the form $E_i = (S_i, T_i)$ with $|S_i| = 1$. An F-graph H can be easily transformed into an and/or graph as follows: for each vertex $v \in V(H)$ do $f(v)$=or; for each F-arc $E_i = (S_i, T_i)$, where $|T_i| \geq 2$, do: create an and-vertex v_i, add an edge (u, v_i) where $\{u\} = S_i$, and add an edge (v_i, w_j) for all $w_j \in T_i$.

The optimization problems associated with and/or graphs and x-y graphs are formally defined below.

Min-and/or
Instance: An and/or graph $G = (V, E)$ where each edge e has an integer weight $\tau(e) > 0$.
Goal: Determine the minimum weight of a subdigraph $H = (V', E')$ of G (*solution subgraph*) satisfying the following properties:
• $s \in V'$;
• if a non-sink node v is in V' and $f(v)$=and then every out-edge of v belongs to E';
• if a non-sink node v is in V' and $f(v)$=or then exactly one out-edge of v belongs to E'.

Min-x-y
Instance: An x-y graph $G = (V, E)$ where each edge e has an integer weight $\tau(e) > 0$.
Goal: Determine the minimum weight of a subdigraph $H = (V', E')$ of G satisfying the following properties:
• $s \in V'$;
• for every non-sink node v_i in V', exactly x_i of its y_i out-edges belong to E'.

A. Main obtained results on the problem

Theorem 1: Min-and/or remains NP-hard even for a very restricted family of and/or graphs where edges have weight one and or-vertices have out-degree at most two.

Theorem 2:
(a) The parameterized problem Min-and/or$^0(k)$, whose domain includes and/or graphs allowing zero-weight edges, is W[2]-hard.
(b) Min-x-y(k) is *W[1]*-hard.
(c) Min-and/or(k) is fixed-parameter tractable.

Using a framework developed by Bodlaender et al. (2009) and Fortnow and Santhanam (2011), based upon the notion of compositionality, we show that the above parameterized problem does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

Theorem 3: MIN-AND/OR(k) has no polynomial kernel unless $NP \subseteq coNP/poly$ and consequently $PH \subseteq \Sigma_3^p$.

IV. Flooding-filling games

The second group of studied problems consists of variants of a one-player combinatorial game known as the Flood-Filling Game, which is played on a colored board and whose objective is to make the board monochromatic ("flood the board") with the minimum number of flood moves. A flood move consists of assigning a new color c_i to the pivot tile p and also to all the tiles connected to p by a monochromatic path immediately before the move. The flood-filling game where all moves use the same pivot p is denoted by Flood-It. When the player can freely choose which tile will be the pivot of each move the game is denoted by Free-Flood-It.

Flood-It game, originally played on a colored board consisting of an $n \times m$ grid, where each tile of the board has an initial color from a fixed color set. In the classic game, two tiles are *neighboring* tiles if they lie in the same row (resp. column) and in consecutive columns (resp. rows). A sequence C of tiles is a *path* when every pair of consecutive tiles in C is formed by neighboring tiles. A *monochromatic path* is a path in which all the tiles have the same color. Two tiles a and b are *m-connected* when there is a monochromatic path between them. In Flood-It, a move consists of assigning a new color c_i to the top left tile p (the *pivot*) and also to all the tiles m-connected to p immediately before the move. The objective of the game is to make the board monochromatic ("flood the board") with the minimum number of moves. Figure 2 shows a sequence of moves to flood a 3×3 grid colored with five colors.

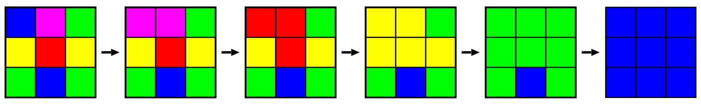


Fig. 2. An optimal sequence of moves to flood a 3×3 grid.

We consider these games when played on any graph with an initial coloring.

Many complexity issues on Flood-It and Free-Flood-It have recently been investigated. In [25], Arthur, Clifford, Jalsenius, Montanaro, and Sach show that Flood-It and Free-Flood-It are NP-hard on $n \times n$ grids colored with at least three colors. Meeks and Scott [28] prove that Free-Flood-It is solvable in polynomial time on $1 \times n$ grids and on 2-colored graphs, and also that Flood-It and Free-Flood-It remain NP-hard on $3 \times n$ grids colored with at least four colors. Up to the authors' knowledge, the complexity of Flood-It on $3 \times n$ grids colored with three colors remains as an open question. Clifford, Jalsenius, Montanaro, and Sach present a polynomial-time algorithm for Flood-It on $2 \times n$ grids. In [29], Meeks and Scott show that Free-Flood-It remains NP-hard on $2 \times n$ grids. Fleischer and Woeginger [27] proved that Flood-It is NP-hard on trees.

Flood-filling games in bioinformatics. Since the 90's, an increasing number of papers on biological applications have been dealt with as combinatorial problems. Vertex-colored graph problems have several applications in bioinformatics. The Colored Interval Sandwich Problem has applications in DNA physical mapping and in perfect phylogeny; vertex-recoloring problems appear in protein-protein interaction networks and phylogenetic analysis; the Graph Motif Problem was introduced in the context of metabolic network analysis; the Intervalizing Colored Graphs Problem models DNA physical mapping; and the Triangulating Colored Graph Problem is polynomially equivalent to the Perfect Phylogeny Problem.

Flood-Filling games on colored graphs are also related to many problems in bioinformatics. As shown in this paper, Flood-It played on trees is analogous to a restricted case of the Shortest Common Supersequence Problem. Consequently, these games inherit from the Shortest Common Supersequence Problem many applications in bioinformatics, such as: microarray production, DNA sequence assembly, and a close relationship to multiple sequence alignment. In addition, some disease spreading models, work in a similar way to flood-filling games.

We present below the formal definitions of the two flood-filling games studied in this chapter.

Flood-It (decision version)
Instance: A colored graph G with a pivot vertex p, an integer λ.
Question: Is there a sequence of at most λ flood moves which makes the graph monochromatic, using p as the pivot in all moves?

Free-Flood-It (decision version)
Instance: A colored graph G with a pivot vertex p, an integer λ.
Question: Is there a sequence of at most λ flood moves which makes the graph monochromatic?

A complete mapping of the complexity of flood-filling games on trees is made, charting the consequences of single and aggregate parameterizations. Furthermore, we show that Flood-It on trees and Restricted Shortest Common Supersequence (RSCS) are analogous problems, which proves some FPT and W[1]-hard of cases of Flood-it. In addition, we prove that Flood-It remains NP-hard when played on 3-colored trees, which closes an open question. We also present a general framework for reducibility from Flood-It to Free-Flood-It. Analyzing the behavior of these games when played on other classes of boards, we describe polynomial time algorithms, and some NP-hard cases. Finally, we show that Flood-it is fixed-parameter tractable when parameterized by the vertex cover number, and it has a polynomial kernel if the number of colors is a second parameter.

Let Π be a flood-filling game and let $S = \{s_1, \ldots, s_n\}$ be a subset of the aspects of Π. $[S_1]$-$\Pi(S_2)$ is the family of parameterized problems where the aspects in $S_1 \subseteq S$ are fixed constants and the aspects in $S_2 \subseteq S$ are aggregate parameters.

We consider the following aspects of the problem: c - number of colors; λ - number of moves; d - maximum distance of the pivot; o - maximum orbit; k - number of leaves; r - number of bad moves, $r = (\lambda - c)$.

A. Main obtained results on the problem

Theorem 4:
(a) $[d]$-Flood-It on trees remains NP-hard when $d = 2$.
(b) $[d]$-Flood-It(c) is in FPT and admits a polynomial kernelization.

Theorem 5:
(a) Flood-It on trees and RSCS are analogous problems.
(b) Flood-It(c, k, λ) on trees is p-analogous to RSCS$(|\Sigma|, \ell, \Lambda)$.

We remark that the book [2] (see Appendix) reports that the SCS problem is solvable in polynomial time when each string has size two. This assertion is false, as we can see by Theorems 4 and 5.

Corollary 2: Flood-It(k, c) on trees is W[1]-hard.

A colored rooted tree is a pc-tree (phylogenetic colored tree) if no color occurs more than once in any path from the root to a leaf. A pc-tree T is a cpc-tree if each color occurs exactly once in any path from the root to a leaf. Restricting attention to phylogenetic colored trees, we have significant effects on problem complexity.

Theorem 6:
(a) Flood-It(k) on pc-trees with pivot root is W[1]-hard.
(b) Flood-It on trees remains NP-hard even when restricted to cpc-trees.
(c) Flood-It(r) on cpc-trees with pivot root is in FPT.
(d) Flood-It(k, r) on trees is in FPT.

Let G be a graph, $v \in V(G)$, and ℓ a positive integer. The graph $\psi(G, v, \ell)$ is constructed as follows: (i) create ℓ disjoint copies $G_1, \ldots, G_\ell$ of G; (ii) contract the copies $v_1, v_2, \ldots, v_\ell$ of v into a single vertex v^*. Let $\mathscr{F}$ be a class of graphs. Then:

$\psi(\mathscr{F}) = \{G \mid G = \psi(G', v, \ell)$ for some triple $(G' \in \mathscr{F}, v \in V(G'), \ell > 0)$ $\}$.

Theorem 7: Flood-It on $\mathscr{F}$ is reducible in polynomial time to Free-Flood-It on $\psi(\mathscr{F})$.

Theorem 8:
(a) Flood-It on trees remains NP-hard even restricted to 3-colored trees.
(b) Free-Flood-It on trees remains NP-hard even restricted to 3-colored trees..

Theorem 9: In Free-Flood-It on pc-trees, there always exists an optimal free-flooding which is a flooding with pivot root.

Theorem 10:
(a) Flood-it is solvable in polynomial time on P_n^2, C_n^2, and $2 \times n$ circular grids.
(b) Free-Flood-it remains NP-hard on P_n^2, C_n^2, and $2 \times n$ circular grids.

Theorem 11: Flood-it on graphs is fixed-parameter tractable, FPT, when parameterized by the size of the minimum vertex cover (k).

Theorem 12: Flood-it on graphs admits a polynomial kernelization when parameterized by the size of the minimum vertex cover (k) and the number of colors (c).

V. P_3-CONVEXITY

The third group comprises some problems on P_3-convexity. More specifically we are interested in identifying either the minimum P_3-geodetic set or the minimum P_3-hull set S of a graph, from which the whole vertex set of G is obtained either after one or eventual iterations, respectively. Each iteration adds to a set S of vertices all the vertices of $V(G) \setminus S$ with two neighbors in S.

One of the most elementary models of the spreading of a property within a network – like sharing an idea or disseminating a virus – one can consider a graph G, a set S of vertices of G that initially possesses the property, and an iterative process whereby new vertices u are added to S whenever sufficiently many (usually two) neighbors of u are already in S. Similar models were studied in various contexts, such as statistical physics, social networks, marketing, and distributed computing.

Let $G = (V, E)$ be a graph. For $U \subseteq V$, let the interval $I[U]$ of U in G be the set $U \cup \{u \in V(G) \setminus U \mid |N_G(u) \cap U| \geq 2\}$. A set S of vertices of G is P_3*-geodetic* if $I[S]$ contains all vertices of G. The P_3*-geodetic number* $g_{P_3}(G)$ (or just $g(G)$) of a graph G is defined as the minimum cardinality of a P_3-geodetic set. The decision problem related to determining the P_3-geodetic number is known to be NP-complete for general graphs, and coincides with the well-studied 2-domination number [40], [43], [44].

A P_3*-hull* set U of G is a set of vertices such that:

- $U^0 = U$
- $U^k = I[U^{k-1}]$, for $k \geq 1$.
- $\exists\ k \geq 0 \mid U^k = V(G)$

We define $H_G(S) \subseteq V(G)$ as $I[S]^{k+1}$ such that $I[S]^{k+1} = I[S]^k$, $k \geq 0$. The cardinality of a minimum P_3-hull set of G is the P_3*-hull number* of G, denoted by $h_{p3}(G)$ (or just $h(G)$). Again, the decision problem related to determining the P_3-hull number of a graph is still a well known NP-complete problem [36].

We analyze the complexity of these problems when some parameters related to the maximum and minimum degree of a graph are known. In the following subsection we review some results on planar satisfiability problems. In Section 2 we present some results on finding a minimum P_3-hull set of graphs with bounded degree. Finally, in Section 3 we analyze complexity aspects of finding a minimum P_3-geodetic set on planar graphs with bounded degree.

A. Main obtained results on the problem

Theorem 13: Let c be a positive integer. If G is a graph with $\delta(G) \geq \frac{|V(G)|}{c}$, then

$$h_{P_3}(G) \leq 2 \left\lceil \frac{\log(2c)}{\log\left(\frac{2c^2}{2c^2-1}\right)} \right\rceil + 2c^3.$$

Corollary 3: A minimum P_3-hull set of a graph G with $\delta(G) \geq \frac{|V(G)|}{c}$ (for some constant c) can be found in polynomial time.

Theorem 14:
(a) P_3-HULL NUMBER remains NP-complete on planar graphs G with $\Delta(G) = 3$.
(b) A minimum P_3-hull set of a cubic graph can be found in polynomial time.

Theorem 15:
(a) P_3-GEODETIC NUMBER remains NP-complete on planar graphs G with $\Delta(G) = 3$.
(b) P_3-GEODETIC SET(k) is W[2]-hard.
(c) P_3-GEODETIC SET(k, Δ) is fixed-parameter tractable.

The complementary prism $G\bar{G}$ of G arises from the disjoint union of the graph G and its complement $\bar{G}$ by adding the edges of a perfect matching joining pairs of corresponding vertices of G and $\bar{G}$.

Theorem 16:
(a) To decide whether a compl. prism $G\bar{G}$ has a P_3-geodetic set of size k is NP-complete.
(b) A minimum P_3-hull set of a compl. prism $G\bar{G}$ can be found in polynomial time.

VI. Problems on induced matchings

The last group of problems studied in this thesis focus on a classical topic in graph theory. These problems are related to maximum matchings, maximum induced matchings, and the distance between them in a graph. The matching number $\nu(G)$ of G is the maximum cardinality of a matching in G, and a matching with $\nu(G)$ edges is a maximum matching of G. An induced matching is a set M' of edges of G at pairwise distance at least 2. The induced matching number $\nu_2(G)$ of G is the maximum cardinality of an induced matching in G, and an induced matching with $\nu_2(G)$ edges is a maximum induced matching. The distance between a maximum matching of a graph G and its maximum induced matching is the difference between the cardinality of these sets ($\nu(G) - \nu_2(G)$). We study graphs G with $\nu(G) - \nu_2(G) \leq k$, for some non-negative integer k.

Let G be a graph with $\nu(G) - \nu_2(G) \leq k$ for some non-negative integer k.

Let M_1 and M_2 be a maximum matching and a maximum induced matching of G, respectively, and let $H = (V(G), M_1 \Delta M_2)$.

If some component of H is a path with 2 edges, say $e_1 \in M_1$ and $e_2 \in M_2$, then $(M_1 \setminus \{e_1\}) \cup \{e_2\}$ is a maximum matching of G having more edges in common with M_2 than M_1. Iteratively applying this exchange operation to M_1, we may assume that no component of H is a path with 2 edges.

While maximum matchings can be found efficiently [57], it is algorithmically hard to find a maximum induced matching [61], [49]. It is even hard to approximate the induced matching number under substantial restrictions, and efficient exact and approximation algorithms have been proposed for several special graph classes (cf. [52], [59] for a detailed discussion). The fixed parameter tractability of induced matchings when parameterized by their cardinality was studied in [60], [59], [52]. While this problem is $W[1]$-hard in general, it was shown to be fixed parameter tractable for several restricted graph classes.

We study graphs where the matching number is not much larger than the induced matching number. Kobler and Rotics [56] showed that the graphs where these two numbers coincide, can be recognized efficiently. Cameron and Walker [51] extended this result and gave a complete structural description of these graphs. We review the results from [56], [51] and present shorter proofs. We study graphs G where $\nu(G) - \nu_2(G) \leq k$. We show that the recognition of these graphs can be done in polynomial time for fixed k and is fixed parameter tractable when parameterized by k for graphs of bounded maximum degree.

A. Main obtained results on the problem

Lemma 2: The components of H are

- isolated vertices,
- paths of length 1 whose edge belongs to $M_1 \setminus M_2$, and
- paths of length 3 whose two leaf edges belong to $M_1 \setminus M_2$ and whose middle edges belong to $M_2 \setminus M_1$.

Furthermore, H has exactly $\nu(G) - \nu_2(G)$ non-trivial components.

Theorem 17: For a fixed non-negative integer k, the graphs G with $\nu(G) - \nu_2(G) \leq k$ can be recognized in polynomial time.

Our next result states that the recognition of those graphs G with $\nu(G) - \nu_2(G) \leq k$ that are of bounded maximum degree is fixed parameter tractable when parameterized by k.

Theorem 18: Let Δ and k be positive integers. The graphs G with $\nu(G) - \nu_2(G) \leq k$ of maximum degree at most Δ can be recognized in $f(k, \Delta)n^c$ time where the constant c does not depend on Δ or k.

VII. Conclusions

In this thesis, a multivariate investigation of NP-hard problems has been carried out as a systematic application of classical and parameterized complexity techniques. This approach focused on drawing for each analyzed problem its boundaries between: (i) polynomial-time solvable and NP-hard subproblems; (ii) tractable and intractable parameterized versions. This strategy presents a more refined analysis of the complexity of problems, as well as their possibilities of solvability in practice. The idea is to map out when a problem Π becomes polynomial-time intractable and the sets of aspects that are responsible for this NP-hardness, i.e., the sets of aspects for which we can isolate its non-polynomial time complexity to solve Π as a purely function of them.

The tools used in this systematic analysis are first and foremost methods for algorithm design, or polynomial-time reductions (Karp reductions), to demonstrate polynomial time solvability or NP-hardness of a given problem, respectively. To show that a problem is fixed-parameter tractable, or in FPT, with respect to a set of parameters, we apply the bounded search tree and problem kernel (kernelization) methods. In another direction, we apply FPT-reductions (parametric reductions) to identify the level of parameterized intractability (W[t]-hardness, $t \geq 1$) of parameterized problems. Since every problem in FPT has a kernelization algorithm, we also analyze whether an FPT problem has a kernel of polynomial size with respect to its parameters. We establish the infeasibility of polynomial kernels for parameterized problems by a framework developed by Bodlaender et al. [6] and Fortnow and Santhanam [7], based upon the notion of or-compositionality, which shows that a problem does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

This thesis make a multivariate investigation of different groups of NP-hard problems: (i) and/or graph solution and its variants; (ii) flooding-filling games; (iii) problems on P_3-convexity; (iv) problems on induced matchings.

1) **And/or graph solution and its variants.** We have proved that the problem MIN-AND/OR remains NP-hard even for and/or graphs where edges have weight one, `or`-vertices have out-degree at most two, and vertices with in-degree greater than one are within distance at most one of a sink; and that deciding whether there is a solution subtree with weight exactly k of a given x-y tree is also NP-hard. In a parameterized point of view, we have shown that MIN-AND/OR$^0(k)$ is W[2]-hard, and MIN-X-Y(k) is W[1]-hard. We also deal with the main question: "Is MIN-AND/OR$(k) \in$ *FPT*?". We answer positively to this question via a reduction to a problem kernel. Finally, we analyze whether MIN-AND/OR(k) admits a polynomial kernelization algorithm, and using the framework based upon the notion of or-compositionality, we show that MIN-AND/OR(k) does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

2) **Flood-filling games**. We analyze the complexity consequences of parameterizing Flood-it and Free-Flood-it by one or two of the following parameters: c - number of colors; λ - number of moves; d - maximum distance of the pivot (or diameter, in the case of Free-Flood-It); o - maximum orbit; k - number of leaves; r - number of bad moves. During our analysis we have shown that Flood-It on trees is analogous to Restricted Shortest Common Supersequence, and Flood-It remains NP-hard on 3-colored trees, closing an open question. We also present a general framework for reducibility from Flood-It to Free-Flood-It. Analyzing the computational complexity of these games on other classes of graphs such as powers of paths, power of cycles, circular grids, and graphs with bounded vertex cover, we conclude that: (i) Flood-it can be solved in polynomial time when played on P_n^2, C_n^2, and $2 \times n$ circular grids; (ii) Free-Flood-it is NP-hard when played on P_n^2, C_n^2; and $2 \times n$ circular grids. Finally, we prove that Flood-it on graphs is fixed-parameter tractable considering the size of a minimum vertex cover as the parameter; in addition, we show a polynomial kernelization algorithm for Flood-it when, besides the minimum vertex cover, the number of colors is also a parameter.

3) **Problems on P_3-convexity**. We prove that: (i) a minimum P_3-hull set of a graph G can be found in polynomial time when $\delta(G) \geq \frac{n(G)}{c}$ (for some constant c); (ii) deciding if the size of a minimum P_3-hull set of a graph is at most k remains NP-complete even on planar graphs with maximum degree four; (iii) a minimum P_3-hull set of a cubic graph can be found in polynomial time; (iv) a minimum P_3-hull set can be found in polynomial time in graphs with minimum feedback vertex set of bounded size and with no vertices of degree two; (v) deciding if the size of a minimum P_3-geodetic set of a planar graph with maximum degree three is at most k is NP-complete. Some trivial parameterized results on P_3-geodetic sets are also shown.

4) **Problems on induced matchings.** We present a short proof of a structural description of the graphs G where the matching number $\nu(G)$ equals the induced matching number $\nu_2(G)$, and use it to study graphs G with $\nu(G) - \nu_2(G) \leq k$. We show that the recognition of these graphs can be done in polynomial time for fixed k, and is fixed parameter tractable when parameterized by k for graphs of bounded maximum degree. Finally, we extend some of Cameron and Walker's results to k-matchings in graphs of sufficiently large girth.

REFERENCES

[1] R. G. Downey, M. Fellows. *Parameterized complexity*, Springer, 1999.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[3] J. Flum, M. Grohe. *Parameterized complexity theory*, Springer, 2006.

[4] R. Niedermeier. *Invitation to fixed-parameter algorithms*, Oxford University Press, 2006.

[5] H. T. Wareham, M. R. Adviser-Fellows. Systematic parameterized complexity analysis in computational phonology, PhD thesis, University of Victoria, 1999.

[6] Hans L. Boadlaender, Rodney G. Downey, Michael R. Fellows, Danny Hermelin, On problems without polynomial kernels, *Journal of Computer and System Sciences*, v. 75, p. 423-434, 2009.

[7] L. Fortnow, R. Santhanam, Infeasibility of instance compression and succinct PCPs for NP, *Journal of Computer and System Sciences*, v.77, p. 91-106, 2011.

[8] H. L. Boadlaender, Kernelization: New upper and lower bound techniques, in: J. Chen, F. V. Fomin (Eds.), Proceedings of the 4th International Workshop on Parameterized and Exact Computation, IWPEC 2009, in: Lecture Notes in Computer Science, v. 5917, Springer Verlag, p. 17-37, 2009.

[9] Hans L. Boadlaender, Stéphan Thomassé, Anders Yeo, Kernel bounds for disjoint cycles and disjoint paths, *Theoretical Computer Science*, v. 412, p. 4570-4578, 2011.

[10] Hans L. Bodlaender, Bart M.P. Jansen, Stefan Kratsch, Kernel bounds for path and cycle problems, *Theoretical Computer Science*, doi:10.1016/j.tcs.2012.09.006, 2012

[11] Cao T., Sanderson, A. C., And/or net representation for robotic task sequence planning, *IEEE Trans. Systems Man Cybernet, Part C: Applications and Reviews*, v. 28, p. 204-218, 1998.

[12] Corandi R., Westfechtel B., Version models for software configuration management, *ACM Computing Surveys*, v. 30, p. 233-282, 1998.

[13] DeMello L. S. H., Sanderson A. C., A correct and complete algorithm for the generation of mechanical assembly sequences, *IEEE Trans. Robotics and Automation*, v. 7, p. 228-240, 1991.

[14] Gallo, G., Longo, G., Nguyen, S., Pallottino, S., Directed hypergraphs and applications, *Discrete Applied Mathematics*, v. 42, p. 177-201, 1993.

[15] Guo, J., Niedermeier, R., Invitation to Data Reduction and Problem Kernelization, *ACM SIGACT News*, v. 38, n. 1, p. 31-45, 2007.

[16] Neeldhara Misra, Venkatesh Raman, Saket Saurabh, Lower Bounds on Kernelization, *Discrete Optimization*, v. 8, p. 110-128, 2011.

[17] Karp R. M., Reducibility among combinatorial problems, in R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, Plenum Press, 1972.

[18] Medeiros, R. P., Souza, U. S., Protti, F., Murta, L. G. P., Optimal Variability Selection in Product Line Engineering, *Proc. of the 24th International Conference on Software Engineering and Knowledge Engineering - SEKE 2012.*

[19] Morabito R., Pureza V., A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem, *Annals of Operation Research*, v. 179, p. 297-315, 2010.

[20] Sahni S., Computationally related problems, *SIAM Journal on Computing*, v. 3, n. 4, p. 262-279, 1974.

[21] Souza U. S., Protti F., Dantas da Silva M., Complexidade parametrizada para problemas em grafos E/OU, *Pesquisa Operacional para o Desenvolvimento*, v. 4, n. 2, p. 160-174, 2012.

[22] Souza, U. S., Protti, F., Dantas da Silva, M., Revisiting the Complexity of And/Or Graph Solution, *Journal of Computer and System Sciences*, v. 79, p. 1156-1163, 2013.

[23] Souza, U. S., Protti, F., Dantas da Silva, M., Parameterized And/Or Graph Solution, *Proc. of the 12th Cologne Twente Workshop on Graphs and Combinatorial Optimization - CTW 2013*, 205-208, 2013.

[24] Chee-Keng Yap, Some Consequences of Non-Uniform Conditions on Uniform Classes, *Theoretical Computer Science*, v. 26, p. 287-300, 1983.

[25] D. Arthur, R. Clifford, M. Jalsenius, A. Montanaro, and B. Sach, The Complexity of Flood-Filling Games, in Paolo Boldi and Luisa Gargano, editors, *Proceedings of FUN*, *Lecture Notes in Computer Science* 6099 (2010) 307–318, Springer.

[26] M. R. Fellows, M. T. Hallett, and U. Stege, Analogs & Duals of the MAST problem for Sequences & Trees, *Journal of Algorithms* 49:1 (2003) 192–216.

[27] R. Fleischer, G. J. Woeginger, An Algorithmic Analysis of the Honey-Bee Game, *Theoretical Computer Science* 452, pp. 75-87, 2012.

[28] K. Meeks and A. Scott, The Complexity of Flood-Filling Games on Graphs, *Discrete Applied Mathematics* 160 (2012) 959–969.

[29] K. Meeks and A. Scott, The Complexity of Free-Flood-It on $2 \times n$ Boards, arXiv:1101.5518v1 [cs.DS], January 2011.

[30] U. S. Souza, F. Protti, and M. Dantas da Silva, Parameterized Complexity of Flood-Filling Games on Trees. In: D.-Z. Du, G. Zhang, Editors, *Proceedings of COCOON 2013 - 19th International Computing & Combinatorics Conference*, Hangzhou, China, June 2013, *Lecture Notes in Computer Science* v. 7936, pp. 531-542.

[31] Souza, Protti and Dantas da Silva, *"Inundação em Grafos"*, 16th Congreso Latino Iberoamericano de Investigación Operativa & 44th Simpósio Brasileiro de Pesquisa Operacional, CLAIO/SBPO 2012.

[32] M.R. Cappelle, L. Penso, D. Rautenbach, Recognizing Some Complementary Products, Theoretical Computer Science, to appear.

[33] P. Balister, B. Bollobás, J.R. Johnson, M. Walters. Random majority percolation, *Random Struct. Algorithms*. v. 36, p. 315–340, 2010.

[34] J. Balogh, B. Bollobás. Sharp thresholds in Bootstrap percolation. *Physica A*, v. 326, p. 305–312, 2003.

[35] J.-C. Bermond, J. Bond, D. Peleg, S. Perennes. The power of small coalitions in graphs. *Discrete Appl. Math.*, v. 127, 399–414, 2003.

[36] C. C. Centeno, M. C. Dourado, L. D. Penso, D. Rautenbach, J. L. Szwarcfiter. Irreversible conversion of graphs. *Theor. Comput. Sci.*, v. 412, p. 3693–3700, 2011.

[37] C. C. Centeno, L. D. Penso, D. Rautenbach, V. G. P. de Sá. Immediate versus eventual conversion: comparing geodetic and hull numbers in P 3-convexity. In: *Graph-Theoretic Concepts in Computer Science*, Springer, p. 262-273, 2012.

[38] P.A. Dreyer Jr, F.S. Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion, *Discrete Appl. Math.*, v. 157, 1615–1627, 2009.

[39] S.A. Cook, The complexity of theorem-proving procedures, in: *Proc. 3rd Ann. ACM Symp. on Theory of Computing Machinery*, New York, p. 151-158, 1971.

[40] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater. Fundamentals of domination in graphs. Marcel Dekker, 1998.

[41] D. Lichtenstein, Planar satisfiability and its uses. *SIAM Journal on Computing*, v. 11, p. 329-343, 1982.

[42] "L. D. Penso, F. Protti, D. Rautenbach, U. S. Souza", "On P_3-convexity of Graphs with Bounded Degree", "10th International Conference on Algorithmic Aspects of Information and Management, AAIM 2014", 2014.

[43] A. Hansberg, L. Volkmann. On graphs with equal domination and 2-domination numbers. *Discrete Mathematics*, v. 308, n. 11, p. 2277-2281, 2008.

[44] C. C. Centeno, L.D Penso, D. Rautenbach, V. G. P. de Sá. Geodetic Number versus Hull Number in P_3-Convexity. *SIAM Journal on Discrete Mathematics*, v. 27, n. 2, p. 717-731, 2013.

[45] A. Brandstädt and C.T. Hoáng, Maximum induced matchings for chordal graphs in linear time, Algorithmica 52 (2008) 440-447.

[46] A. Brandstädt and R. Mosca, On distance-3 matchings and induced matchings, Discrete Appl. Math. 159 (2011) 509-520.

[47] K. Cameron, R. Sritharan, and Y. Tang, Finding a maximum induced matching in weakly chordal graphs, Discrete Math. 266 (2003) 133-142.

[48] J.-M. Chang, Induced matchings in asteroidal triple-free graphs, Discrete Appl. Math. 132 (2003) 67-78.

[49] K. Cameron, Induced matchings, Discrete Appl. Math. 24 (1989) 97-102.

[50] K. Cameron, Induced matchings in intersection graphs, Discrete Math. 278 (2004) 1-9.

[51] K. Cameron and T. Walker, The graphs with maximum induced matching and maximum matching the same size, Discrete Math. 299 (2005) 49-55.

[52] K.K. Dabrowski, M. Demange, and V.V. Lozin, New results on maximum induced matchings in bipartite graphs and beyond, Theor. Comput. Sci. 478 (2013) 33-40.

[53] W. Duckworth, D.F. Manlove, and M. Zito, On the approximability of the maximum induced matching problem, J. Discrete Algorithms 3 (2005) 79-91.

[54] M.C. Golumbic and M. Lewenstein, New results on induced matchings, Discrete Appl. Math. 101 (2000) 157-165.

[55] Z. Gotthilf and M. Lewenstein, Tighter approximations for maximum induced matchings in regular graphs, Lecture Notes in Comput. Sci. 3879 (2006) 270-281.

[56] D. Kobler and U. Rotics, Finding maximum induced matchings in subclasses of claw-free and P_5-free graphs, and in graphs with matching and induced matching of equal maximum size, Algorithmica 37 (2003) 327-346.

[57] L. Lovász and M.D. Plummer, Matching Theory, vol. 29, Annals of Discrete Mathematics, North-Holland, Amsterdam, 1986.

[58] V.V. Lozin, On maximum induced matchings in bipartite graphs, Inf. Process. Lett. 81 (2002) 7-11.

[59] H. Moser and S. Sikdar, The parameterized complexity of the induced matching problem, Discrete Appl. Math. 157 (2009) 715-727.

[60] H. Moser and D.M. Thilikos, Parameterized complexity of finding regular induced subgraphs, J. Discrete Algorithms 7 (2009) 181-190.

[61] L.J. Stockmeyer and V.V. Vazirani, NP-completeness of some generalizations of the maximum matching problem, Inf. Process. Lett. 15 (1982) 14-19.

Dynamic Composition of REST services

Jesus Bellido

Abstract—Service composition is one of the principles of service-oriented architecture; it enables reuse and allows developers to combine existing services to create new services. Dynamic composition requires that service components are chosen from a set of services with equal or similar functionality at runtime. The adoption of the REST services in the industry has led to a growing number of services of this type, many with similar functionality. The existing dynamic composition techniques are method-oriented whereas REST is resource-oriented, and considers only traditional services. The REST architectural style has attracted a lot of interest from the industry due to the non-functional properties it contributes to Web-based solutions. In this thesis, we contribute to the area of web service composition in REST by proposing three techniques oriented to improve static and dynamic composition of this type of service. First we introduce a technique for static composition proposing a set of fundamental control flow patterns in the context of decentralized compositions of REST services. In contrast to current approaches, our proposal is implemented using the HTTP protocol and takes into account REST architectural principles. Afterwards, we present a technique to improve the dynamic composition in security domain extending ReLL to ReLL-S and allowing a machine-client to interact with secured resources, where security conditions may change dynamically. Finally, we propose SAW-Q, an extension of Simple Additive Weighting (SAW), as a novel dynamic composition technique that follows the principles of the REST style. SAW-Q models quality attributes, in terms of response time, availability and throughput, as a function of the actual service demand instead of the traditional constant values. Our results validate our main hypotheses indicating improvements with respect to alternative state-of-the-art methods. This also shows that the ideas presented in this thesis represent a relevant contribution to the state-of-the-art of REST service compositions.

Keywords—SOA, Web Services, Dynamic Compostion, REST, scalable architectures

1 INTRODUCTION

Web services are software entities that provide a determined functionality, they are platform-independent and can be described, published and consumed following a defined set of standards such as WSDL, UDDI and SOAP. These characteristics emphasize loose coupling between components and allow designers to develop interoperable, and evolving applications that can be massively distributed. Web services also promote the reuse of software, which reduces development costs, increases maintainability, and enables organizations to create composed services by combining basic and other composed services resulting in new software with aggregated value [1].

Service composition is performed statically if it occurs at design-time or dynamically if it occurs at runtime. Also, depending on how the composition is made, it can be manual or automatic. The dynamic and automatic service composition is an active research area due to the benefits of reduced need for pre-installation and configuration of the service composition, adaptability to change and contexts, high decoupling and personalization, smaller development time, and reuse [1]. However, static and manual techniques dominate in the industry due to the complexity of discovering services dynamically (WSDL descriptions lack domain semantics) and automatic composition. In addition, virtually all research on service composition focuses on the classic Web services (i.e. based on the WSDL, UDDI and SOAP standards). Currently, emerging services technology such as REST [1] is becoming an alternative to conventional services in the industry.

Unlike classical Web services which are centralized and operation-centric, the REST architectural style is resource-centric (e.g. `www.puc.cl/course/12`) that are manipulated through a limited set of standard and known operations (e.g.`HTTP. GET`, `POST`, `PUT`, `DELETE`). REST services are popular because they allow

• *J. Bellido is with the Department of Computer Science, Pontifical University, Chile.*
E-mail: jbellido@uc.cl

Manuscript received April 19, 2005; revised January 11, 2007.

1. Representational State Transfer.

massive scalability, decoupling, and fault tolerance (e.g. Amazon API, Facebook API, etc.). REST has attracted the interest of the scientific community to serve as a supporting framework for defining business processes and service composition. A REST resource should be able to be discovered by a machine at runtime, and it should be able to understand the mechanism of interaction with the resource (e.g. to use `GET` to read, and to use `DELETE` to delete it). This property would make possible to determine at runtime (dynamically) if the resource serves the consumer purposes, as well as to determine the order in which the operations, that change the resource state, should be invoked. Thus, it would be possible to generate workflows that implement B2B processes dynamically and automatically.

Currently, this is not possible because the REST resources lack a description that encapsulate domain semantics and can be interpreted by a machine. Moreover, current techniques of service composition are oriented to classic Web services that invoke an unlimited set of operations, rather than resources whose state are transformed by invoking a limited set of operations. The overall objective of this thesis is to contribute to the state of the art research on static and dynamic composition of REST services through the design and implementation of a dynamic composition model. In particular, we propose a RESTful decentralized, stateless composition model, a QoS model focused on security in order to determine the feasibility of service composition at runtime, and a QoS model focused on scalability, throughput and response time in order to dynamically select the best service component automatically.

2 Background

2.1 Service Oriented Architecture

Service orientation has existed for some time and has been used in various contexts with different purposes. The most used form of this term has been to identify approaches that focus on the separation of concerns, which means that the logic required to solve a large problem that can be developed and managed, if such logic is decomposed into a collection of related pieces, and each piece gives a solution to a specific part of the problem [2].

This approach transcends technology but when combined with software architecture, service orientation acquires a technical connotation. Service-oriented architecture (SOA) is a model in which an application's logic is decomposed into several smaller units that together implement a larger piece of business logic. SOA promotes that these individual units exist independently but not isolated from each other, in fact they could be distributed. This requires that these units operate on the basis of certain principles that allow them to evolve independently while maintaining uniformity and standardization among them. In SOA these logical units are known as services.

Web services encapsulate business logic and can provide solutions to problems of various sizes. The service logic may include the logic provided by other services, in this case, the service is called a *composed* service, and the logic providers are called *component* services. A component service may be responsible for a single or a complex task. Web services interoperate among them due to a common understanding provided by services' descriptions. The description of a service determines the service endpoint, the data received and the expected data returned by the service through message passing. In this way, programs or services use a description and a message channel independent from a protocol to interact and create a loosely coupled relationship.

In summary, SOA services maintain a relationship that minimizes dependencies (loose coupling), adhere to communication agreements (service contract), independently manage the logic they encapsulate (autonomy), hide logic that is not relevant to a determined context (abstraction), separate responsibilities in order to promote reuse (reusability), can be coordinated and assembled to form new services (composability), avoid to retain or store information specific to an activity (interaction stateless), and are designed to be described so they can be found and executed through discovery mechanisms (discoverability). The collection of services raises an inventory of services that can be managed independently [1].

SOA establishes an architectural model that aims to improve the efficiency, agility and productivity when developing software. It places services as first-class entities through which the business logics are implemented in alignment with service-oriented computing goals.

2.2 Web services

According to the context in which they are used, Web services may assume different roles: providers, clients and intermediaries. A Web service plays a provider role if it is invoked from an external source, and a client role if it invokes a provider service. Client services look for and evaluate which is the appropriate service to invoke based on the provider service description. Intermediary services are those which play a role of routing and processing messages sent between client and provider services until they reach their destination [1].

Traditional Web Services

The platform to implement Web services has been traditionally defined by a set of industry standards. This platform is divided into two generations, each associated with a set of standards and specifications [1]. The first Web services generation is comprised of the following technologies and specifications: Web Service Description Language (WSDL), XML Schema Definition Language (XSD), Simple Object Access Protocol (SOAP) and Universal Description Discovery and Integration (UDDI). These specifications have been widely adopted in the industry; however, they lack information about service quality, which is required to address mission critical functionality at production level. The second generation adds extensions (WS-* extensions) to fill the gap, related to service quality, left by the first generation. The main issues addressed by these extensions are service security, transactions, reliable messaging services, among others.

A traditional Web service is comprised of [1]:

- A service contract that is, a WSDL document describing the service interface (endpoint, operations and parameters) and an XML schema definition defining data types.
- The program logic. This logic may be implemented by the service itself, or inherited and wrapped by the service so that the legacy functionality can be consumed.
- Message processing logic consists of a set of parsers, processors and service agents. The runtime environment generally provides this logic.
- Software components that implement non-functional requirements defined by the WS-* standards extension.

REST Services

REpresentational State Transfer (REST) [3], [4] is an architectural style that underlies the Web. This approach is another way of implementing a service that follows the design constraints and requirements specified by this architectural style. Each constraint can have positive and negative impact on various non-functional attributes. The non-functional goals of REST are: performance, scalability, simplicity, modifiability, visibility, portability and reliability. An architecture that lacks or eliminates one of the REST constraints is usually not considered a REST architecture.

The REST architectural style constraints are:

- Client-Server: Enforces separation of responsibilities between two components of the architecture, the client and the server. This establishes a distributed architecture where each component evolves independently. This restriction requires the server to process the requests sent by the client.
- Stateless: Determines that the past communication between the client and the service are not kept stored on the server-side (service), that is, the interaction state is not remembered by the service. This implies that each client request must contain all the information necessary for the service to process the request and respond accordingly, without the use of session information.
- Cache: Services responses may be temporarily stored in a Web intermediary component (e.g. routers, gateways, proxies, etc.) and thus avoid the service to process a similar request again, diminishing the workload on the service-side.
- Uniform Interface: This constraint states that the architectural components must share a single interface to communicate, which is detailed below.
- Layered System. A REST-based solution can contain multiple architectural layers. These layers may be added, modified and rearranged according to the evolvability need of the solution.
- Code On Demand: This is an *optional* restriction that allows that some logic is dispatched from the server to the client, to be executed on the client-side. It allows to customize Web applications

The REST uniform interface is a set of architectural constraints that differentiates REST from any other style:

- Resources must be uniquely identified (e.g. through a URI), and the identifiers must be opaque to prevent coupling, that is, the structure of a URI shall not include any particular meaning that can be guessed by a customer.
- Resources' state must be manipulated through operations defined by standard (or ad-hoc) network protocols. For example, for the case of HTTP, the operations are `POST` that initialize the state of a resource whose identifier is unknown and could possibly create a subordinate resource, `GET` which obtains a representation including the current state of a resource, `PUT` that modifies the state of an existing resource, `DELETE` that indicates a request to eliminate a resource (but could be ignored by the server). `GET`, `PUT` and `DELETE` operations are idempotent and reliable. The `POST` operation allows an unsafe interaction since the invocation of this operation may cause changes to the server [3].
- A resource can support multiple representations that encode the state of the resource in a particular format (e.g. XML denoted by `application + xml`). The format can be negotiated through HTTP headers to facilitate interoperability between client and server.
- The HATEOAS (Hypermedia as the Engine of Application State) property indicates that the state transitions modeled in a Web application (e.g. buy a book, pay the bill, provide the deliver address, etc.) are served as hyperlinks that indicate the user the set of actions available to him or her at a given time (representation).

2.3 Service Composition

Software integration involves connecting two or more applications that may or may not be compatible, even when they are not built on the same platform or were not designed to interact with one another. The growing need for reliable data exchange is one of the strategic objectives of integration. Web services are inherently designed to be interoperable and are built with the knowledge that they will have to interact with a long-range of potential service consumers.

Web service composition is a technique that coordinates the combination of service components operating within an agnostic business process context. A composed service is comprised of component services that have been assembled in order to provide the functionality required to automate a task or a specific business process. These service components may be part of other compositions. The ability of a service to be naturally and repeatedly composable is essential to achieve the strategic goals of service-oriented computing.

When Web services are composed, the business logic of the composed service is implemented by several services, which allows the creation of complex applications by progressively adding components.

Service composition is associated with business processes automation. When a business process workflow is defined, several decision points are created in order to define the dataflow and actions according to variables and conditions evaluated at runtime. A business process definition can be done manually if a person defines the sequence of invocations, or automatically if an algorithm defines such sequence. In addition, service components can be chosen at design time (static composition) or at runtime (dynamic composition).

Orchestration and Choreography

In service composition, the coordination of service components invocation can follow two styles: orchestration or choreography. The orchestration is a centralized control of a business processes execution; it defines the business logic and the order of service invocation and execution. Unlike the orchestration, the choreography has a decentralized approach; services collaborate and determine its role in the interaction.

Orchestration is the process by which various resources can be coordinated to bring out the logic of a business process. Orchestration technology is commonly associated with a centralized platform for activities management [5]. Service orchestration encapsulates a service business process and other services invocations. The most widely used technology in the industry to implement orchestrations is the Web Service Business Process Execution Language (WS-BPEL) [6].

REST's navigational style naturally supports a workflow style of interaction between components.

However, interaction is decentralized, components are loosely coupled and can mutate at any time. This characteristic, called evolvability, poses a challenge to service composition since components (resources) may change unexpectedly. Hence, clients must make few assumptions about resources and must delay the binding with the actual resources up to the invocation-time (dynamic late binding) [7].

REST composition research focuses on orchestration, with JOpera [7] being the most mature framework. In JOpera, control and data flow are visually modeled while an engine executes the resulting composed service. In [8], control and data flow is modeled and implemented using a Petri Net whereas interaction and communication with the resources themselves is mediated by a service description called ReLL [9]. In [10], control flow is specified in SPARQL and invoked services could be WSDL/SOAP-based endpoints or RESTful services (i.e., resources); from the orchestrator perspective, services are described as graph patterns. In [11] resource's graph descriptions are publicly available (can be discovered using HTTP OPTIONS). A set of constraints regulates when certain controls can be executed on resources (e.g., a required state), so that an orchestrator engine could perform a composition, but no indication is given about how to express such constraints. The two former approaches support dynamic late binding and the hypermedia constraint. An RDF-based approach for describing RESTful services where descriptions themselves are hyperlinked is proposed in [12]. The approach is promising for service discovery at a high level of abstraction; however, no support for dynamic late binding is provided and the composition strategy is not detailed.

Choreographies can be described from a global and local (one party) perspective. WS-CDL [13] is a W3C candidate recommendation that describes global collaboration between participants in terms of types of roles, relationships, and channels. WS-CDL defines *reactive* rules, used by each participant to compute the state of the choreography and determine which message exchange will or can happen next. Stakeholders in a choreography need a global picture of the service interaction, but none of them sees all the messages being exchanged even though such interaction has an impact on related parties [14].

In [15], REST-based service choreography standards evolution is presented. Business processes, modeled as choreographies, are implemented as single resources at a higher level. The lower level comprises the resources themselves (process instances). Although the approach provides process state visibility, it is not clear whether the higher level corresponds to a centralized orchestration or to a partial view of choreography.

2.4 REST services composition

Resources and resource collections are the components in a RESTful scenario [7], [16]. Unlike WSDL-based services, REST resources have standardized, few, and well-known assumptions at the application level (i.e. the Web) instead of the domain level. Resources must be identified with a URI (Universal Resource Identifier) that binds the proper semantics (at the application level) to the resource ([3] section 6.2.4), and must be manipulated through *links* and controls (i.e. an HTML form) embedded in a resource's *representations* (e.g., HTML page). Representations dynamically determine the set of resource's state transitions available to consumers (Hypermedia as the engine of application state [3]).

Composition requirements that are specific to REST are *dynamic late binding*, that is, the URI of the resource to be consumed can be known only at runtime; the composed service must also support the REST uniform interface; data types are known at runtime; content negotiation must be allowed; and clients must be able to inspect the composed service [16], [17].

Currently there are no proposals for automatic and dynamic composition of REST services. REST services composition research is mainly focused on static service composition [16], [17]. The most comprehensive work in the area is the JOpera project [18] that aims to provide a similar implementation of BPEL for REST. JOpera allows the static composition of Web services by means of two graphs. The first, Flow Control Dependency Graph (CFDG) describes the sequence of invocation of services, and the second, the Data Transfer Flow Graph (DFTG) defines the relationship between the data inputs and outputs when invoking the REST components. Unlike BPEL, service composition in JOpera allows that the service URI to be invoked is known at runtime (dynamic late binding), it also suports the uniform interface, and content negotiation at runtime. However, JOpera does not consider the HATEOAS constraint, and the CFDG is performed by a centralized process engine, in a stateful fashion which negatively impacts on service scalability.

Similarly, Bite [19] proposed a BPEL-inspired composition language describing data and control flow. Bite partially supports a uniform interface based on HTTP, dynamic data types, and state inspection (only `GET` and `POST`). Regarding dynamic late binding, Bite can generate URIs to created resources, but cannot inspect the service's responses and find out the links provided by the service (HATEOAS).

Decker [20] presents a formal model for implementing REST processes based on Petri nets, they uses PNML (Petri Net Markup Language) as a language for specifying the states, transitions and an execution engine. Decker considers only partial support for dynamic late binding since it can generate URIs for resources created but cannot inspect the responses and retrieve the embedded links. In this model, control flow is driven by the decisions of a human user. Guard conditions, such as autentication are not supported and like others [21] it assumes XML as the content type for representations leaving out REST content negotiation constraint.

Zhao [22] presents an automatic service composition approach for REST typifying and semantically describing the services in three categories according to the operations that the service can perform (`GET`, `PUT`, `POST`, `DELETE`). Service composition is automated by a first-order logic situation calculus representing changes and actions. This approach does not consider several of the REST principles such as HATEOAS, content negotiation, opaque URIs, nor dynamic late binding.

2.5 QoS

The quality of service (QoS) is a combination of several qualities or properties of a service [23]. Dynamic composition of Web services requires the consumer to choose the best component services that satisfy the functional and non-functional requirements of the composition. Non-functional requirements involve attributes of quality of service as response time, availability, security and performance [23].

The definition and measurement of these attributes of service quality vary by approach. For example, the response time can be measured as the average of the results obtained of the last 15 minutes, or an average vector of 15-minute intervals during the day.

The quality attributes of a service depend on the load current at which the service is submitted, however the technical description of these quality attributes of a service are focused on the description of a discrete value associated with each property.

QoS in REST

For the case of REST, research initiatives on service composition are fairly recent, and interest on QoS properties have focused mainly on security. For instance, in [24] a RESTful service API is defined to support service-level agreements based on the WS-Agreement standard. Agreements (and templates) are REST resources encoded in the application/xml media-type, whose life cycle and state is handled by means of HTTP verbs. Graf et al. present a server-side authorization framework based on rules that limit access to the served resources according to HTTP operations [25]. In [26] a server-side obligation framework allows designers to extended existent policies with rules regulating users' information access. Rules may trigger additional transactions (e.g., sending a confirmation e-mail, register the information access attempt in a log) or even modify the response content or the communication protocol (e.g., require HTTPS). Allam [27] aims to homologate WSDL-based and RESTful services by considering them black boxes, where interaction occurs as simple message passing between clients and servers. Security policies can be placed when receiving or sending a message as well as locally (e.g., at the server or client side). This vision does not consider complex interaction involving third parties (e.g., OAuth), or service's interface heterogeneity [28], where industry standards are implemented in various ways. We assume a workflow perspective where data is transformed locally in successive steps until the message constraints required by the service provider are achieved. In addition, clients, servers, and third party services may engage in an interaction that implements sub-workflows.

3 REST LIMITATIONS

The REST architectural style offers several advantages when compared to traditional Web services in terms of non-functional attributes, namely, low latency (small response time), massive scalability (due mainly to statelessness, cacheability, and Web intermediaries), modifiability, and simplicity, among others. However, REST considers humans as its principal consumer and they are expected to drive service discovery and state transition by understanding the representation content. The lack of a REST machine-readable description forces service providers to describe their APIs in natural language, which makes difficult to properly design machine-clients that can perform discovery and service composition in an automated way.

Furthermore, REST has been used in the industry as an infrastructure layer for supporting massive service provisioning in the form of Web APIs which have given rise to the evolution of application ecosystems that constitutes simple service composition basically consuming services in a sequential control-flow pattern. A massively used example of complex control-flow is the OAuth protocol that constitutes an orchestration where various parties cooperate through redirections in order to implement a workflow. The control-flow patterns involved in the OAuth are sequence and conditional execution. Researchers also recognize the lack of a complex service behavior model in REST as one of the difficult issues to be addressed in order for REST to support rich SOA (Issarny, 2009). Research proposals for REST service composition focus either on operation-centric [29], centralized [7], stateful and static service composition, violating REST architectural constraints.

Traditional Web services dynamic and automatic composition focuses on diverse techniques (e.g. planning, graph models, semantic models, QoS constraints, etc. in order to make possible the automatic or dynamic generation of a composition plan (i.e. a workflow) and the selection of the services components [30]. QoS has been a focus of extensive research and is being recently addressed in REST but not in its capacity for determining service composition on runtime. Furthermore, the very nature of the non-functional attributes makes it hard to reduce them to uniform representations such as numbers or Boolean values. Security for instance requires not only a complex combination of algorithms but also protocols (in the case of OAuth a choreography) in order to determine whether a service can be composed or not. For the case of response time, availability and throughput, research in Web sites provisioning, also known as capacity planning, have demonstrated the variables such as the user demand and cache policies are relevant to determine the quality of a Web site (equivalent to a single REST service). Hence such quality attributes cannot be reduced to simple numeric variables as is the case of traditional QoS-aware compositions in SOA.

4 General Goals

The overall objective of this thesis is to facilitate the dynamic composition of REST services. In particular, we propose:

1) A decentralized RESTful, stateless composition model that places an emphasis on service behavior (control-flow),
2) A QoS model focused on security in order to determine the feasibility of composition at runtime, and
3) A QoS model focused on scalability, throughput and response time in order to select the service component automatically at runtime.

5 Hypothesis

The main hypothesis of this thesis is related to the relevance of decentralized service composition, in that *"a stateless and decentralized composition technique follows the REST architectural style constraints and generates a composite service with the same REST properties whereas a centralized composition does not"*. Second, *"QoS-based dynamic and automatic RESTful service composition must take into account the characteristics of the non-functional attributes in order to preserve REST architectural constraints in the composite service"*.

6 Thesis Work and Main Contributions

The first part of this thesis presents a technique for decentralized, hypermedia-centric and stateless REST services composition. The proposed technique models services behavior through a set of well-known, simple and complex control-flow patterns and focuses on the following REST constraints, namely, client-server interaction, layered client-server, client-stateless-server, and the uniform interface. The uniform interface (messages between server and client are self-descriptive) was extended through `300 HTTP` redirection codes in order to include control-flow information in the message; hypermedia (Web linking) is used as the application state machine. We present the advantages of this approach (centralized Vs. decentralized) in terms of response time, availability and throughput, which are non-functional goals in REST. By definition a stateful approach is not RESTful so that such comparison is left out.

The second part of this thesis describes automatic and dynamic REST services composition based on non-functional attributes. In this part two QoS domains are analyzed, the first corresponds to security and the proposal is a hybrid between static and dynamic service composition in the sense that a service description (created at design-time) is used to determine the feasibility of service composition (at runtime) and actually enforce the composition (at runtime). The environment is decentralized, stateless and promotes the use of hypermedia as a state machine.

The third part of the thesis focuses on a second QoS domain considering scalability. That is, the non-functional attributes: response-time, throughput and availability were used to support automatic and dynamic service composition. In this case, a queuing theory-based model was used to identify response-time, throughput and availability. These models were later used in a technique called SAW-Q to identify the compositions with the highest quality. SAW-Q was compared with a well-known technique, SAW (from which SAW-Q was derived) with good results.

Accordingly, the main contributions of this thesis are:

1) Part 1
   a) A decentralized, stateless, hypermedia-centric technique for designing composed service behavior in REST.
   b) A set of control-flow patterns that implement decentralized, stateless, hypermedia-centric REST service composition.

   These contributions were published in the following journal:
   J. Bellido, R. Alarcon and C. Pautasso. *Control-Flow Patterns for Decentralized RESTful Service Composition*. ACM Transactions on the Web 8:1 (5:1âĂŞ5:30). ACM 2013.
2) Part 2
   a) A centralized, hybrid (design-time and runtime) and manual REST service composition based on the security QoS attribute.
   b) A security domain model.
   c) An extension to ReLL, a hypermedia REST service description, that considers the security domain model in order to determine the feasibility of consuming a protected service and actually execute the OAuth choreography:

   These contributions were published in the following journal:
   C. Sepulveda, R. Alarcon, and J. Bellido. *QoS aware descriptions for RESTful service composition: security domain*. World Wide Web 1-28, 2014
3) Part 3
   a) A decentralized, dynamic REST service composition technique based on the response-time, throughput and availability QoS attributes.
   b) A queuing theory-based REST model.
   c) A response-time, throughput and availability models based on the proposed queuing model.
   d) SAW-Q, a refinement of SAW, a technique for scoring service compositions in terms of various variables such as response-time, throughput and availability that is sensible to user demand and service capacity.

   These contributions were submitted to the following journal:
   J. Bellido, R. Alarcon, and C. Pautasso. *SAW-Q: An approach for dynamic composition of REST services*. IEEE Transactions on Services Computing, 2014.

Other articles were also produced during this investigation:

- R. Alarcon, E. Wilde and J.Bellido. *Hypermedia-driven RESTful service composition*. Service-Oriented Computing. Springer Berlin Heidelberg, 2011. 111- 120.
- J.Bellido, R. Alarcon and C. Sepulveda. *Web Linking-based protocols for guiding RESTful M2M interaction*. Current Trends in Web En-

gineering (pp. 74-85). Springer Berlin Heidelberg, 2012 15

The remainder of this document is organized as follows: Section 7 presents the journal that summarizes the results of the first part of the thesis. Section 3 explores the attributes of service quality in the RESTful services composition, focused on security. Section 8 presents the dynamic composition approach based on SAW-Q and scalability (response-time, throughput, availability). Section 8 presents the main conclusion of this thesis and future research.

7 Control Flow Patterns for Decentralized RESTful Service Composition

Describes a technique for decentralized REST services composition that takes into account the constraints of REST architectural style in the composition process. The proposed technique involves the creation of control-flow patterns that allow seamless interaction between the client and the composite service and uses hypermedia as a state machine. The main idea is to implement control-flow patterns through callbacks and redirections. Finally, we discuss the impact of our design decisions according to the architectural principles of REST.

The redirection code (`303`) was designed to inform the user agent it must fetch another resource, and it is widely used for services to interact with other services and accomplish business goals. For example, OAuth and OpenID are popular authorization and identity protocols implemented using redirection codes; payment entities which offer online transactions are also implemented using redirection codes in order to allow e-commerce applications to sell products online in a security context under their control.

Due to constraints on the `303` redirection code, it cannot support complex interaction successfully. For instance, parameters should be serialized in a text format and concatenated to the URI (`application/x-www-form-urlencoded`), and information that cannot be serialized as plain text cannot be passed between applications in the URI parameters (e.g., images, pdf documents, etc). The resulting URI must not exceed the limit established by the server, otherwise the server should return a `414 Request-URI Too Long` status code message. In order to send large quantities of data, the media type `multipart/form-data` and the `POST` method shall be used for submitting forms that contain files, non-ASCII data, and binary data. In addition, only the `GET HTTP` method can be used to automatically fetch the redirected URI, but as seen in our example, applications may be required to interact with each other using additional methods without requiring end-user confirmation (e.g., `POST` and `PUT` messages 3 and 10 in Figure**??**).

More importantly, control flow may be more complex than sequential invocation of REST resources. Business processes also require parallel or alternative invocation as well as determining the conditions for choosing the right response; more complex control flows consider the invocation of two services in non established order but only one at a time (unordered sequence), or service invocation for a determined number of times iterator.

Finally, notice that the composed REST service (`PO` and `/stateN` resources) encapsulates the knowledge about which services to invoke (URI), which parameters or state information should be sent and be expected to be received, which methods shall be used (e.g., `GET` (7) or `POST` (12) in Figure **??**), as well as the order of the invocation; that is, they must know the service interface of the resource, which in our case is accomplished through ReLL.

7.1 Control flow patterns

In the context of stateless, decentralized compositions of services described with ReLL and with the assumption that clients can process the Callback link header, in this section we model control-flow patterns for RESTful service composition and the `HTTP` protocol. The set of patterns includes sequence, unordered sequence invocation, alternative, iteration, parallel, discriminator, and selection [31], [32].

Sequence, Unordered Sequence

The sequence pattern is a basic control-flow structure used to define consecutive invocations of services which are executed one after the other without any guard condition associated. As seen in figure **??**, this pattern could be implemented using the `303` redirection code; however, only automatic redirection of `GET` messages are allowed by the standard, making it difficult to update the composed resource state (i.e., PUT message of lines 5, 9). In addition, there is no clear indication on how to handle the payload of the message. We extend the status codes with a set of codes identified with a two digit prefix: 31x. The sequence pattern is implemented with a new code: 311 (Sequence) indicating the invocation of a service without any guard condition (see Figure 1).

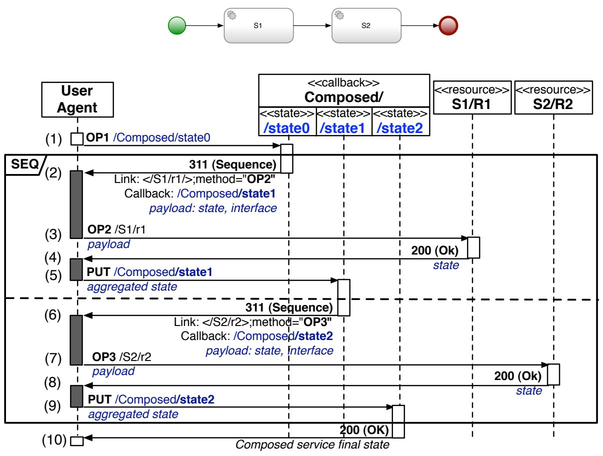


Fig. 1. A sequence control flow pattern implemented for REST and HTTP

The server responds with a 311 message including the component resource address (2, 6) in a Link header as well as the HTTP method in a link target attribute, and a Callback address in an additional header indicating the state of the composition. Additional information such as state (e.g., a digested value) and, depending on the service interface, data formats or URI schemes to create the request, can be included in the payload. Actual data values for such templates shall be provided by the user agent either by requesting them to the user through an appropriate interface or retrieving them from the local storage. Such process is out of the scope of this proposal. The server may close the connection with the client after issuing a 311 message unless metadata indicating otherwise is included. When a user agent receives this code, it must store locally the callback address and automatically request the component resource using the indicated method (3, 7). Similarly to Figure **??**, if additional communication shall occur between the component resource and the user agent it must be modeled as out-of-band communication and is omitted for readability. When the response is available, the component replies with a 200 status code. The composed service shall not issue another request until the response has been passed by the user agent through a PUT message (5), then the composed service can proceed with the next component (6 to 9).

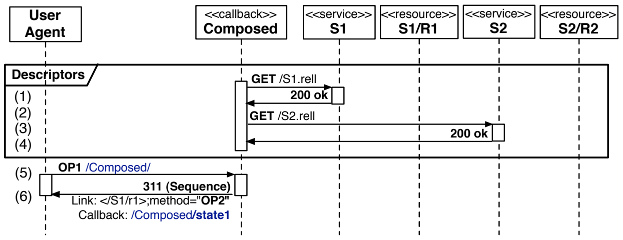


Fig. 2. ReLL descriptors are fetched considering the root resource of a service

When all the components have been fetched (i.e., the final state of the sequence has been reached), the response is provided with a 200 status code and the composed service representation (10). Notice that the actual HTTP methods to be used when invoking component services must be determined by the composed resource. In order to know how to handle the resources, the composed service prefetches the component services descriptors which detail the interface of a set of resources at domain-level; the descriptors are themselves REST resources (Figure 2). This phase is omitted in the figures detailing the remaining patterns for readability, although it is assumed it takes place before invoking a component resource.

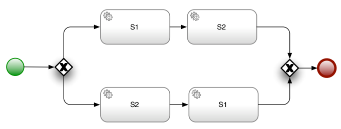


Fig. 3. Unordered Sequence

For the case of the unordered sequence pattern, it specifies the execution of two or more services in any order but not concurrently (Figure 3). That is, given two services S1 and S2, the services execution can result as S1 followed by S2 or S2 followed by S1. Since the list of services to be invoked is known by the composed resource and the order is irrelevant, the composed resource (server) has the information to decide which service to invoke as part of its own logic. For the user agent, all that matters is the address of a particular component resource to be invoked as well as the method; that is, this case is not different from a sequential invocation.

Alternative, Exclusive choice

The alternative pattern is a basic control-flow structure that allows the execution of only one of two possible services. The service to be executed could depend on the outcome of preceding service execution, values of the parameters, or a user decision. The `312 (Alternative)` status code is proposed for this pattern. When a composed service requires executing one of two services, it responds to the client request with a `312` coded message, indicating the list of services to choose as `Link` headers, including the HTTP `method` as an attribute, and a `Callback` header indicating the connector state to resume interaction. The message payload is a conditional expression to be evaluated by the user agent, as well as information required to build proper request messages (i.e., data formats or URI schemes).

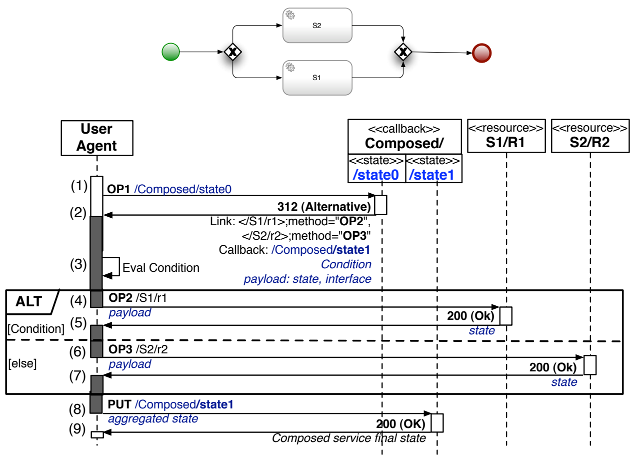


Fig. 4. An alternate control flow pattern implemented for REST and HTTP

The composed resource closes the connection after issuing the response unless otherwise indicated by additional headers. Link services may differ on the resources (URIs), or the methods to be used (Figure 4, message 2). Since in REST user agents keep application-state information [33], they shall have enough information to perform the evaluation. A good practice is to express the condition in languages well-known to the Web, such as XPath, although its format escapes the scope of this thesis. Once the user agent has evaluated the condition it determines which link to follow (4 or 6). Again, additional communication may occur between a user agent and an origin server. Eventually when the component has a final response, it issues a `200` coded response including its state in the payload (5 or 7). This causes the user-agent to send an update message (`PUT`) to the composed resource carrying on the received payload (8). Once the interaction finishes, the composed resource replies with a `200` message including the representation of its final state (9).

Iteration, Structured Loop - while variant

This pattern is an advanced control-flow structure that allows the execution of a service repeatedly, the number of times depending on a fixed number, a time frame, etc. which can be modeled by a conditional expression. We propose the `313 (Iteration)` status code for representing iterations.

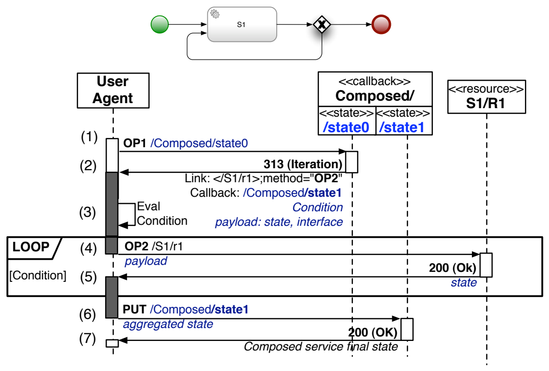


Fig. 5. An iterator control flow pattern implemented for REST and HTTP

This interaction begins when the composed resource issues a `313` message (Figure 5, message 2) including a `Link` header with the address of the component resource, a `Callback` header indicating the callback connector state address, a conditional expression, and additional information to create the message request as part of the payload. After evaluating the conditional expression (3) and obtaining positive results, the message is redirected to the component resource using the indicated operation and payload (4). Communication between client and server may include several messages interchanged. When a response is available, the component resource will issue a `200` message (5). The condition will then be evaluated again. If it still holds, the component is invoked again (4);

or a `PUT` message is sent to the callback address carrying along the response content served by the component service aggregated with previous state information (6). Finally, at the end of the interaction, the component replies with a `200` message and the representation of the composed resource final state (7).

Parallel Split - Synchronization, Structured Discriminator, Structured Partial Join, Local Synchronization Merge (Selection)

The Parallel Split is a simple pattern that allows a single thread of execution to be split into two or more branches invoking services concurrently. The parallel split pattern can be paired with either one of four advanced control-flow structures. Under the paradigm of a composed service - component services, it is the former which determines whether it waits for all the responses (Synchronization, Figure 6.a), just one of them (Structured Discriminator, Figure 6.b), or a fixed number (Structured Partial Join, Figure 6.c). Finally, for the case of Local Synchronization Merge (also called Selection, Figure 6.d), the composed service shall wait for a number of responses that cannot be determined with local information.

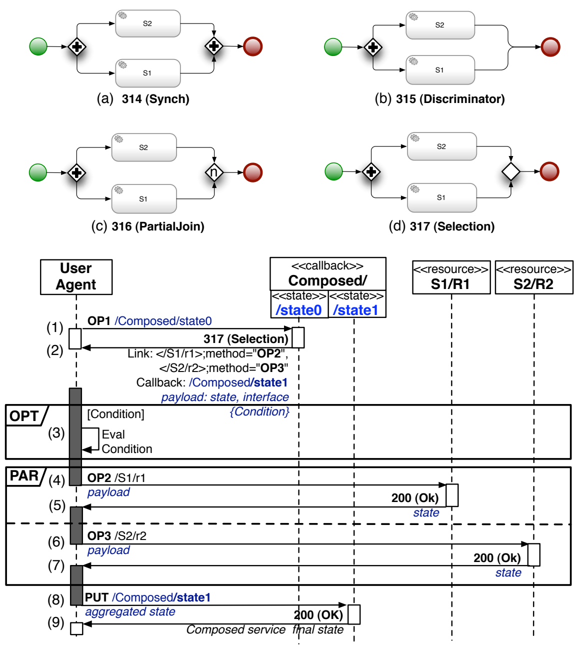


Fig. 6. A parallel control flow pattern implemented for REST and HTTP

In order to avoid violating the REST stateless principle, servers do not store information about how many answers are expected per client, but make explicit server's expectancies through the pattern status codes, `314 Synch` (Synchronization), `315 Discriminate` (Structured discriminator), `316 PartialJoin` (Structured Partial Join) and `317 Selection` (Local Synchronization Merge). The four patterns follow the same conversational structure; however, the client's decision to inform the server about the availability of a final response is affected by the corresponding pattern.

Figure 6 shows the details for the pattern. Interaction starts when the composed resource issues either a `314`, `315`, `316` or `317` message (Figure 6, message 2). The message includes a list of `Link` headers annotated with a `method` attribute, a `Callback` header indicating the callback connector state address, and a payload with instructions to format input data for the operations according service interface. It may also include state information such as the number of expected service components to be addressed by the client for the case of the `316 Partial Join` pattern.

For the case of a `317 Selection` message, a conditional expression must be included. The condition must be evaluated considering application-state information stored locally at the client side (3), and the result shall be the number of request messages the client must issue to service components.

Once the user agent determines how many responses to provide to the composed resource (all, one, n out of m, or n), it invokes all the service components indicated in the list with the appropriate methods concurrently (4, 6). In practice the number of links to be fetched is limited by the number of concurrent connections the client is able to maintain with the servers involved. Again, there may occur several messages interchanged between clients and origin servers as an out-of-band conversation, but once the final response is available, it is aggregated until the number of responses expected to be sent to the composed service is reached. The aggregated state is sent as a `200` coded request (8). The composed resource processes the aggregated state (e.g., it could merge the results) and issues a `200` coded response with the final state.

8 QoS AWARE DESCRIPTIONS FOR RESTFUL SERVICE COMPOSITION

We explore QoS aware RESTful services composition, which is characterized by a decentralized, stateless and hypermedia-driven environment. We focus particularly on the security domain since current security practices on the Web illustrate the differences between both the centralized, function-based approach and the decentralized, hypermedia and resource-based approach. We rely on ReLL (a REST service description) that can be processed by machine-clients in order to interact with RESTful services. Our approach identifies key security domain elements as an ontology. Elements serve to model hypermedia-based, decentralized security descriptions supporting simple and complex interaction such as protocols and callbacks. In this paper, we propose an extension to ReLL that considers security constraints (ReLL-S) and allows a machine-client to interact with secured resources, where security conditions may change dynamically. A case study illustrates our approach.

8.1 ReLL-S security description

Compared to traditional services, security is addressed in a different way on the Web. In Maleshkova et al. [34] a review of the self-declared REST Web APIs corresponding to the ProgrammableWeb site [2] was analyzed regarding their support of security mechanisms. The Maleshkova et al. study analyzes a Web site that is a well-known repository for services (including WSDL, REST, XML-based, Web APIs, etc.). They picked 222 services from the RESTful service category (18%) covering the 51 service categories (e.g., search, geolocalization, etc.) available. The security mechanisms found range from very simple practices (the majority), to the W3C standards on security on the Web. For instance, 38% uses API Keys whilst the OAuth protocol is used by 6% of the reported service. Notice that mainstream so-called REST services (e.g., Facebook, Twitter, Google, etc.) support and require OAuth authentication. Therefore, we believe the study is representative from a practical (and informal) and a theoretical (standards) point of view. In this section we present ReLL-S descriptions supporting each of the security mechanisms identified in the Maleshkova et al. survey.

2. http://www.programmableweb.com

8.2 OAuth

OAuth [35] is an Open Authorization protocol that allows a third-party application - a client application - to access resources provided by a service - resource server - and owned by a user. The user has to authorize the third-party application to access the resources stored by the resource server, without exposing his or her authentication credentials, to the third-party application. The authorization grant is represented as a token.

OAuth defines four grant types: authorization code, implicit, resource owner password credentials, and client credentials; and provides an extension mechanism for defining additional grant types. The protocol flow is flexible and depends on the type of authorization that is going to be granted. So in the flow shown in Figure 7, up to four parties could be involved: the resource owner, the resource server, the authorization server, and the client. The result of the protocol is an access token that represents the authorization granted by the resource owner and that is sent later by the client to the resource server to access the restricted resources.

What is interesting about this protocol is that it involves more than two entities that can communicate using a protocol that has been designed for client-server communication, as it is HTTP, in a stateless one-to-one conversation. However, this characteristic presents particular issues to deal with, which makes it really interesting to describe in more detail.

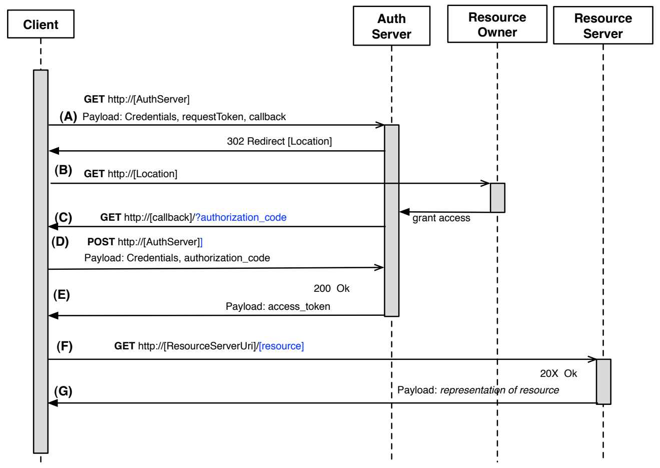

Client
Auth Server
Resource Owner
Resource Server
GET http://[AuthServer]
(A) Payload: Credentials, requestToken, callback
302 Redirect [Location]
(B) GET http://[Location]
grant access
(C) GET http://[callback]/?authorization_code
(D) POST http://[AuthServer]]
Payload: Credentials, authorization_code
200 Ok
(E)
Payload: access_token
(F) GET http://[ResourceServerUri]/[resource]
20X Ok
(G)
Payload: representation of resource

Fig. 7. Abstract OAuth interaction between four parties

Once the client has required the access grant, the authorization server starts a conversation with the resource owner that, from the client's perspective, occurs completely out of its control. This conversation's goal is asking the user to give authorization to access the resources. It could be implemented many ways; the authorization server could send an email to the user, an SMS, or any other kind of conversation. In most implementations the conversation occurs synchronously by redirecting the user-agent from the client domain to the authorization server domain. So that, to restore the interaction between user and client, the authorization server must also redirect the user-agent to the client. If the user grants access rights to the client, such redirection message will contain an authorization grant. This conversation implements an asynchronous conversation through a callback connector provided by the client, which becomes the target of the redirection.

Once the client has the authorization grant, it can use it to retrieve an access token from the server. The client sends its credentials and the authorization grant, representing the permissions granted by the user, to the server. The server verifies the permission and generates a final token (access token, or oath token) to be used in future calls, so that the resource server allows access to restricted resources. It could have an expiration time and some authorization servers provide the feature to refresh or renew the token.

Listing 1. An authorization constraint specification (OAuth)

```
<scope resource="restrictedResources">
  <constraint id="oauthAuth"
     type="Authorization">
    <accessToken>
      <query_param name="auth_token"
         select="$auth_token||wl:OAuth"/>
    </accessToken>
  </constraint>
</scope>

<protocol name="wl:OAuth">
  <invoke
     url="http://www.authserver.com/oauth"
     pre="not($auth_code)">
    <query_param select="$state"
       name="extra"/>
    <query_param select="$callback"
       name="callback"/>
  </invoke>
  <invoke
     url="http://api.service.com/getToken"
     pre="$auth_code">
    <query_param select="$auth_code"
       name="auth_code"/>
    <store selector="token"
       persist="auth_token"/>
  </invoke>
</protocol>
```

In this case the client makes two calls to get authorization to the user's restricted resources. The first one could include the callback address and also a parameter, named `state`, that must be sent back by the authorization server when issuing the callback request to the client. The client can use the parameter to keep the flow state. The callback call will include also the authorization code used by the client to get the final authorization token during a second call. The order of both calls is defined by their preconditions (i.e. they play the roles of guard conditions). That is, when the client runs the protocol, it will invoke the first call initially because it doesn't have the `auth_code` stored locally. Once the request is issued, the client blocks itself waiting for an answer. When the callback arrives, the client is restarted and since its current condition has changed (i.e., it has the code) then it will make the second call (the condition is met) to finally resolve the authorization constraint.

9 SAW-Q: An approach for dynamic composition of REST services

SAW-Q: An approach for dynamic composition of REST services: We propose SAW-Q an extension of SAW, as a technique of dynamic composition according to the principles of the REST style and considering quality attributes modeled as a demand function instead of the traditional constant values. Our model is much more accurate when compared to real implementation, positively improving dynamic composition.

9.1 Modeling REST QoS service using Queue Models

There are multiple ways to measure the performance of a service. The most common metrics are response time, availability and throughput. A service response time is the time interval from the request arrival to the server until the response is received by the client and is determined by two factors: the network latency and bandwidth and the service capacity. The way a REST service processes requests and replies can be modeled using queuing theory. The client requests follow the Poisson distribution [36], [37] and are randomly distributed over a period of time. The generalized queuing model considers the long-term behavior of the queue or steady state, reached after the system has been running for a sufficiently long period of time.

Performance measures of a queue are based on the probability that a certain number of requests may exist in a system at a given time (p_n). For instance, p_0 would represent the probability of a minimal waiting time due to an empty queue (0). More in detail, useful performance measures are[37]: the expected amount of requests in the system (L_s), which includes the expected number of requests from customers in the queue (L_q), the waiting time of a request to be processed by the service (W_s) and the waiting time of the request in the queue (W_q).

Some specializations of the generalized queue system that calculates performance measures have been proposed and proved. The specialized queue model corresponding to a REST service is $M/M/C : GD/N/INF$, where $N \geq C$ as described before. The corresponding performance measure formulas originally defined in [37] are described below.

Given the estimated rate of requests (λ) that arrive at the system (S), the rate of denied request depends on the rate of arrival at a given time (Formula 1). The higher the arrival, the higher the probability of denied requests.

$$\lambda_{lost} = \lambda \cdot p_N \quad (1)$$

The effective rate of arrival is the difference between the arrived requests minus the denied requests (Formula 2)

$$\lambda_{eff} = \lambda - \lambda_{lost} = (1 - p_N) \cdot \lambda \quad (2)$$

The rate between arrival (λ) and processed requests rate of the system (μ) is defined by $\rho = \lambda/\mu$. It is possible to calculate the expected number of requests waiting in the queue $L_q(s)$ using Formula 3 [37] , and the expected number of requests in the $L_s(s)$ system using Formula 4.

$$L_q(s) = \frac{\rho^{c+1}}{(c-1)!(c-\rho)^2} \Bigg\{ 1 - \left(\frac{\rho}{c}\right)^{N-c+1} - (N-c+1) \Bigg(1 \quad (3)$$

$$L_s(s) = L_q(S) + \rho \quad (4)$$

The processing time for a REST service is defined as a rate between the number of processed request and the effective arrival rate of requests (Formula 5).

$$W_s(s) = \frac{L_s(s)}{\lambda_{eff}} \quad (5)$$

The waiting time in the waiting queue for a request in a REST system is defined as a rate between the number of requests waiting in the queue and the effective arrival rate of requests (Formula 6).

$$W_q(s) = \frac{L_q(s)}{\lambda_{eff}} \quad (6)$$

The average service response time $Q_{ResponseTime}(s)$ is calculated as the sum of the processing time of the services (W_s) and the waiting time in the queue of requests (W_q), as seen in Formula 7. We are not considering network latency since this is out of the scope of responsibility and control of the service.

$$Q_{ResponseTime}(s) = W_s(s) + W_q(s) = \frac{L_q(s)}{\lambda_{eff}} + \frac{L_s(s)}{\lambda_{eff}} \quad (7)$$

Replacing the factors in the formula in order to reflect the clients demand dependency, we obtain the following Formula 8.

$$Q_{ResponseTime}(s) = \frac{L_q(s)}{(1-p_N)\lambda} + \frac{L_s(s)}{(1-p_N)\lambda} \quad (8)$$

Service processing capacity is defined as the number of requests that a system (the whole number of service replicas) can process at a given time. The service availability $Q_{Availability}(s)$ is usually measured as the percentage of time that the service is ready to be consumed at a period of time [37]. However, the availability of a service depends on the number of customers attempting to access a service. If the number of requests (n) exceeds the capacity of the service ($C \leq n < N$), the availability is downgraded because many request try to access and compete for the service resources, hence the probability of being served is reduced. Otherwise, if the number of requests (n) falls below the service capacity ($0 \leq n < C$), the availability of the service grows. Service availability is calculated as the probability that n requests exist in the system (p_n). Service availability is calculated using Formula 9.

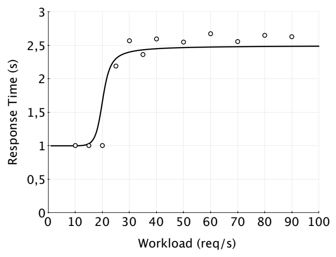
$$Q_{Availability}(S) = \begin{cases} \frac{\rho^n}{n!} p_0, & 0 \leq n < c \\ \\ \frac{\rho^n}{c! c^{n-c}} p_0, & c \leq n < N \end{cases} \tag{9}$$

The service throughput $Q_{Throughput}(s)$ is the number of requests that the service can process in a unit of time (i.e. seconds). If $Q_{ResponseTime}$ is the function to calculate the time that takes a request to be processed by the service, then the quantity of requests that can be processed by the service in one second is calculated as the inverse function of $Q_{ResponseTime}$, as seen in Formula 10 .

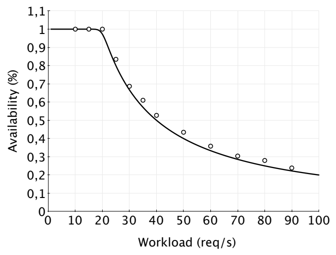
$$Q_{Throughput}(S) = \frac{(1 - p_N)\lambda}{L_q(S) + L_s(S)} \tag{10}$$

9.2 Applying the Queue Model in a real scenario

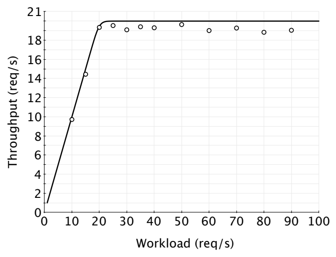
We implemented a REST service as a Python script running on a Apache Web server. The experiment was performed locally, in order to discard the effect of network latency, on an Intel PC with 4GB RAM, and 4 cores. We also modeled response time, availability and throughput using the equations proposed in section 3.4. The implemented service is characterized as $W_s = (1, 20, 50)$ where each instance is able to process one request per second, the service is replicated 20 times and there may be up to 50 requests in the system at a given time. The response time was calculated using the proposed formulas and we obtained the experimental results from the stress tests using Apache JMeter, the tests were run 10 times automatically and averaged, in order to eliminate external factors. For each scenario, results between the models and the experiments are very similar, for the case of response time the results show that when the number of requests exceeds the processing capacity of the service, the response time increases until the requests are denied (Figure 8(a)); service availability decreases exponentially (Figure 8(b)), and throughput remains constant once the service reaches the maximum capacity (Figure 8(c)).



(a) Response Time



(b) Availability



(c) Throughput

Fig. 8. Comparison between estimated (continuous line), using the proposed formulas, and actual (shown as dots) behavior of a service in different scenarios. (a) The average response time of a service increases when the workload exceeds the capacity of the service (20 req/s); (b) service availability is reduced when the service is unable to process more requests (20 req/s); (c) and service throughput grows according to the workload until reaches its maximum capacity (20 req/s).

9.3 Dynamic Composition based in QoS using Queue Model

In order to compose a service dynamically and based on the quality attributes, we need to select the service components as late as possible (dynamic late binding) and at runtime. In this paper we propose a strategy of hybrid dynamic composition that combines the techniques of workflow driven composition and declarative composition. Composition techniques guided by a workflow define an abstract process model for the composed service and later each activity of the model is bound to a particular service. Declarative composition techniques, on the other hand, use customer-defined rules and constraints to determine if the resulting composite service meets customer expectations. In this case, service selection will depend on the fulfillment of each user-defined condition. Quality attributes of the resulting concrete service composition can be evaluated using a utility function using global and local optimization techniques [38].

On the other hand, the workflow can be seen as an arrangement of control-flow operators (e.g. sequence, conditional, parallel, etc.) that impact the utility function defined in a declarative technique. For example, the response time of a composed service that includes the sequential invocation of service components, results in the sum of the response time of the service components. If the service invocation happens in parallel, the response time of the composed service is the maximum value of the response time of the invoked services. When the invocation occurs within a loop, the composed service response time is the product of service component response time and the times that it is invoked. For the case of the conditional control-flow pattern, the response time of the composed service, in the worst case, is the maximum response time of the components to be chosen by the alternative condition. In [38], the influence of control-flow operators in the quality attributes of composed services are defined as aggregation functions.

A typical example of service composition is the Travel Planner. In this composite service, the search of places for entertainment (Attraction search) is performed in parallel with the reservation service and an airline reservation service of a hotel. Then, the distance between the place of entertainment and the hotel is calculated and depending on the outcome, a car rental or a bike rental service may be chosen. The abstract process of this composite service can be seen in Figure 9.

According to the algebra of service composition [39], the Travel Planner service is defined by $W_{tp} = \{(AttractionSearch \| (TicketBooking \odot Hotelbooking)) \odot DriveTimeCalculation \odot (BikeRental \diamond CarRental)\}$. In the notation, control-flow dependencies are denoted by symbols, for instance $\|$ defines parallel invocation, $\odot$ defines sequential invocation, and $\diamond$ defines a conditional invocation. Each task of an abstract process can be implemented using one of many existing services, which results in a composite service that should meet the user expectations. The number of possible ways for composing this Travel Planner service depends on the number of service candidates for each task. That is, if each task in the abstract process has two service candidates, then there are 2^6 different ways to implement the composite service. The objective of dynamic composition is to find the combination that gives the user the greatest benefits. In the remainder of this section we introduce SAW, which proposes a utility function that identifies the best combination according to multiple criteria.

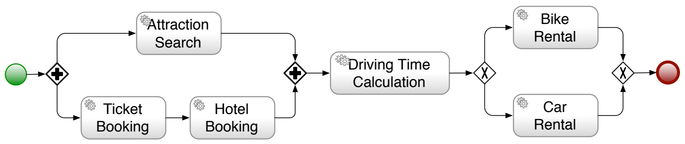


Fig. 9. Abstract BPMN model of the composite service travel planner

Simple Additive Weighting (SAW)

One way to assign a score to each possible service combination is by using the Simple Additive Weighting (SAW) utility function defined by Formula 13 [40]. The SAW technique consists of a process of assigning scores to each combination through two phases: scaling and aggregation or weighting. At the stage of scaling the values of the quality attributes of the candidate services are normalized between 0 and 1 by a comparison between the maximum and minimum values. Then, during the aggregation phase the following formulas are used to calculate the score of a composite service.

To define a particular composed service, we consider an abstract model $CS_{abstract} = S_1, S_2, ..., S_n$, restricted by customer defined constraints $QoS_{constraints} = C1, C2, ..., Cm$. The process that assigns a score for each combination of services $CS_x = s_1, s_2, ..., s_n$ of the abstract model works as follows: the quality attributes of each candidate service s is represented by the feature vector $q(s) = q_1, q_2, ..., q_k$, therefore $q_k(s)$ represents the k-quality attribute value for service s. Thus, $Q_{min}(j,k)$ and $Q_{max}(j,k)$ defined in Formula 11 represent the minimum and maximum value of the k quality attribute for the service candidates to implement a service S_j task.

$$Q_{min}(j,k) = \min_{\forall s \in S_j} q_k(s)$$
$$Q_{max}(j,k) = \max_{\forall s \in S_j} q_k(s) \tag{11}$$

Then $Q'_{min}(k)$ and $Q'_{max}(k)$ defined in Formula 12 calculate the minimum and maximum value of the k quality attribute for the abstract model using the aggregation functions F, defined in Table **??** for each operator of the model.

$$Q'_{min}(k) = F^{\ n}_{k\,j=1}(Q_{min}(j,k))$$
$$Q'_{max}(k) = F^{\ n}_{k\,j=1}(Q_{max}(j,k)) \tag{12}$$

Finally, the utility function assigns a score to the composite service CS_x, calculated by Formula 13 where the scores obtained for each quality attribute k, are weighted by the user preferences $W = (w_1, w_2, \ldots, w_k)$

$$U'(CS_x) = \sum_{k=1}^{r} \left(\frac{Q'_{max}(k) - q'_k(CS_x)}{Q'_{max}(k) - Q'_{min}(k)} \cdot w_k\right) \tag{13}$$

Simple Additive Weighting using Queue Model (SAW-Q)

Since dynamic composition aims to find the best service for each task at runtime, the time required to meet this objective has an important role. It is for this reason that techniques such as integer programming, dynamic programming, heuristic and distributed processing are mainly used in order to reduce the time service selection. However, most techniques ignore service demand while evaluating quality scores, which causes that the resulting composition may not be the most useful in practice. Hence, we modify the SAW technique to take into account the expected customer demand.

Unlike the previous model, in this model each service candidate s is defined by the vector of quality attributes $q(s,\lambda) = q_{rt}(s,\lambda), q_{av}(s,\lambda), q_{th}(s,\lambda)$. That is, by modeling REST services as a queuing system, we can calculate the quality attributes of a service according to the expected customer demand for the composite service (λ), using the formulas 8, 9 and 10 (Section 9.1). Hence, the SAW formulas presented before are modified introducing the user demand:

$$Q_{min}(j,k,\lambda) = \min_{\forall s \in S_j} q_k(s,\lambda)$$
$$Q_{max}(j,k,\lambda) = \max_{\forall s \in S_j} q_k(s,\lambda) \tag{14}$$

$$Q'_{min}(k,\lambda) = F^{\ n}_{k\,j=1}(Q_{min}(j,k,\lambda))$$
$$Q'_{max}(k,\lambda) = F^{\ n}_{k\,j=1}(Q_{max}(j,k,\lambda)) \tag{15}$$

The proposed utility function including clients' demand, SAW-Q, is defined by Formula 16:

$$U'(CS,\lambda) = \sum_{k=1}^{r} \frac{Q'_{max}(k,\lambda) - q'_k(CS,\lambda)}{Q'_{max}(k,\lambda) - Q'_{min}(k,\lambda)} \cdot w_k \tag{16}$$

The component services are selected according to the values obtained from SAW-Q for each abstract workflow. There are four factors that determine such value: the number of tasks the abstract model of the composite service; the number of candidates of the composite service services; quality constraints defined by the customer; and the control-flow patterns used in the abstract model. The control-flow patterns determine the execution path of a composite service. Depending on this, the component services can be invoked in sequence, in parallel, iteratively, or following alternative conditional execution paths. The quality attributes of a composite service are determined by the aggregate functions in Table **??**. The utility value ranks an implementation (of all possible) by calculating the expected behavior for each quality constraint. Therefore, since each control-flow pattern influences the utility function in various ways, the number and variety of control-flow patterns of an abstract model influence the choice of the best implementation and the best service for each task. As implementation options grow, the utility function becomes more important in the decision process to choose the best service for a task component.

9.4 Implementation and Evaluation

Implementation

Our principal hypothesis is that a dynamic composition approach based on quality restrictions that does not considers the user demand leads to implementations that can be erroneous in practice. In this section, we describe the implementation of a REST service composer based on SAW-Q, which is

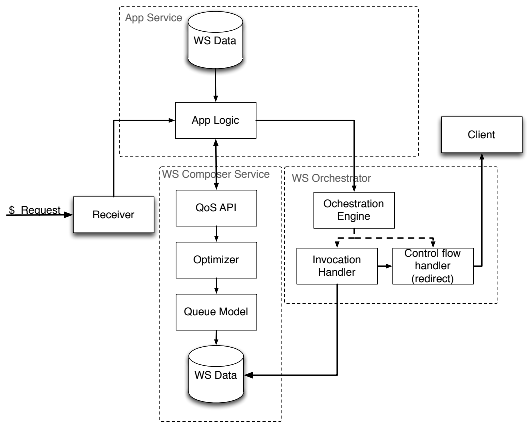


Fig. 10. Prototype architecture for dynamic composition using SAW-Q

a prototype that allows us to illustrate our results experimentally.

Figure 10 describes the prototype architecture. The module Receiver accepts the arriving requests ($\$\_Request$); it is responsible for recording service access and delivering the message to the next component. The App Logic module handles the request and determines its purpose in order to create a new instance of an existing composed service (e.g. `POST`) or to update the status of an existing one (e.g. `PUT`). When a new instance is requested, the WS Composer Service module determines the service components for the abstract process (CS_x). It accomplishes this task by invoking the Optimizer component which runs either SAW or SAW-Q in order to determine the utility values for the combination of service candidates. The Optimizer uses the Queue model library to perform equations 8, 9, and 10 (Section 9.1). The WS Composer Service defines the service invocation plan to follow. On the other hand, if the application intends to update the status of an existing composed service, then the request is derived to the WS Orchestrator module (Orchestrator Engine) that is responsible for executing the composition plan.

The Orchestrator Engine chooses at runtime the next service candidate with the highest score according the utility function determined by the WS Composer (Invocation Handles), and finally the Control-flow Handler executes the composition plan (i.e. prepares the HTTP request messages to be sent).

Control-flow patterns were implemented considering REST architectural constraints as proposed in [41]. Hence, we use redirections in order to keep application state on the client. The Control-flow Handler module extends the service response with header metadata corresponding to control-flow patterns as defined. This allows us to implement service compositions that are fully decentralized and stateless.

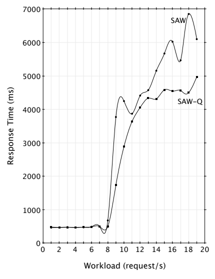
Services were implemented similarly to those of section 3.5, that is, as Python scripts (Django) running on an Apache Web server with PostgreSQL persistence. The experiment was performed online, the client run on an Intel PC with 4GB RAM, and 4 cores, the server (Ubuntu) runs on a virtual machine with 4 cores, 3GB RAM. Each service candidate was replicated up to 10 times in a LAN configuration.

Evaluation

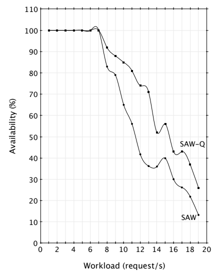
In order to validate the obtained results empirically, we implemented both techniques SAW and SAW-Q to choose at runtime the best service candidates using the previously described prototype. Services workload (requests/second) analysis was performed using the load-testing tool JMeter, for a time long enough to obtain stable results. The results obtained in this experiment confirms that SAW-Q's rankings fit better the experimental results.

Our experimental scenario comprehends a dataset of 6 tasks corresponding to the abstract BPMN model shown in Figure 9. For each task in the business process we implemented 2 alternative services, that is, we implemented 12 services which were characterized with a random arrival rate between 4 to 10 requests per second, with random replicas (between 1 to 5) and a random capacity for each replica of 10 to 40 services in the waiting queue.

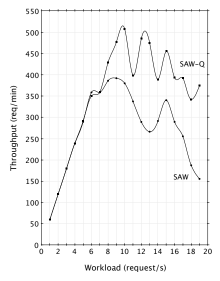
The response time is a quality attribute that negatively affects the quality of service when it grows. In this experiment the average response time for a service following SAW is higher than the one obtained using SAW-Q when the user demand is taken into account. Figure 13 shows the impact of changes in the demand from 1 to 19 requests per second on SAW and SAW-Q. For each value of the demand, we considered the average response time of the composed service with the highest score for SAW and SAW-Q (Figure 11(a)). The differences between both techniques can be appreciated when the demand scales up to 8 requests per second, from then, the services with the highest score according to



(a) Response Time according SAW and SAW-Q



(b) Availability according SAW and SAW-Q



(c) Throughput according SAW and SAW-Q

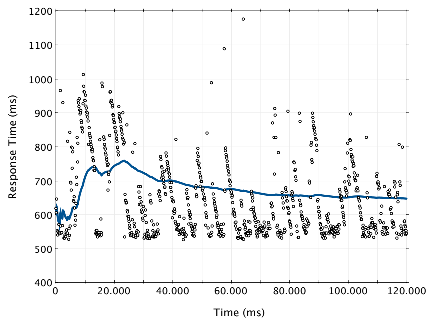
Fig. 11. SAW/SAW-Q comparison.

SAW obtains a higher response time when compared to the top score services obtained by SAW-Q, hence quality of the services selected by SAW is worst than those services selected by SAW-Q.Figure 11(b) and Figure 11(c) shows the impact of SAW and SAW-Q selections on the availability and throughput of the services with highest score. Again the difference between both techniques appears since the workload is 9 requests per second.

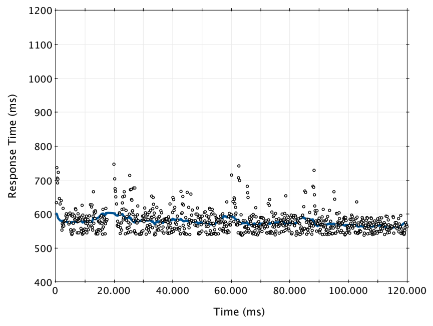
In Figure 12 we go further analyzing the situation when the demand reaches 8 requests per second. ($\lambda = 8req/s$). The dots represent the obtained results and the continuous line, the average. Note that in the composition proposed by SAW the standard deviation is bigger than the composition using SAW-Q.

The availability of the services composed using SAW-Q, on average, is greater than the compositions obtained using SAW. Figure 13 shows the availability of both services when the demand is $8req/s$. The dots are the obtained results an the continuous line shows, the measures average.

Composite service throughput using SAW-Q is slightly higher than the one using SAW which means that the SAW-Q composite service has the capacity to serve more requests per minute than the SAW composite service. Figure 14 shows the results of

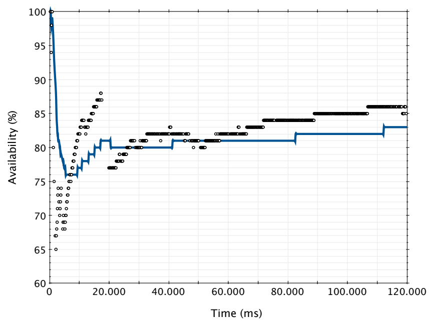


(a) Response Time SAW

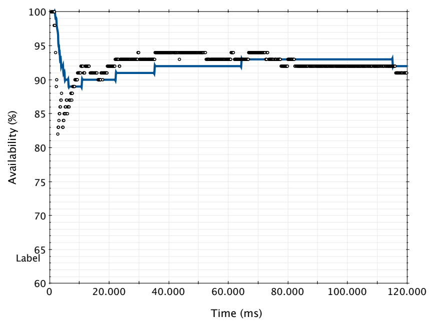


(b) Response Time SAW-Q

Fig. 12. SAW/SAW-Q comparison: Composed service response time.



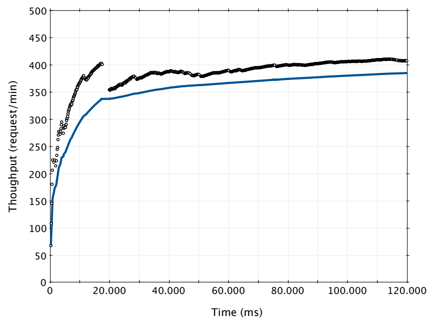
(a) Availability SAW



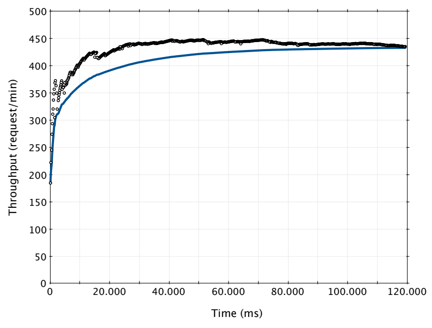
(b) Availability SAW-Q

Fig. 13. Availability comparison between both techniques of composition.

both services when the demand is 8 requests per second. Again, the dots are the obtained results and the continuous line shows the average.



(a) Throughput SAW



(b) Throughput SAW-Q

Fig. 14. Throughput comparison between SAW and SAW-Q.

10 Conclusions

10.1 Regarding Decentralized, stateless, complex service behavior in REST

In Section 7 a proposal for the design and implementation of complex composed service behavior is presented. This proposal places emphasis on REST architectural constraints, having as goal to achieve scalability and statelessness for the composed service behavior. With these considerations in mind, a set of well-known control-flow patterns that are used to implement simple and complex behavior in traditional Web services were recreated.

The main conclusion from the experience is that a decentralized, stateless implementation of a composed service satisfies REST architectural constraints and provides significant improvements regarding throughput and availability, which are non-functional goals of REST.

Second, when following REST architectural constraints and a decentralized, stateless approach where the client (User Agent) shares the interaction responsibility with the server, control-flow patterns design in REST differ from those in SOA where state (information interaction) is kept in a centralized component (the orchestrator).

Third, one of the extensibility mechanisms of HTTP, namely status codes, was used to implement the presented approach. Other alternatives could be used, such as link headers, or ad-hoc media types (e.g. a specialized JSON document). However, the precedence for link processing indicates that such messages must be processed after the representation is fully received and processed by the client and after users have performed the actions they required (e.g. click on buttons, or run javascript controls), which introduces not only delays but also security risks.

Finally, the presented approach requires that the client knows in advance how to process the messages, so that, it shall be a process-oriented User Agent. In addition, this design choice also includes the typical vulnerabilities of nowadays User Agents. Additional measures such as digital signatures must be included in order to guarantee a safe interaction between services (mediated by the User Agent).

10.2 Regarding hybrid (static and dynamic) service composition in REST

In Section 8 a technique that exploits hypermedia-centric REST service descriptions (defined at design time) is used at runtime to determine the feasibility of service composition and actually enacting a composition with an authentication service based on OAuth. Again a decentralized approach, a choreography, was followed.

The main conclusion from this approach is that hypermedia-centric REST service descriptions can actually serve as a basis for a well-behaved User Agent traverses complex paths on the Web of services. However, since such descriptions are created at design-time, dynamic changes on the service provision (e.g. changes on the service interface) could not be reflected on the descriptions. Descriptions that are out of sync with the service implementations may impede the User Agent to continue its work, although a good service description can provide information to the client developer so that the changes can be easily noticed.

Second, a QoS domain (security) is addressed in this section not only because an important authorization technique (OAuth) is an example of a highly scalable and well-known choreography, but also because QoS-aware service composition is a field that have been extensible studied in order to support automatic and dynamic service composition. QoS attributes, particularly security are playing a major role in the current Web due to the massive scale, performance, availability and evolvability requirements that pervade modern Web applications. However, most techniques reduce QoS complexity to single values (e.g. booleans or numbers) when in practice some, such as security, shall be represented as a combination of diverse algorithms and protocols that could be available or not, or even worst shall be tried to follow in order to discover the feasibility of choosing a service as a component for a composed service.

10.3 Regarding dynamic service composition in REST

In Section 9, SAW-Q (an extension of SAW) is proposed as a novel dynamic composition technique that follows the principles of the REST style. SAW-Q models quality attributes as a function of the actual service demand instead of the traditional constant values.

The main conclusion is obtained when comparing both techniques SAW-Q is much more accurate than SAW when compared to real implementations, positively improving the quality of dynamic service compositions. Quality attributes of a REST service such as, availability, response time and throughput can be modeled with better accuracy using queuing theory since it considers implementation details that are particularly relevant for service architecture, such as the processing time of the application, the number of times the service is replicated, the maximum number of clients that can be handled by each service replica at a given time and the request that remain in the waiting queue.

Second, choosing the right candidate service in dynamic composition is a critical task. Different ways of measuring the quality of a service can lead to errors. The dynamic composition technique proposed, SAW-Q, considers the attributes of service quality as a function of the demand for requests thus obtained composed services that behave better than those determined by SAW.

Finally, the present approach contributes a deeper analysis on the scalability related attributes. Considering the request demand or workload when modeling services and composed services is critical particularly to certain quality attributes such as throughput, response time, and availability but also fault tolerance and even price, which are not considered in our study.

References

[1] T. Erl, *Soa: principles of service design*. Prentice Hall Upper Saddle River, 2008, vol. 1.

[2] ——, *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.

[3] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, University of California, Irvine, 2000. [Online]. Available: http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm

[4] T. Erl, B. Carlyle, C. Pautasso, and R. Balasubramanian, *SOA with REST: Principles, Patterns &Constraints for Building Enterprise Solutions with REST*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2012.

[5] J. Mendling and M. Hafner, “From ws-cdl choreography to bpel process orchestration,” *Journal of Enterprise Information Management*, vol. 21, no. 5, pp. 525–542, 2008. [Online]. Available: http://dx.doi.org/10.1108/17410390810904274"

[6] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland *et al.*, “Web services business process execution language version 2.0,” *OASIS standard*, vol. 11, p. 11, 2007.

[7] C. Pautasso, “Composing restful services with jopera,” in *Software Composition*, ser. Lecture Notes in Computer Science, A. Bergel and J. Fabry, Eds. Springer Berlin Heidelberg, 2009, vol. 5634, pp. 142–159.

[8] R. Alarcón, E. Wilde, and J. Bellido, “Hypermedia-driven restful service composition,” in *6th Workshop on Engineering Service-Oriented Applications (WESOA 2010)*. Springer, 2010.

[9] R. Alarcon and E. Wilde, “Linking data from restful services,” in *Third Workshop on Linked Data on the Web, Raleigh, North Carolina (April 2010)*, 2010.

[10] R. Krummenacher, B. Norton, and A. Marte, “Towards linked open services and processes,” in *Future Internet - FIS 2010*, ser. Lecture Notes in Computer Science, A. Berre, A. GÃşmez-PÃľrez, K. Tutschku, and D. Fensel, Eds. Springer Berlin Heidelberg, 2010, vol. 6369, pp. 68–77.

[11] S. Stadtmüller and A. Harth, “Towards data-driven programming for restful linked data,” in *Workshop on Programming the Semantic Web (ISWCâĂŹ12)*, 2012.

[12] R. Verborgh, T. Steiner, D. Deursen, R. Van de Walle, and J. Valles, “Efficient runtime service discovery and consumption with hyperlinked restdesc,” in *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*, oct. 2011, pp. 373 –379.

[13] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto, “Web services choreography description language version 1.0,” *W3C candidate recommendation*, vol. 9, 2005.

[14] G. Decker, “Process choreographies in service-oriented environments,” Ph.D. dissertation, Masters thesis, Hasso Plattner Institute at University of Potsdam, 2006.

[15] M. zur Muehlen, J. V. Nickerson, and K. D. Swenson, “Developing web services choreography standardsâĂŤthe case of {REST} vs. {SOAP},” *Decision Support Systems*, vol. 40, no. 1, pp. 9 – 29, 2005, web services and process management. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167923604000612

[16] C. Pautasso, “Restful web service composition with bpel for rest,” *Data Knowl. Eng.*, vol. 68, no. 9, pp. 851–866, September 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1550965.1551240

[17] O. Nierstrasz and T. D. Meijler, “Requirements for a composition language,” *Lecture Notes in Computer Science*, vol. 924, pp. 147–161, 1995.

[18] C. Pautasso and G. Alonso, “The {JOpera} visual composition language,” *Journal of Visual Languages and Computing*, vol. 16, no. 1âĂŞ2, pp. 119 – 152, 2005, 2003 {IEEE} Symposium on Human Centric Computing Languages and Environments. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1045926X04000400

[19] F. Rosenberg, F. Curbera, M. Duftler, and R. Khalaf, “Composing restful services and collaborative workflows: A lightweight approach,” *Internet Computing, IEEE*, vol. 12, no. 5, pp. 24–31, Sept 2008.

[20] G. Decker, A. Lüders, H. Overdick, K. Schlichting, and M. Weske, “Restful petri net execution,” in *WS-FM*, 2008, pp. 73–87.

[21] X. Xu, L. Zhu, Y. Liu, and M. Staples, “Resource-oriented architecture for business processes,” in *Software Engineering Conference, 2008. APSEC ’08. 15th Asia-Pacific*, Dec 2008, pp. 395–402.

[22] H. Zhao and P. Doshi, “Towards automated restful web service composition,” in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, July 2009, pp. 189–196.

[23] D. Menasce, “Qos issues in web services,” *Internet Computing, IEEE*, vol. 6, no. 6, pp. 72 – 75, nov/dec 2002.

[24] R. Kübert, G. Katsaros, and T. Wang, “A restful implementation of the ws-agreement specification,” in *Proceedings of the Second International Workshop on RESTful Design*, ser. WS-REST ’11. New York, NY, USA: ACM, 2011, pp. 67–72. [Online]. Available: http://doi.acm.org/10.1145/1967428.1967444

[25] S. Graf, V. Zholudev, L. Lewandowski, and M. Waldvogel, “Hecate, managing authorization with restful xml,” in *Proceedings of the Second International Workshop on RESTful Design*, ser. WS-REST ’11. New York, NY, USA: ACM, 2011, pp. 51–58. [Online]. Available: http://doi.acm.org/10.1145/1967428.1967442

[26] J. P. Field, S. G. Graham, and T. Maguire, “A framework for obligation fulfillment in rest services,” in *Proceedings of the Second International Workshop on RESTful Design*, ser. WS-REST ’11. New York, NY, USA: ACM, 2011, pp. 59–66. [Online]. Available: http://doi.acm.org/10.1145/1967428.1967443

[27] D. Allam, “A unified formal model for service oriented architecture to enforce security contracts,” in *Proceedings of the 11th Annual International Conference on Aspect-oriented Software Development Companion*, ser. AOSD Companion ’12. New York, NY, USA: ACM, 2012, pp. 9–10. [Online]. Available: http://doi.acm.org/10.1145/2162110.2162120

[28] J. Hongbin, Z. Fengyu, and X. Tao, “Security policy configuration analysis for web services on heterogeneous platforms,” *Physics Procedia*, vol. 24, Part B, no. 0, pp. 1422 – 1430, 2012, international Conference on Applied Physics and Industrial Engineering 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1875389212002544

[29] F. Rosenberg, F. Curbera, M. J. Duftler, and R. Khalaf, “Composing RESTful services and collaborative workflows: A lightweight approach,” *IEEE Internet Computing*, vol. 12, no. 5, pp. 24–31, 2008. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MIC.2008.98

[30] D. DâĂŹMello, V. Ananthanarayana, and S. Salian, “A review of dynamic web service composition techniques,” in *Advanced Computing*, ser. Communications in Computer and Information Science, N. Meghanathan, B. Kaushik, and D. Nagamalai, Eds. Springer Berlin Heidelberg, 2011, vol. 133, pp. 85–97.

[31] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, “Workflow patterns,” *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.

[32] N. Russell, Arthur, W. M. P. van der Aalst, and N. Mulyar, “Workflow control-flow patterns a revised view,” BPMcenter.org, New York, NY, USA, Tech. Rep. BPM-06-22, 2006.

[33] G. D. Ivan Zuzak, Ivan Budiselic, “A finite-state machine approach for modeling and analyzing restful systems,” *Web Engineering*, vol. 10, no. 4, pp. 353–390, 2011.

[34] M. Maleshkova, C. Pedrinaci, J. Domingue, G. Alvaro, and I. Martinez, “Using semantics for automating the authentication of web apis,” in *The Semantic Web - ISWC 2010*, ser. Lecture Notes in Computer Science, P. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Pan, I. Horrocks, and B. Glimm, Eds. Springer Berlin / Heidelberg, 2010, vol. 6496, pp. 534–549.

[35] E. Hammer-Lahav, “The oauth 1.0 protocol,” 2010.

[36] Q. Tao, H. you Chang, C. qin Gu, and Y. Yi, “A novel prediction approach for trustworthy qos of web services,” *Expert Systems with Applications*, vol. 39, no. 3, pp. 3676 – 3681, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417411013698

[37] H. A. Taha, *Operations research: an introduction*. Pearson-/Prentice Hall, 2007.

[38] M. Alrifai, T. Risse, and W. Nejdl, “A hybrid approach for efficient web service composition with end-to-end qoS constraints,” *TWEB*, vol. 6, no. 2, p. 7, 2012. [Online]. Available: http://doi.acm.org/10.1145/2180861.2180864

[39] R. Hamadi and B. Benatallah, “A petri net-based model for web service composition,” in *Fourteenth Australasian Database Conference (ADC2003)*, ser. CRPIT, K.-D. Schewe and X. Zhou, Eds., vol. 17. Adelaide, Australia: ACS, 2003, pp. 191–200. [Online]. Available: "http://crpit.com/confpapers/CRPITV17Hamadi.pdf"

[40] M. Zeleny and J. L. Cochrane, *Multiple criteria decision making*. McGraw-Hill New York, 1982, vol. 25.

[41] J. Bellido, R. Alarcón, and C. Pautasso, “Control-flow patterns for decentralized restful service composition,” *ACM Trans. Web*, vol. 8, no. 1, pp. 5:1–5:30, December 2013. [Online]. Available: http://doi.acm.org/10.1145/2535911

Graph Laplacian for Spectral Clustering and Seeded Image Segmentation

Wallace Casaca
Institute of Mathematics and Computer Science (ICMC)
University of São Paulo (USP)
13566-590, São Carlos, SP, Brazil
https://sites.google.com/site/wallacecoc
wallace@icmc.usp.br

***Abstract*—Image segmentation is an indispensable tool to enhance the ability of computer systems to perform elementary cognitive tasks such as *detection*, *recognition* and *tracking*. In particular, interactive algorithms have gained much attention lately, specially due to their good performance in segmenting complex images and easy utilization. However, most interactive segmentation methods rely on sophisticated mathematical tools whose effectiveness strongly depends on the kind of image to be processed. In fact, sharp adherence to the contours of image segments, uniqueness of solution, high computational burden, and extensive user intervention are some of the weaknesses of most existing techniques. In this work we proposed two novel interactive image segmentation techniques that sort out the issues discussed above. The proposed methods rely on Laplace operators, spectral graph theory, and optimization approaches towards enabling highly accurate segmentation tools which demand a reduced amount of user interaction while still being mathematically simple and computationally efficient. The good performance of our segmentation algorithms is attested by a comprehensive set of comparisons against representative state-of-the-art methods. Indeed, qualitative and quantitative results obtained from well-known image benchmarks show that our methodologies outperform others. As additional contribution, we have also proposed two new algorithms for image inpainting and photo colorization, both of which rely on the accuracy of our segmentation apparatus.**

Keywords*—*Image segmentation, graph laplacian, spectral graph theory, inpainting, colorization, computer vision applications

I. Introduction

Image segmentation is the key task for an enormous quantity of computer vision problems. A typical procedure in image segmentation is to interpret an image as a graph, which enables the use of powerful mathematical tools such as Laplace operators and spectral graph theory. Moreover, the flexibility introduced by a graph representation as to pixel connectivity and edge weighting greatly increases the capability of segmentation algorithms to distinguish patterns, structures, and shapes. However, outperforming human skills in terms of recognition is a difficult task. Therefore, *semi-supervised image segmentation methods* have become a trend by combining the human ability for object/background detection with the solid mathematical foundation of graph theory [1].

In this scenario, the use of interact mechanisms to properly settle Laplace operators on image graph representations have proven to be an effective alternative [2], [3]. Those user-assisted mechanisms typically define the Laplace operators in a similarity/affinity graph which encodes image information such as colors, textures and gradients. Moreover, it involves a cost function defined on the graph [4], [5], [6], [3], or solving a spectral-cut problem [7], [8], [9], [10]. However, as pointed out in [11], [12], [13], existing partitioning techniques are circumstantially prone to fail in many pragmatic situations. For instance, common drawbacks not tackled by state-of-the-art algorithms are:

1) The resulting segmentation generally exhibits low adherence on the contours of the image regions, failing to capture fine details or, in many cases, producing a low quality segmentation output.
2) Make use of sophisticated optimization tools to be effective, impacting negatively on the computational cost, implementation and portability of the code.
3) Demand extra computational effort to the user, specially for processing high resolution images, such those obtained nowadays by mobile devices.
4) They are highly sensitive to the adjustment of the edge weights in the graph.

A. Contributions

In this thesis we proposed two novel user-assisted image segmentation techniques that address the issues discussed above. The proposed algorithms rely on Laplace operators, spectral graph theory, and optimization tools towards reaching highly fitting on object boundaries which demand a reduced amount of user involvement while still being mathematically easy to solve and computationally efficient.

While most of our research has been focused on the particular problem of image segmentation, we develop as side results new methodologies for the problem of *image inpainting* [14], [15] and *photo colorization* [16], [17], both of which derived from the proposed segmentation methodologies. Figure 1 presents some illustrative examples while the list below provides the main publications originated during the development of this thesis:

Contributions in Computer Vision: [10], [18], [17], [19], [2], [15], [3], [20], [21].

Contributions in Graphics and Visualization: [22], [23], [24], [25].

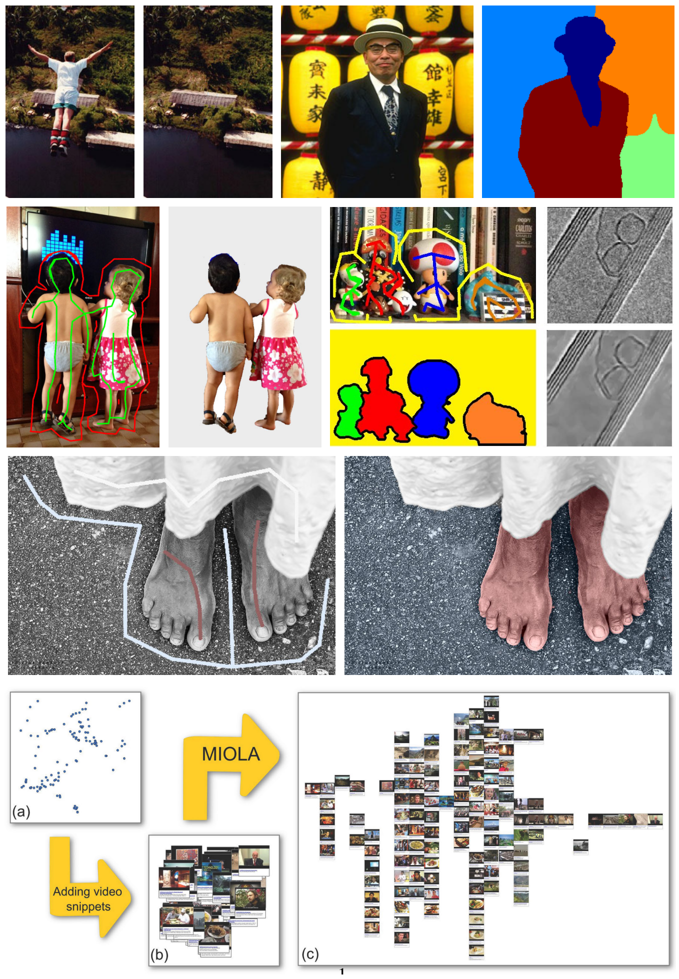


Fig. 1. A few results obtained during the PhD research.

B. Awards

The following papers have been awarded as "best papers" or received "honorable mention" in the renowned SIBGRAPI and ICCP conferences. According to Google Scholar, SIBGRAPI (Conference on Graphics, Patterns and Images) is the most relevant Latin American conference in the fields of *Graphics*, *Visualization* and *Computer Vision*. ICCP (IEEE International Conference on Computational Photography) is a prominent conference in the field of *Digital Photography*.

- **Best paper award** for the paper "Spectral Segmentation using Cartoon-Texture Decomposition and Inner Product-based Metric", 24th SIBGRAPI, Maceio, Brazil, 2011.
- **Best paper award** for the paper "Mixed Integer Optimization for Layout Arrangement", 26th SIBGRAPI, IEEE Computer Society, Arequipa, Peru, 2013[1].
- **Honorable Mention** for the poster presentation "Image Colorization based on Multidimensional Projection", 5th IEEE ICCP, Harvard, Cambridge, United States, 2014.

C. Scientific Dissemination to the General Public

The research conducted during the thesis had a good impact in terms of diffusion to the general public (see Figure 2 for an illustration). In fact, part of our research has been advertised on TV news channels, news papers, and on the internet, as listed below (in portuguese):

1) **News published on *www.usp.br*** (2013): "Pesquisadores do ICMC recebem prêmio internacional em computação gráfica" (www.icmc.usp.br/e/53ddb).
2) **Report published on *www.usp.br*** (2014): "ICMC desenvolve ferramenta inovadora para segmentação de imagens" (www.icmc.usp.br/e/f37b3).
3) **Report published on *DCI (Newspaper)*** (2014): "Pesquisadores desenvolvem nova ferramenta para imagem" (www.icmc.usp.br/e/0ebf1).
4) **TV report broadcasted on *Rede Globo*** (2014): "Programa criado pela USP São Carlos remove pessoas de foto de forma fácil" (www.icmc.usp.br/e/49c86).
5) **TV report broadcasted on *TV Educativa de São Carlos*** (2014): "USP cria ferramenta que facilita a vida de quem trabalha com imagens" (www.icmc.usp.br/e/9daca).

[1]See the video of our tool at https://www.youtube.com/watch?v=zGgIYX7oSqI

Universidade de São Paulo
Nunca foi tão fácil alterar uma imagem
SÃO CARLOS E ARARAQUARA
Programa criado pela USP São Carlos remove pessoas de foto de forma fácil
Ferramenta contribui para aprimorar personalização artística de imagem. Segundo os desenvolvedores, até uma criança consegue manipulá-lo.

Fig. 2. Dissemination of the technologies originated to the wider public.

A summary of the main results obtained during the development of this thesis is presented in the following sections.

II. Spectral Image Segmentation

Spectral graph theory [1] has been the basic tool for the so-called spectral cut methodology [7], [8], [26], [27], which exploits the eigenstructure of an image affinity graph so as to perform clustering. In fact, spectral graph theory enables great flexibility in the segmentation process, as different choices can be made towards defining the similarity graph connectivity as well as the assignment of weights to the edges of the graph. Such a flexibility has leveraged a multitude of techniques, making spectral cuts an attractive image segmentation tool.

Among the vast amount of techniques inspired in spectral cuts, three approaches have gain a lot of attention in recent years, being widely used as source of segmentations in many practical applications:

1) *Spectral and Normalized Cuts-based methods* [7], [28], [29], [30], [31], [8], [9], [32];
2) *Multiscale Segmentation-based methods* [33], [34], [35], [36], [37];
3) *Random Walker-based methods* [38], [5], [39], [40].

Despite their effectiveness and powerfulness, methods inspired on spectral cuts present some weaknesses that must be observed when performing segmentation. For example, the accuracy in detecting the boundaries between image regions

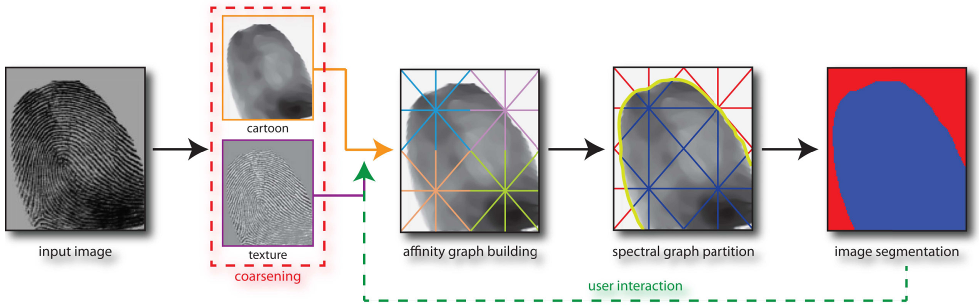


Fig. 3. Pipeline of the proposed image segmentation framework.

is highly dependent on the weights assigned to the edges of the graph. Although automatic schemes have been proposed to accurately compute those weights [7], [34], [41], [35], [42], it is well-known that user intervention is essential in many cases to correctly define object boundaries [39]. Therefore, incorporating user knowledge into the segmentation process is of paramount importance since the identification of boundary information is subject to human judgment in many practical situations. Another important issue in the context of spectral cuts is the computational cost, as computing the eigenstructure of a graph is a very time consuming task, hampering the direct use of spectral segmentation in high resolution images [43].

A. *Spectral Segmentation via Cartoon-Texture Decomposition and Inner Product-based Metric*

In this section we introduce a new methodology for image segmentation that relies on spectral cuts but addresses the issues raised above. We show that the proposed approach outperforms classical spectral segmentation techniques in aspects such as accuracy and robustness on the well-known image dataset from UC-Berkley [44]. Figure 4 shows an example of the proposed framework. We can summarize the novelties introduced by our methodology as:

1) An image segmentation technique that combines cartoon-texture decomposition and spectral cuts [2];
2) A novel method to compute and assign weights to the edges of the similarity graph using the cartoon component extracted from the image;
3) A new strategy to modify the weights of the graph according to user interaction, taking into account the texture component of the image.

B. *Pipeline Overview*

The proposed approach, first reported in [10], [2], comprises five main steps, as illustrated in Figure 3. The first step, *Cartoon-Texture Decomposition*, separates the target image $\mathcal{I}$ into two images, $\mathcal{C}$ and $\mathcal{T}$, where $\mathcal{C}$ and $\mathcal{T}$ hold the cartoon and texture information contained in $\mathcal{I}$. In the second step, an *Image Coarsening* is applied to $\mathcal{C}$ and $\mathcal{T}$ so as to build smaller affinity graphs in the third step of the pipeline, namely *Affinity Graph Construction*. Besides speeding up the spectral decomposition, the reduced number of edges also lessen the computational burden during the weight assignment stage, which allow us to handle large images. Weights are derived from an inner product-based metric defined on the coarse cartoon image. The spectral decomposition is carried out in the *Spectral Partition* step, being the result mapped back to the original image through a coarse-to-fine interpolation procedure. The user can change the partition by stroking the resulting segmentation. This step is performed by combining the coarse texture component with a recent technique of harmonic analysis [45], [46] in order to incorporate the high-level oscillatory information into the spectral cut process. Details about each step of the pipeline are provided below.

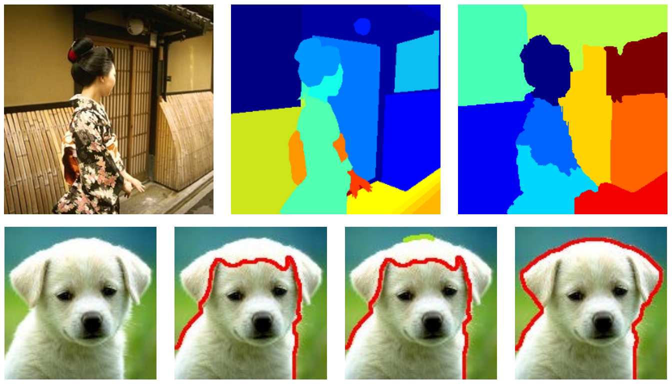
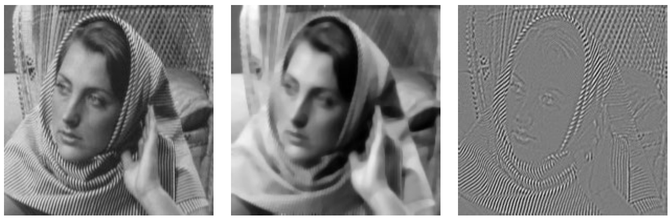


Fig. 4. First row: from the left to right, the ground truth image and the result obtained with our method without user intervention. Second row: illustrates the user interaction (green scribes) to improve the segmentation.

1) Cartoon-Texture Image Decomposition: CTD splits the input image $\mathcal{I}$ into two disjoint images, $\mathcal{C}$ and $\mathcal{T}$. The cartoon component $\mathcal{C}$ holds the geometric structures, isotopes and smooth-pieces of $\mathcal{I}$ while the texture component contains textures, oscillating patterns, fine details and noise. This decomposition schemes satisfies the important relationship $\mathcal{I} = \mathcal{C} + \mathcal{T}$ (see [47], [18] and the underlying mathematical theory proposed in [48]). Similar to [47], where a functional minimization problem has been formulated and solved through



(a) Input Image (b) Cartoon (c) Texture

Fig. 5. Image decomposition into a cartoon and texture component.

a system of partial differential equations, both cartoon $\mathcal{C}$ and texture $\mathcal{T}$ components are computed by solving the following system of equations:

$$\begin{cases} \mathcal{C} = \mathcal{I} - \partial_x g_1 - \partial_y g_2 + \dfrac{1}{2\lambda}\,\mathrm{div}\left(\dfrac{\nabla\mathcal{C}}{|\nabla\mathcal{C}|}\right) \\ \mu\dfrac{g_1}{\sqrt{g_1^2+g_2^2}} = 2\lambda\left[\dfrac{\partial}{\partial x}(\mathcal{C}-\mathcal{I}) + \partial^2_{xx}g_1 + \partial^2_{xy}g_2\right] \\ \mu\dfrac{g_2}{\sqrt{g_1^2+g_2^2}} = 2\lambda\left[\dfrac{\partial}{\partial y}(\mathcal{C}-\mathcal{I}) + \partial^2_{xy}g_1 + \partial^2_{yy}g_2\right] \end{cases}, \qquad (1)$$

with initial conditions for $\mathcal{C}$, g_1, and g_2 given by

$$\begin{cases} \dfrac{\nabla\mathcal{C}}{|\nabla\mathcal{C}|}\cdot(n_x,n_y) = 0 \\ (\mathcal{I}-\mathcal{C}-\partial_x g_1 - \partial_y g_2)\cdot nx = 0 \\ (\mathcal{I}-\mathcal{C}-\partial_x g_1 - \partial_y g_2)\cdot ny = 0 \end{cases}. \qquad (2)$$

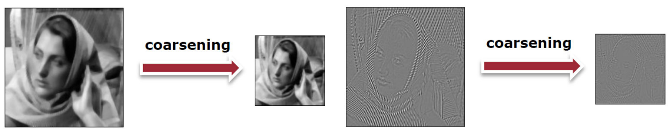
Mathematically, the cartoon component $\mathcal{C}$ is a bounded variation function, $\overrightarrow{g} = (g_1, g_2) \in L^2(\mathbb{R}^2)$ where the texture component $\mathcal{T} = \mathrm{div}(\overrightarrow{g})$, and the constants $\lambda, \mu > 0$ are tuning parameters. Equations (1) are usually discretized by a semi-implicit finite difference schemes and solved using an iterative algorithm based on fixed point iteration (for more details about numerical aspects, see [47], [49]). Figure 5 shows the result of the CTD scheme applied to a digital image.

In our methodology, both $\mathcal{C}$ and $\mathcal{T}$ are used to compute the weights assigned to the edges of the affinity graph. Since $\mathcal{C}$ is a texture-free denoised image, edge and shape detectors work well when applied to $\mathcal{C}$ as pointed out in [47]. This fact is exploited to define the weights, as we detail later. Information contained in $\mathcal{T}$ is handled only at the end of pipeline, during user interaction stage.

2) Image Coarsening: In order to reduce the size of the affinity graph towards alleviating the computational burden during the spectral decomposition, we perform a fine-to-coarse transformation on $\mathcal{C}$ (resp. $\mathcal{T}$), resulting in a coarse scale $\tilde{\mathcal{C}}$ (resp. $\tilde{\mathcal{T}}$) of $\mathcal{C}$ (resp. $\mathcal{T}$). Such a transformation is accomplished using the bicubic interpolation method described in [50], which minimizes the blurring effect while still preserving gradients in the coarse image (see Figure 6 for an illustration). Other downsampling techniques such as [51] can be alternatively used to convey essential image information among scales. Our experiments showed that coarsening the image to one-fourth of its original resolution is a good trade-off between computational time and accuracy, speeding up the processing up to 6 times, as outlined in the comparison section.

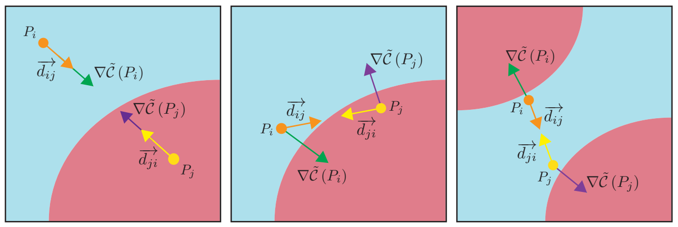
3) Building the Affinity Graph: The affinity graph G is built by associating each pixel from $\tilde{\mathcal{C}}$ to a node of the graph, connecting the nodes according to the distance r between corresponding pixels, in mathematical words,

$$\|P_i - P_j\|_\infty < r\,. \qquad (3)$$



(a) Cartoon image coarsening (b) Texture image coarsening

Fig. 6. Fine-to-coarse step illustration obtained from [50].


P_i
$\overrightarrow{d_{ij}}$
$\nabla\tilde{\mathcal{C}}(P_i)$
$\nabla\tilde{\mathcal{C}}(P_j)$
$\overrightarrow{d_{ji}}$
P_j

(a) Same direction (b) Oblique direction (c) Opposite direction

Fig. 7. Geometric interpretation of the inner product-based metric. Maximum weights occur when the gradient and the direction defined from the graph edge point to the same direction (a). Moderate weight is highlighted in (b) and the third case, where opposite directions (c) produce minimum weights (zero).

The weight assigned to each edge of G is derived from the proposed inner product-based metric. Our metric considers the variation of the image in the directions defined by the edges of the graph. More specifically, the weight w_{ij} associated to the edge e_{ij} is defined as:

$$w_{ij} = \frac{1}{1+\eta h_{ij}^2}, \quad h_{ij} = \max\left\{\frac{\partial\tilde{\mathcal{C}}(P_i)}{\partial\overrightarrow{d_{ij}}}, \frac{\partial\tilde{\mathcal{C}}(P_j)}{\partial\overrightarrow{d_{ji}}}, 0\right\}, \qquad (4)$$

$$\frac{\partial\tilde{\mathcal{C}}(x)}{\partial\overrightarrow{d_{ij}}} = \langle\nabla\tilde{\mathcal{C}}(x), \overrightarrow{d_{ij}}\rangle, \quad \text{with} \quad \overrightarrow{d_{ij}} = \frac{\overrightarrow{P_iP_j}}{|\overrightarrow{P_iP_j}|}. \qquad (5)$$

The left most term in Equation (5) is the directional derivative of $\tilde{\mathcal{C}}$ in the direction $\overrightarrow{d_{ij}}$, which is defined from the graph G and $\eta > 0$ is a tuning constant. Therefore, image properties as well as the adjacency structure of the affinity graph is taken into account when assigning weights to the edges of G. Furthermore, our formulation accounts for the intensity and geometric information to define the weights through the inner-product in the edge direction. Figure 7 provides a geometric interpretation of the proposed metric.

The effective weights w_{ij} are chosen from Eq. (4) rather then using the exponential measure usually employed by other authors [7], [30], [36]. The scheme proposed in Eq. (4) does not push values to zero as fast as the exponential function, which allows for considering the influence of a larger number of edges when carrying out the spectral decomposition. Eq. (4) is indeed derived from the Malik-Perona diffusivity term [52], [53], which was originally used for establishing the notion of anisotropy in partial differential equations. Moreover, the inner product-based metric (4) holds that $w_{ij} = w_{ji}$, which ensures symmetry for the graph Laplacian matrix $\mathbf{L}$. This fact is of paramount importance to guarantee that the eigenstructure of $\mathbf{L}$ is made up of only real numbers.

4) Spectral Cutting and Coarse-to-Fine: Given the affinity graph G built from $\tilde{\mathcal{C}}$ and the number of partitions initially defined by the user, we carry out the spectral decomposition using the same methodology proposed in [7]. More specifically, we first decompose the graph Laplacian matrix as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ and $\mathbf{W}$ contain the diagonal and off-diagonal elements of $\mathbf{L}$. Then, the Fiedler vector $\mathbf{f}$ is obtained by solving the generalized eigenvalue problem

$$(\mathbf{D} - \mathbf{W})\mathbf{x} = \lambda\mathbf{D}\mathbf{x}, \qquad (6)$$

getting $\mathbf{f}$ as the eigenvector associated to the smallest non-zero eigenvalue. The Fiedler vector splits $\tilde{\mathcal{C}}$ into two subsets,

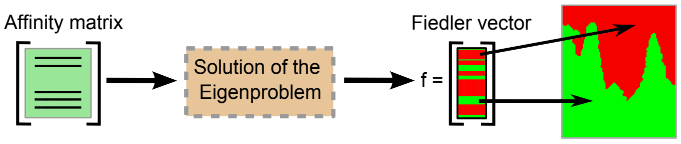


Fig. 8. Spectral cut pipeline to partition the image from the zero-set of the Fiedler vector.

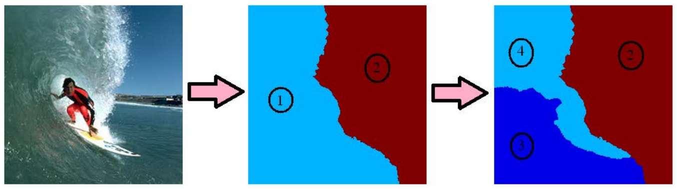


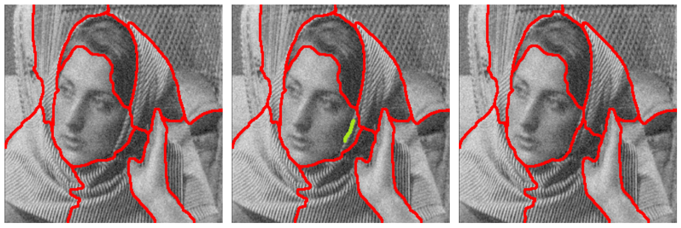
Fig. 9. Hierarchical segmentation by recursively computing the spectral decomposition for multiple parts of the image.

one containing the pixels corresponding to nodes of the graph where the entries of $\mathbf{f}$ are positive and other containing the pixels with negative values of $\mathbf{f}$. Therefore, the zero-set of $\mathbf{f}$ is a curve that separates the regions with different signs. Figure 8 portrays the spectral cut procedure. The partitioning created in $\tilde{\mathcal{C}}$ is brought back to $\mathcal{C}$ using bicubic interpolation from $\mathbf{f}$.

Multiple partitions can also be reach by recursively computing the spectral decomposition for each part of the image before the interpolation process, as depicted in Figure 9. In fact, the recursive process may be driven by the user, who can specify the highest level of recursion, moreover, the user can brush any pieces of the image during each one of recursion steps in order to better set weights and thus improve the segmentation quality (see the next section for details).

Figure 10 shows the result of applying our methodology to segment a fingerprint image. For the sake of comparison, we show in Fig. 10(a) the result of computing our framework directly from the original image $\mathcal{I}$, that is, skipping the CTD, while Fig.10(b) depicts the result using CTD and the classical weighting metric [7] instead of Eq.(4) to define the graph weights. Notice, from Fig.10(c), how better the segmentation is when Eq.(4) and CTD are combined.

5) Interactive Weight Manipulation: Weights can be interactively tuned so as to force the spectral cut to accurately fit boundaries between textured regions of the image. Our tuning scheme relies on the texture component $\mathcal{T}$ obtained from the cartoon-texture decomposition. The component $\mathcal{T}$ is processed by an harmonic analysis tool called *Wave Atoms* [45], [46]. Wave atoms-based techniques bear high directional anisotropy



(a) Initial segmentation (b) Brush made by user (c) Final result

Fig. 11. Improving segmentation of a noise-textured image.

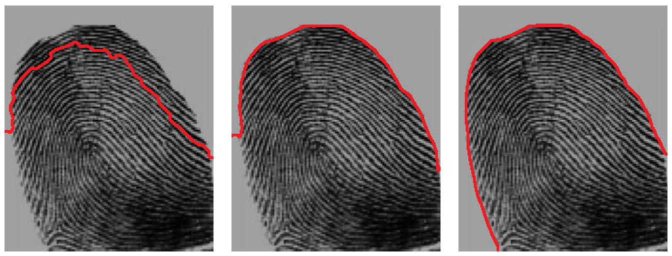
and sensitivity to noise, which makes them suited to identifying oscillatory patterns in high-frequency domains. In our approach, we use this tool to assign a scalar $S(\mathcal{T}_i) \in [0, 1)$ to each pixel $\mathcal{T}_i$ of $\mathcal{T}$, where values close to "one" means the pixel belongs to the "wave" of a texture pattern, similar to that used in [54] to produce a texture mapping. Therefore, pixels nearby the boundary between two textured regions tend to be identified as not belonging to a texture wave, thus assuming values close to "zero".

Starting from this premise, the weights of edges incident to pixels brushed by the user are modified as follows:

$$w_{ij} = \rho \left(\min_{e_{ij} \in E, w_{ij} \neq 0} w_{ij} \right) \left(1 - \max\{S(\tilde{\mathcal{T}}_i), S(\tilde{\mathcal{T}}_j)\} \right), \quad (7)$$

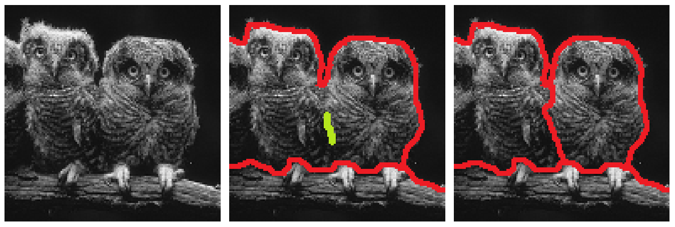
where the constant $\rho \in (0, 1)$ is the smallest non-zero weight of the edges in G obtained during the automatic spectral cutting performed on $\tilde{\mathcal{C}}$, and $\tilde{\mathcal{T}}$ is the coarse version of $\mathcal{T}$. The constant ρ enforces a more drastic change of weights in the region stroked by the user since it holds the new weights that will have the lowest possible non-zero value within the target graph.

Figure 11 shows the result of segmenting a noisy image using our method setting 10 partitions. Notice from Fig. 11(a) that most parts of the image is accurately segmented, attesting the accuracy of the proposed method for the case where the image contains texture and moderate gaussian noise. The spectral cut deviates from the correct boundary in just a few small regions which are easily fixed through user interaction, as depicted in Fig.11(b) and Fig. 11(c). This post-segmentation was only feasible because the texture mapping used to accomplish this task is sensitive to noise. Figure 12 shows that is not necessary to perform a large number of user interventions to achieve the desired segmentation. The simple greenish stroke depicted on the texture region between the two owls were enough to separate the birds, as depicted in Fig.12(c).



(a) Without CTD (b) Without our metric (c) Complete pipeline

Fig. 10. Automatic result with the proposed framework.



(a) Original image (b) Small stroke (c) Final result

Fig. 12. A simple stroke (greenish region between the two owls) is sufficient to improve the segmentation.

C. Results, Comparisons and Evaluation

Now we present the results obtained with the proposed framework and a comparative study involving four other state-of-the-art methods. We split this section into three classes of experiments:

1) *Comparison with Automatic Methods.*
2) *Evaluation using Ground-Truth Images.*
3) *Comparison with Interactive Methods.*

The following parameters were used in all experiments presented in this section: $\lambda = 0.05$ and $\mu = 0.1$ in the cartoon-texture decomposition, the default parameters suggested in [50] for the bicubic interpolation and a hard threshold at 3σ (noise-to-signal ratio of the image) combined with cycle spinning [45] for the wave atom transform. We set $r = 1$ and $\eta = 5$ in Equations (3)-(4), respectively. Finally, in order to check the segmentation quality of the proposed approach, we provide comparisons against two automatic and two user-assisted eigenspectrum-based techniques:

- k-way Normalized Cuts method (NCut)[2] [7], [32];
- Multiscale Normalized Cuts method (MS-NCut)[3] [34] (with radius 2.3);
- Random Walker-based Segmentation with pure Eigenvector Precomputation (RWS-EP) [39];
- Random Walker based on Eigenvector Precomputation and Prior scheme (MSFP)[4] (with 80 precomputed eigenvectors) [40];

1) Comparison with Automatic Methods: The first experiment shown in Figure 13 presents a comparative analysis of our technique against the non-interactive NCut and MS-NCut approaches. We can see that both classical NCut (Fig. 13(b)) and MS-NCut (Fig. 13(c)) badly segment parts of the image. Our approach results in a better partitioning (Fig. 13(d)), although some regions are also segmented in an incorrect way. After user intervention, shown in Fig. 13(e), the result improves considerably (Fig. 13(f)).

[2]available at http://note.sonots.com/SciSoftware/NcutImageSegmentation.html
[3]available at http://www.cis.upenn.edu/~jshi/software/
[4]available at http://fastrw.cs.sfu.ca

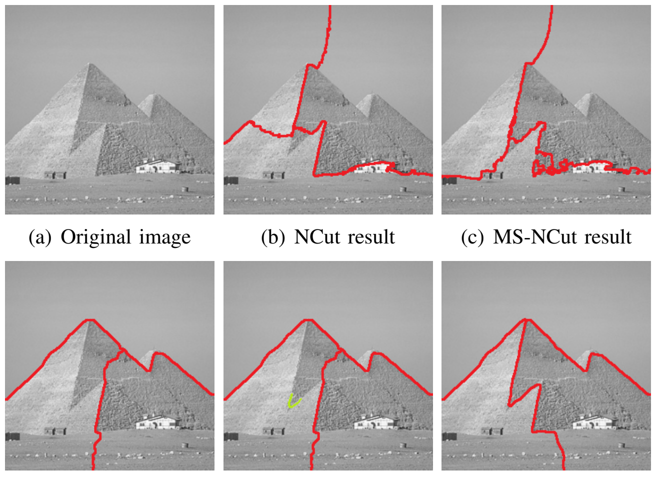

(a) Original image (b) NCut result (c) MS-NCut result
(d) Our initial result (e) With brush (green) (f) Final segmentation

Fig. 13. The infl ence of the user intervention in comparison with static approaches.

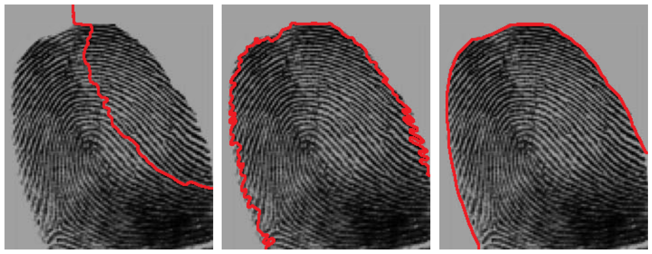

(a) NCut partition (b) MS-NCut partition (c) Our result

Fig. 14. The result of applying NCut, MS-NCut, and the proposed approach (in automatic mode) in a fingerprint image.

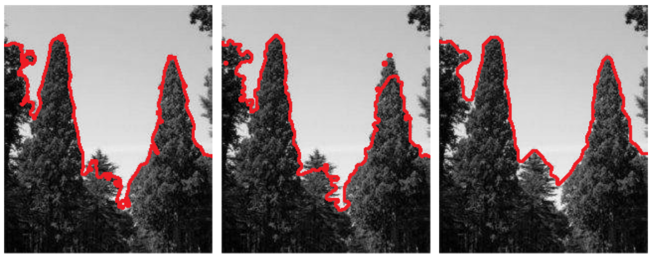

(a) NCut result (b) MS-NCut result (c) Our result

Fig. 15. Our approach (in automatic mode) produces smoother segmentation curves when compared to NCut and MS-NCut.

The result of applying the three above-mentioned methods in a fingerprint image for two partitions is shown in Figure 14. Notice that the NCut (Fig. 14(a)) does not segment the fingerprint correctly while the MS-NCut and our approach do a good job. It is easy to see from Fig. 14(b) that the MS-NCut tends to produce a jagged segmentation curve while our method results in a smoother curve, as shown in Fig. 14(c). It becomes clear from Figure 15 that the smoothness of the result produced by our approach also help increase robustness. While NCut and MS-NCut tend to generate a segmentation curve with many artifacts and some cognition errors, our approach produces a much pleasant result.

2) Evaluation using Ground-Truth Images: We evaluate the performance of the proposed methodology by means of quantitative and visual analysis against the NCut and MS-NCut algorithms. To accomplish the numerical evaluation, we make use of *Recall*, *Precision* and *F-Score* measures [55] on the well-known *Berkeley Segmentation Dataset (BSDS)* [56], which provides 300 natural images with their human-drawn ground-truth segmentations organized by a large number of different human subjects. The metrics are defined as follows:

Recall: is computed by the following expression

$$\text{Recall} = \frac{\text{Matched}(S_{target}, S_{source})}{|S_{target}|}, \tag{8}$$

where S_{source} represents the ground truth segmentation (obtained from the average of the provided BSDS segmentations), S_{target} the partitioning to be evaluated and $|\cdot|$ indicates the total number of boundary pixels in the current segmentation. Recall (8) can be understood as the proportion of boundary pixels in S_{target} for which it is possible to find a matching boundary pixel in S_{source}.

Precision: holds the opposite situation, that is,

$$\text{Precision} = \frac{\text{Matched}(S_{source}, S_{target})}{|S_{source}|}. \qquad (9)$$

F-score: is a standard statistical measure that summarizes the recall and precision measures into a unified metric. It accounts for the trade-off between sensitivity and positive predictive values:

$$F\text{-Score} = 2\frac{\text{Precision Recall}}{\text{Precision} + \text{Recall}}. \qquad (10)$$

Figure 16 presents the computation of Recall (8), Precision (9) and the F-Score (10) for the BSDS dataset. One notices that our framework is better than other methods for most of the metrics, being fairly stable in all cases. In fact, only the recall quantity leads to a similar behavior between the MS-NCut and the proposed method. Figure 17 depicts the partitioning produced by NCut, MS-NCut, and our approach when applied to illustrative images randomly extracted from the BSDS dataset [56]. Notice that the MS-NCut and our method produced much better results than the classical NCut (the ground truth is shown in the last column). In contrast to MS-NCut, our method has two advantages: it produces smooth boundaries between the segmented regions and it clusters the image into slightly wider regions, two characteristics also present in the ground truth images. Moreover, it can be seen that our approach is more robust to identify objects and structures contained in the images. For instance, the input images *monk*, *geisha*, and *horse* were better captured by our technique.

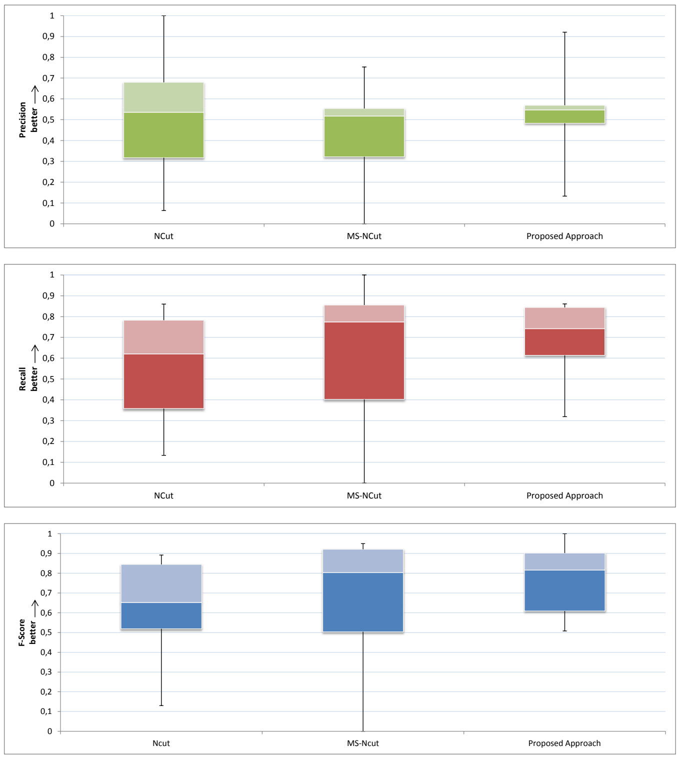


Fig. 16. Quantitative comparison for the Recall, Precision and the F-score segmentation measures. For all images, we compute the average of the recall and precision quantities for $R = 8, 9, 10, 11$ and 12 [55].

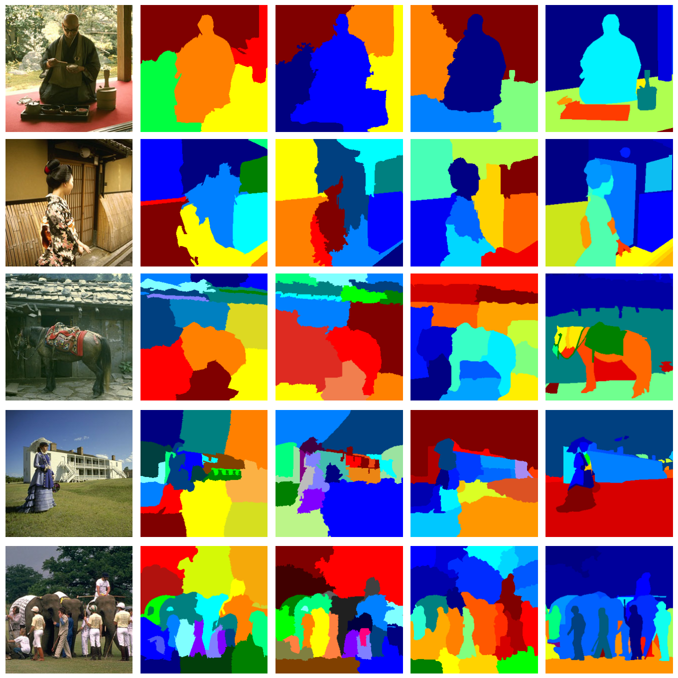


Fig. 17. From left to right: The input image, the segmentations resulting from NCut, MS-NCut, our approach and the human-drawn ground-truth.

3) Comparison with Interactive Methods: The results provided by the proposed methodology considering user intervention has been compared against two user-assisted eigenspectrum-based methods, RWS-EP [39] and RWS-EPP [40], as depicted in Figure 18. In contrast to the Random Walker-based techniques RWS-EP and RWS-EPP, our method does not require an initial user setup to produce the first segmentation result.

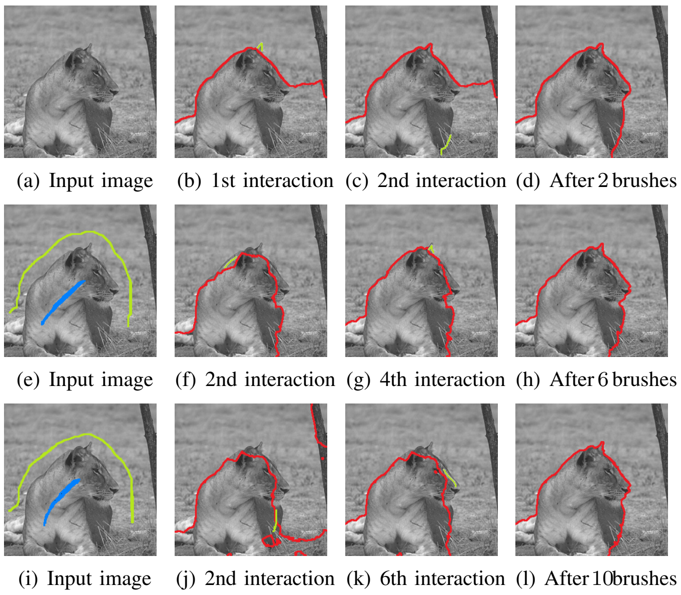


Fig. 18. Segmentation driven by user produced by our technique (top row), RWS-EP (middle row) and RWS-EPP (bottom row). Fig. 18(a) is required by our approach, while the target image and the initial setup with 2000 seeds (blue and green pieces) must be provided by the user in random walker-based methods (Figs. 18(e) and 18(i))

TABLE I. COMPARATIVE TIMING TABLE (IN SECONDS) WITH RESPECT TO EXPERIMENT INVOLVING USER AUTONOMY (FIG. 18, IMAGE DIMENSION: 256×256). THE TIMING $+4$ MEANS THE TIME TAKEN FOR THE INCLUSION OF SEEDS.

Processing stage	**RWS-EP**	**RWS-EPP**	**Our method**
To produce the 1st segmentation	23 (+4)	21 (+4)	5
Average time per interaction	0.5	0.6	0.4
To produce the desired result	29.4	32.3	5.83

Notice that differently from RWS-EP and RWS-EPP, which require much more user interference to reach a pleasant result (Figs. 18(h) and 18(l)), our approach needed only two brushes to yield the desired result (Fig. 18(d)). Besides, our technique is considerably faster when compared against RWS-EP and RWS-EPP (see Table I).

III. SEEDED IMAGE SEGMENTATION

A growing number of semi-supervised image segmentation methods have been proposed in the last few years, motivated mainly by the human capability of recognizing and detecting patterns. In fact, *seeded image segmentation* figures among the most relevant image segmentation methodologies, where traditionally algorithms make use of user's prior knowledge as input data in order to operate suitably. Seed-based image segmentation methods typically rely on a given set of labeled pixels ("seeds") and on affinity weighted graphs whose nodes correspond to image pixels and edges reflect the neighborhood structure of the pixels. Edge weights encode image attributes such as texture, color or gradients and they are used to guide the propagation of the seeded labels throughout the image.

Many mathematical formulations and algorithms have been recently proposed to perform segmentation by seed propagation [57], [5], [4], [58], [59], [6], [60], [61], [62], [63], most of them making use of energy functional minimization on graphs. In terms of energy minimization formulation, [64], [65] showed that most seed-based image segmentation methods can be understood as variations of a small group of basic techniques which differ from each other in terms of their mathematical formulation, pairwise pixel distance and weight computation. In fact, those algorithms were reinterpreted as special cases of the following generalized energy functional [65]:

$$E_{PWS}(\mathbf{x}) = \sum_{i \in B} w_{Bi}^p |x_i - x_B|^q + \sum_{i \in F} w_{Fi}^p |x_i - x_F|^q + \frac{1}{2} \sum_{i \in V} \left(\sum_{j \in N(i)} w_{ij}^p |x_i - x_j|^q \right), \tag{11}$$

where x_i is the sought solution, that is, the saliency map w.r.t. target image, w_{ij} is the weight assigned to the edge (i, j), x_B and x_F represent the values of seeded pixels in the background and foreground, w_{Bi} and w_{Fi} are their penalizations, respectively, and the constant p and q are positive parameters. Functional (11) assumes the form of specific image segmentation algorithms when p and q vary as shown in Table II.

TABLE II. VARIATION OF THE PARAMETERS p AND q IN EQUATION (11) VERSUS CLASSICAL SEGMENTATION ALGORITHMS [65].

q\p	0	**finite**	∞
1	Seed collapse	*Graph Cuts*	*P. Watershed* ($q = 1$)
2	l_2 norm	*R. Walker*	*P. Watershed* ($q = 2$)
∞	l_1 norm	l_1 norm	*S. Path Forest*

Most of the work from the literature particularly focus on the four major groups mentioned in Table II, more specifically:

1) *Graph Cuts-based methods* [66], [57], [4], [58], [67];
2) *Random Walker-based methods* [5], [68], [39], [40];
3) *Watersheds-based methods* [69], [6], [64], [70], [65];
4) *Shortest Path Forest/Geodesic-based methods* [71], [72], [73], [74], [75], [76],

which minimize the same energy functional whose formulation takes into account only first-order pairwise pixels, differing only in terms of exponent values. Furthermore, most existing methods from this set rely on non-quadratic energies, thus demanding the use of sophisticated and computationally cost optimization tools. Ensuring accuracy and smooth solution is also an issue for existing techniques.

A. Laplacian Coordinates for Image Segmentation

Aiming at dealing with the issues raised above, in this section we present a new technique for seeded image segmentation, first reported in [3], which relies on the minimization of a novel quadratic energy functional defined on an affinity graph. The new approach, called *Laplacian Coordinates*, allows for user intervention while leading to smoother and accurate solutions.

The notion of Laplacian Coordinates has initially appeared in [77], [78] to address the problem of surface editing in the field of *Geometry Processing*. In contrast to most existing algorithms, in particular the four ones mentioned earlier that minimize the "distance" between pairwise pixels, the proposed approach minimizes the average of distances while controlling anisotropic propagation of labels during the segmentation process. Therefore, pixels sharing similar attributes are kept closer to each other while jumps are naturally imposed on the boundary between image regions, thus ensuring better fitting on image boundaries as well as a pretty good neighborhood preservation (on average). Moreover, the proposed formulation is guaranteed to have a unique solution, an important trait not always present in seed-based image segmentation algorithms. Other fundamental characteristic of Laplacian Coordinates is that the minimizer of the cost function is given by the solution of a constrained system of linear equations, making the algorithm quite simple to be used and coded.

The Laplacian Coordinates pipeline is very simple and comprises four main steps: *Definition of Seeds*, *Affinity Graph Building*, *Energy Functional Construction and Solution*, and *Assignment of Labels*. Figure 19 illustrates the use of the Laplacian Coordinates approach. In summary, the main contributions of the proposed technique are:

1) A novel and easy-to-implement formulation for seed-based image segmentation, which we call *Laplacian Coordinates* [3].

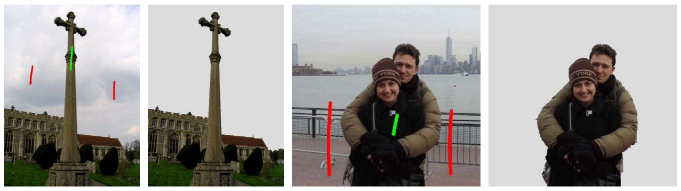


Fig. 19. Segmentations produced by the Laplacian Coordinates approach, where red and green scribes indicate the seedings.

2) Laplacian Coordinates bears several important properties such as boundary fitting, anisotropy and unique solution for the minimizer.
3) The segmentation simply consists of solving a constrained sparse linear system of equations.
4) A comprehensive set of quantitative and qualitative comparisons against state-of-art algorithms that shows the effectiveness of Laplacian Coordinates.

B. Computing the Laplacian Coordinates Energy on Graphs

Let I be a color or grayscale image. For a color image, we denote the RGB vector by $I_i = (R_i, G_i, B_i)$, which indicates the luminance of *red*, *green* and *blue* channels at pixel $P_i \in I$. For a grayscale image, I_i is the pixel intensity. As a basic tool to proceed with the segmentation task, we construct a weighted graph $G = (V, E, W_E)$, where V is the set of nodes $i \in V$ corresponding to the pixel $P_i \in I$, the edge set E corresponds to pairs of pixels locally connected in an 8-neighborhood, and $w_{ij} \in W_E$ is the weight assigned to edge (i, j) of the graph. The set $N(i) = \{j : (i, j) \in E\}$ represents the indices of the pixels P_j that share an edge with pixel P_i and the quantity

$$d_i = \sum_{j\in N(i)} w_{ij} \tag{12}$$

measures the weighted valency associated to pixel P_i.

1) Graph Weight Setup: There are many different ways to define the set of weights W_E, many of them relying on pixel intensity, gradient, scalability, saliency and contour information [34], [35], [79], [2], [42], [80]. Aiming at keeping our approach as simple as possible, we only consider pixel intensities to define the weights. More precisely, the weight $w_{ij} = w(P_i, P_j)$ assigned to each pixel pair (P_i, P_j) is computed as follows:

$$w_{ij} = \exp\left(-\frac{\beta||I_i - I_j\|_\infty^2}{\sigma}\right), \quad \sigma = \max_{(i,j)\in E} ||I_i - I_j||_\infty, \tag{13}$$

where β is a tuning constant. Notice that the weights are positive and symmetric in the sense that $w_{ij} = w_{ji}$. In terms of usage, a small constant $\epsilon = 10^{-6}$ is added into Equation (13) to avoid null weights, as suggested by [61]. Sophisticated weighting functions can also be employed to reach more refined segmentations such as the inner-product similarity metric (4).

2) Laplacian Coordinates Energy Functional: Given the affinity matrix $\mathbf{W}$ computed as in Equation (13), the set of background B and foreground F seeded pixels and their corresponding labels x_B and x_F, the following energy functional $E(\mathbf{x})$ is minimized with respect to vector $\mathbf{x}$:

$$\underbrace{\sum_{i\in B}\|x_i - x_B\|_2^2}_{\text{fidelity term}} + \underbrace{\sum_{i\in F}\|x_i - x_F\|_2^2}_{\text{fidelity term}} + \underbrace{\sum_{i\in V}\left\|d_i x_i - \sum_{j\in N(i)} w_{ij}x_j\right\|_2^2}_{\text{LC term}} \tag{14}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)^t$ is the sought solution, that is, the scalar values assigned to the pixels $(P_1, P_2, ..., P_n)$ so as to minimize the functional $E(\mathbf{x})$, n is the number of pixels and w_{ij} is computed as in Eq. (13). Without loss of generality, assume that $x_B > x_F$. Once the Energy (14) is minimized, the segmentation is accomplished by assigning background and foreground labels $y_i \in \{x_B, x_F\}, i \in V$, as follows:

$$y_i = \begin{cases} x_B, & \text{if} \quad x_i \geq \dfrac{x_B + x_F}{2} \\ x_F, & \text{otherwise} \end{cases}. \tag{15}$$

Laplacian Coordinates energy functional (14) is made up of two main components, one accounting for the constraints imposed by the seeds in B and F, called *fidelity term*, and a second component controlling the label spread in the neighborhood of each pixel, called *LC energy term.* In matricial notation, LC energy term can be rewritten as follows:

$$\sum_{i\in V}\left\|d_i x_i - \sum_{j\in N(i)} w_{ij}x_j\right\|_2^2 = \sum_{i\in V}\left(\sum_{j\in N(i)} w_{ij}(x_i - x_j)\right)^2 =$$
$$\underbrace{\left(\sum_{j\in N(1)} w_{1j}(x_1 - x_j), \ldots, \sum_{j\in N(n)} w_{nj}(x_n - x_j)\right)}_{\mathbf{v}^t} \cdot \mathbf{v} = (\mathbf{Lx})^t(\mathbf{Lx})$$

Thus,

$$\sum_{i\in V}\left\|d_i x_i - \sum_{j\in N(i)} w_{ij}x_j\right\|_2^2 = (\mathbf{Lx})^t(\mathbf{Lx}) = \|\mathbf{Lx}\|_2^2, \tag{16}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the *graph Laplacian matrix*, $\mathbf{D}$ is the diagonal matrix where $D_{ii} = d_i$ (see Equation (12)) and $\mathbf{W}$ denotes the weighted adjacency matrix of the graph, that is,

$$W_{ij} = \begin{cases} w_{ij}, & \text{if} \quad (i,j) \in E \\ 0, & \text{otherwise} \end{cases}. \tag{17}$$

Notice that the i-th component of $\mathbf{Lx}$ corresponds to the *differential/average operator*:

$$\delta_i = x_i - \frac{1}{d_i}\sum_{j\in N(i)} w_{ij}x_j, \text{ that is, } (\mathbf{Lx})_i = d_i\delta_i. \tag{18}$$

In less mathematical terms, quantity δ_i measures how much each node deviates from the weighted average of its neighbors.

3) Minimizing the Energy Functional: Energy (14) can be modeled in a more general matricial form, that is,

$$\begin{aligned} E(\mathbf{x}) &= \sum_{i\in B}(x_i - x_B)^2 + \sum_{i\in F}(x_i - x_F)^2 + \|\mathbf{Lx}\|_2^2 \\ &= \sum_{i\in S} x_i^2 - 2\left(\sum_{i\in B} x_i x_B + \sum_{i\in F} x_i x_F\right) + c + (\mathbf{Lx})^t(\mathbf{Lx}) \\ &= \mathbf{x^t L^2 x} + \mathbf{x^t I_S x} - 2\mathbf{x^t b} + c \end{aligned}$$

$$E(\mathbf{x}) = \mathbf{x^t}(\mathbf{I_S} + \mathbf{L^2})\mathbf{x} - 2\mathbf{x^t b} + c, \tag{19}$$

where $\mathbf{I_S}$ is a diagonal matrix such that $I_S(i,i) = 1, i \in S = B \cup F$, and zero, otherwise, $\mathbf{b}$ is the vector where $b(i) = x_B, i \in B$; $b(i) = x_F, i \in F$, and zero otherwise, and c is a constant. Quadratic form (19) has a unique minimizer since $(\mathbf{I_S} + \mathbf{L^2})$ is a symmetric and positive definite matrix. Moreover, its minimizer vector $\mathbf{x}$ is the solution of the following linear system [81]:

$$(\mathbf{I_S} + \mathbf{L^2})\mathbf{x} = \mathbf{b}. \tag{20}$$

Therefore, minimizing $E(\mathbf{x})$ is equivalent to solving the linear system (20), which, in turn, holds quite attractive properties such as symmetry, positive definiteness and sparsity. After solving Eq. (20), the segmentation is then performed by assigning a label to each image pixel according to Eq. (15).

4) Laplacian Coordinates Properties: Besides being computationally efficient, easy-to-implement and ensuring unique solution, the proposed method has additional properties that render it attractive to segment images, as discussed below.

Boundary and Constraint Fitting. The main characteristic that differs Laplacian Coordinates from other seed-based approaches is its capability to better propagate the seeds (constraint information). Figure 20 illustrates this by comparing Laplacian Coordinates against the Random Walker approach in one-dimensional case. First row of Figure 20 shows a 1D graph with 500 nodes ordered from left to right. Second row in Figure 20 shows two different distributions of edge weights: on the left, unitary weights are assigned to edges, except for edges in the middle of the graph, where weights have a distribution that decreases and gets close to zero increasing back to 1. On the right, weights are distributed similarly, but now with two picks isometrically arranged. Constraints (seeds) are imposed in the yellow and purple nodes. As one can easily see on the third row of Figure 20, Laplacian Coordinates spread the constraint information in a smoother way, taking longer to diffuse the constraint information when compared to Random Walker approach. For the sake of illustration, last row in Figure 20 presents the result of applying Laplacian Coordinates and Random Walker when weights are set equal 1.

Solution in Terms of Extended Neighborhood. An interesting interpretation of the solution of Laplacian Coordinates is that each pixel x_i is written not in terms of the first-order neighbors, but considering distant neighborhoods, instead. In mathematical terms, at an unconstrained pixel P_i, we have:

$$(\mathbf{Lx})_i = \frac{1}{d_i} \sum_{j \in N(i)} w_{ij} (\mathbf{Lx})_j \tag{21}$$

where $(\mathbf{Lx})_j$ is computed as in Equation (18). The solution x_i is then computed taking into account an extended neighborhood, mathematically expressed by the equation:

$$\begin{aligned} x_i &= \frac{1}{d_i} \sum_{j \in N(i)} w_{ij} \left(x_j + \frac{\delta_j}{d_i} \right) \\ &= \frac{1}{d_i} \sum_{j \in N(i)} w_{ij} x_j + \frac{1}{d_i^2} \sum_{j \in N(i)} w_{ij} \left(\sum_{p \in N(j)} w_{jp}(x_j - x_p) \right) . \end{aligned}$$

Therefore, information coming from the constraints takes longer to be diffused by the Laplacian Coordinates approach.

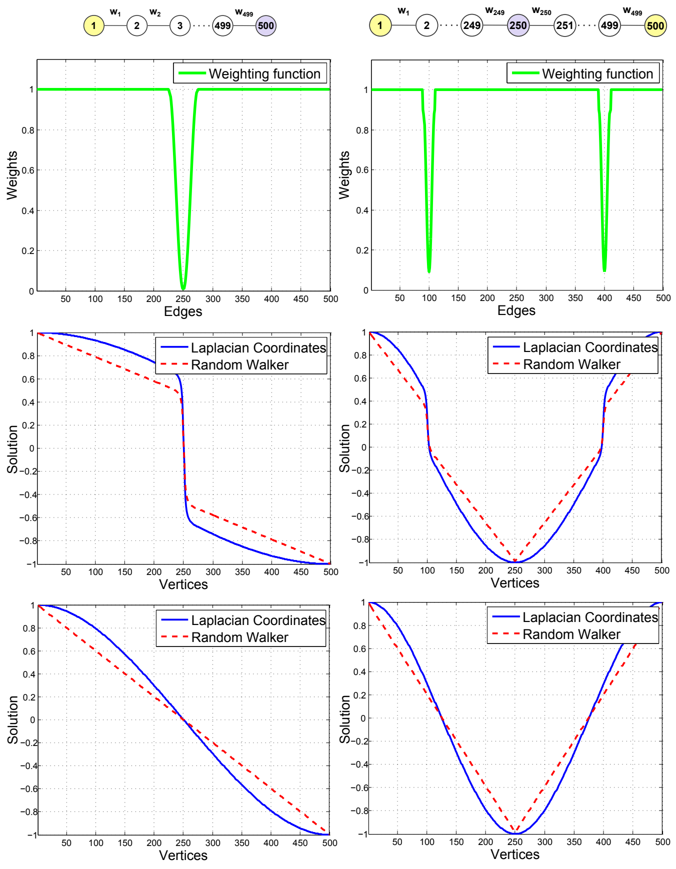


Fig. 20. Comparison between the solution obtained from Laplacian Coordinates and the classical Random Walker algorithm under the same initial conditions. Line graphs are shown in the top row with seeded vertices in yellow and purple while the corresponding edge weights are shown in the second row. The solution with and without the mentioned weights are given in the third and fourth rows.

Seeding Flexibility and Adaptability. Figure 21 shows the robustness of Laplacian Coordinates in producing different segmentations by just selecting distinct targets in the image. Notice from the two initial configurations (left and middle columns) of Fig. 21 that both objects (the boys) are accurately segmented, attesting the accuracy of the proposed approach. In fact, an even more general solution can be obtained by simultaneously seeding the two targets of the image, as depicted in the last column of Fig. 21.

Multiple-Region Segmentation. Laplacian Coordinates can also easily be extended to segment an image into several parts. This extension is carried out by simply solving $(N-1)$ system of linear equations similar to Equation (20):

$$(\mathbf{I_S} + \mathbf{L^2})\mathbf{x^{(j)}} = \mathbf{b^{(j)}}, \tag{22}$$

but setting $I_S(i,i) = 1$ for all seeded pixels in the image and specifying different $b^{(j)}$ for each one of the given labels $K_j \in K = \{K_1, K_2, ..., K_N\}$, $1 \leq j \leq (N-1)$. Let C be a positive constant. We set $b_i^{(j)} = C$, $i \in K_j$, $b_i^{(j)} = -C$, $i \in (K \backslash K_j)$, zero, otherwise. Assuming that all $\mathbf{x^{(j)}}$ are bounded by $[-C, C]$, the last scalar map $\mathbf{x^{(N)}}$ is then obtained as follows:

$$x_i^{(N)} = C - \max_{1 \leq j \leq (N-1)} \{x_i^{(j)}\} . \tag{23}$$

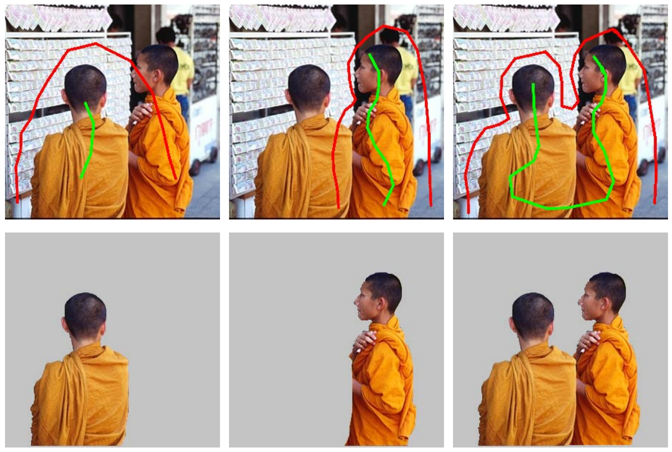


Fig. 21. Selecting different objects from the image by exploiting the seed sensitivity of the Laplacian Coordinates. First row: multiple selections are given as input to the method and Second row: the corresponding segmentations.

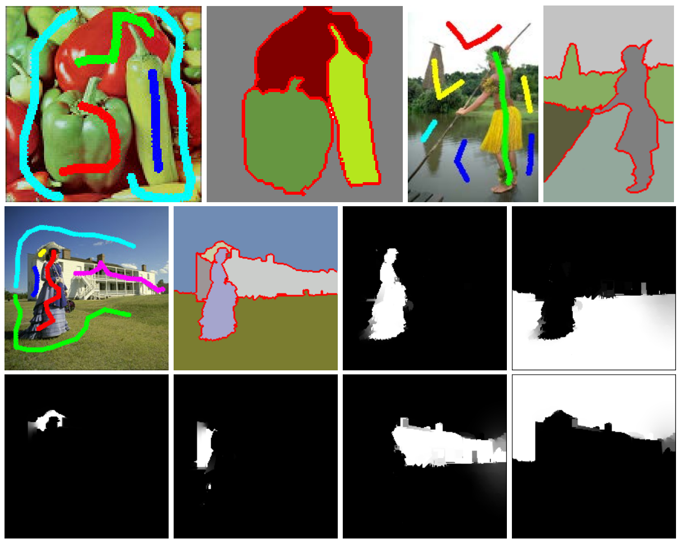


Fig. 22. Extension of the Laplacian Coordinates (14) for multiple segmentation. First row: multiple seeds are sketched as colored strokes, from which Laplacian Coordinates produced the multiple segmented regions. Middle and bottom row: sketched seeds, the final segmentation and the six solution vectors $\mathbf{x}^{(\mathbf{j})}$ that give rise to the multiple segmentation.

Finally, for each $j : 1 \leq j \leq N$, the segmentation $y^{(j)}$ (a binary image) is performed by

$$y^{(j)} = \bigcap_{\substack{p=1,\ldots,N \\ p\neq j}} \left(x^{(j)} > x^{(p)}\right), \tag{24}$$

where $>$ is computed for all pixels of the image. Figure 22 depicts the result of Laplacian Coordinates to segment multiple regions. Color strokes mark the objects (strokes with the same color correspond to the same region), from which Laplacian Coordinates generates the segmentation in multiple regions.

C. Results, Comparisons and Evaluation

In this section we provide a comprehensive experimental evaluation of Laplacian Coordinates against competing state-of-the-art techniques. We evaluate each method using multiple measures traditionally employed by the image segmentation community.

In order to perform the experimental analysis, we use one of the most popular benchmark dataset for seeded-based segmentation: the *"Grabcut" dataset* [57]. This dataset contains 50 prototype images, their ground-truth (obtained from manual human segmentation), and seeded maps marking foreground and background regions of the images. We make use of this benchmark dataset to compare the Laplacian Coordinates approach (LC) [3] against five traditional seed-based segmentation algorithms:

- Graph Cuts algorithm (GC) [57], [4];
- Power Watershed algorithm (PWS) [82], [65];
- Maximum Spanning Forest with Kruskal's algorithm (MSFK) [6], [65];
- Maximum Spanning Forest with Prim's algorithm (MSFP) [6], [65];
- Random Walker algorithm (RW) [5].

Quantitative evaluations are performed comparing the output quality in terms of object/region detection as well as the accuracy in preserving ground-truth boundaries.

1) Region Quality Metrics: We employ three distinct region quality metrics to gauge the quality of Laplacian Coordinates against others, namely

Rand Index (RI): measures the closeness between the output segmentation S and the ground-truth G by counting the number of pixel pairs that have the same label [83]. More formally, it computes the sum of the pixel pairs that share a common label in S and G and those that share distinct labels in both images, divided by the total number of pixel pairs.

Global Consistency Error (GCE): computes how much the segmentation can be interpreted as a refinement of other, forcing the local refinements to be in the same orientation [44]. In fact, GCE does not take into account the scale of images to proceed with the measurements, ensuring consistency even when comparing different scales. Lower values are better.

Variation of Information (VoI): quantifies the distance between ground-truth and segmentation in terms of their relative entropies [84]. More precisely, it quantifies the amount of randomness information in changing from one clustering to another. Moreover, VoI is properly a metric in the sense of the theory of linear algebra satisfying the positivity, symmetry and the triangle inequality [85], [86]. Values close to 0 are better.

2) Boundary Quality Metrics: We also consider in our experiments two popular metrics to measure the quality of the boundaries detected by the segmentations, that is,

Boundary Displacement Error (BDE): quantifies the average of the displacement error taking into account the amount of boundary pixels from the segmentation result that correspond to the closest boundary pixels from the ground-truth [87].

The Harmonic Average Score (F-Score): summarizes the *Recall* and *Precision* baseline metrics [55], [56], which measure how much the segmentation matches the ground-truth boundaries (see Section II-C). The matching is established in terms of the boundary pixel proximity for different values of radius R, as proposed in [55].

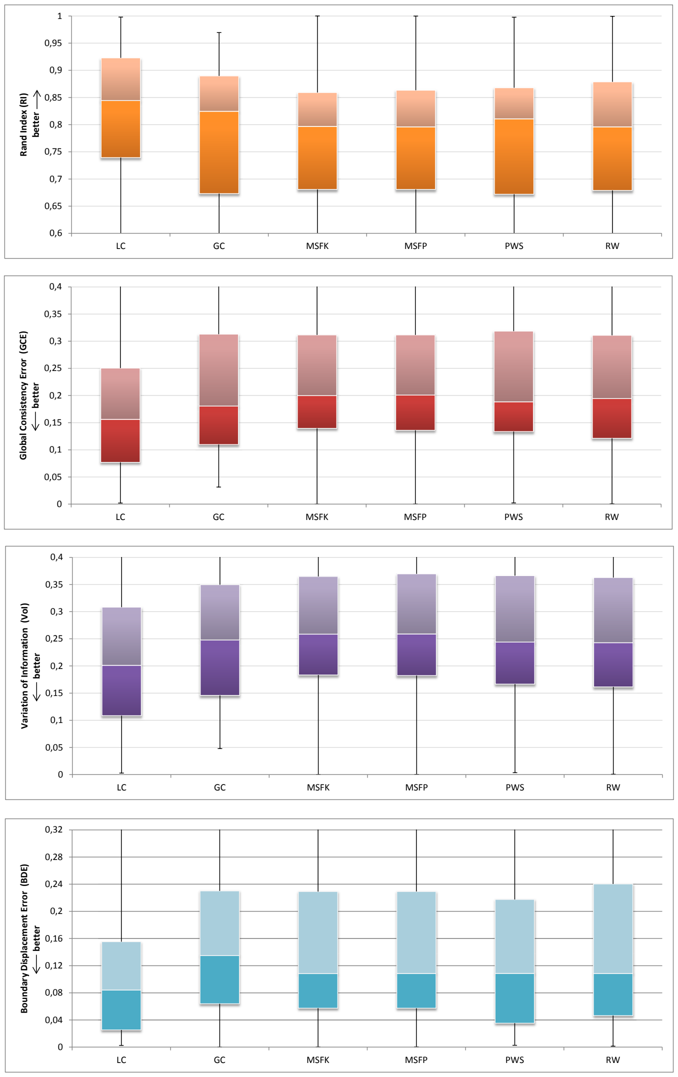


Fig. 23. Comparison of six seed-based segmentation methods regarding to RI, GCE, VoI and BDE quality metrics. In all cases, the proposed Laplacian Coordinates framework outperforms all other five evaluated techniques.

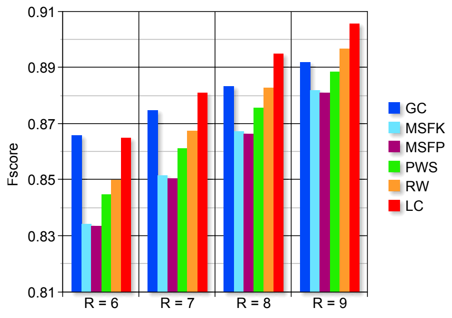


Fig. 24. *F-score* quality metric. Laplacian Coordinates is considerably better than other methods when parameter R increases.

Figure 23 summarize the quantitative results of the metrics RI, GCE, VoI and BDE for the Microsoft “Grabcut” dataset. Notice that the Laplacian Coordinates clearly outperforms the other five methods in all quality metrics, being also fairly stable to those metrics.

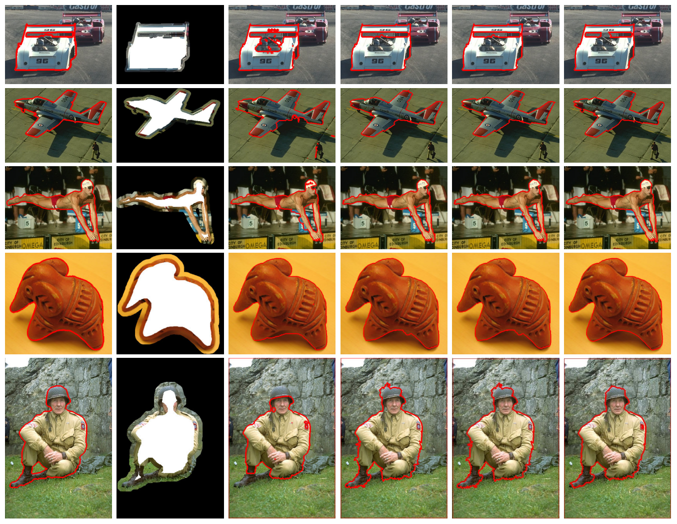


Fig. 25. From left to right: Ground-truth, the tri-map images (seeds and unknown region) provided from Grabcut dataset, and the segmentation results from GC, MSFK, MSKP and LC approach.

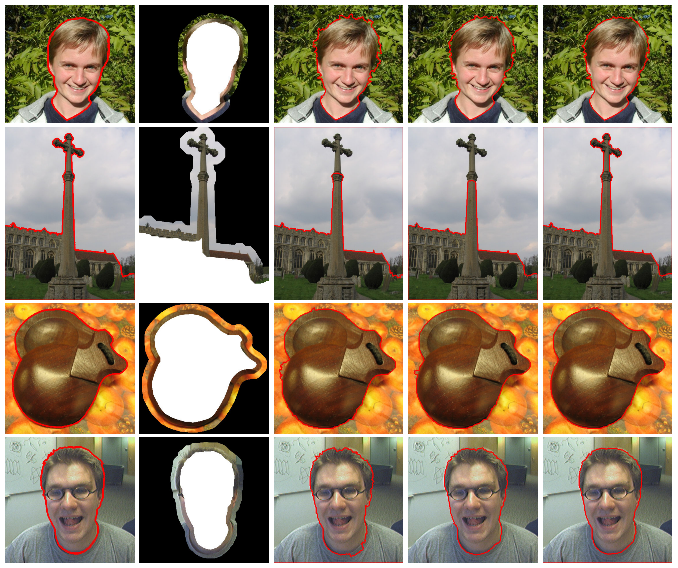


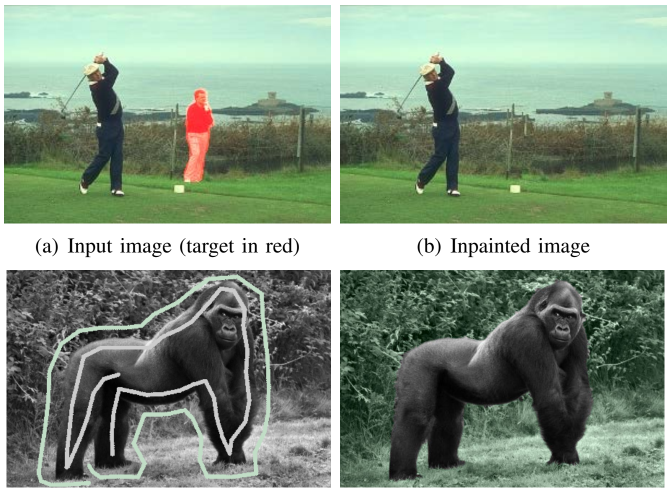
Fig. 26. From left to right: Ground-truth, tri-map images (seeds and the unknown region), and segmentations from PWS, RW and LC approach.

Regarding *F-score*, the LC approach also presents very good performance, specially when the parameter R increases. As one can see from Figure 24, the proposed approach shows a better *F-score* than other techniques, outperforming all for R equal or bigger than 7. These quantitative results show the effectiveness of Laplacian Coordinates as a seeded image segmentation method.

Figures 25 and 26 present qualitative results comparing GC, MSFK, MSFP, PWS and RW against Laplacian Coordinates. One can see that, besides accurately capturing boundaries, Laplacian Coordinates tends to simultaneously generate smoother and better fit boundary curves, a characteristic not present in any other approach, which are less accurate while still producing more jagged boundary curves.

IV. Image Inpainting and Photo Colorization

This section presents the use of the proposed segmentation approaches in two problems typically studied by the computer vision and engineering communities: *image inpainting* and *image colorization*. Image inpainting seeks to recover the natural aspect of an image where data has been partially damaged or occluded by undesired objects. Photo colorization is a computer-assisted process by which grayscale images or black-and-white films are properly colored (see Figure 27 for an illustrative example).



(a) Input image (target in red) (b) Inpainted image

(c) Input image (with color scribbles) (d) Colored image

Fig. 27. Image inpainting and colorization using the proposed frameworks.

Both inpainting and colorization applications share the common underlying idea that how accurately the objects to be filled or colorized are segmented. Therefore, build a good segmentation scheme into those applications is essential to ensure the success of those tasks. In this spirit, we introduce two new techniques that address the problem of image inpainting and photo colorization which make use of our segmentation tools as starting point to reach their final goal. We also compare the effectiveness of those techniques against representative state-of-the-art methods for a variety of synthetic and real images.

A. *Interactive Image Inpainting using Laplacian Coordinates*

Image inpainting is a modern research topic that has received a great deal of attention in recent years. It focuses on studying restoration and disocclusion processes for damaged digital images and artistic edition finalities. Methods devoted to perform inpainting can be arranged in several groups, as suggested by the surveys [88], [89], [15]. In short words, existing approaches differ in terms of pixel propagation, sensibility when synthesizing textures, and filling order criterion [88].

Although techniques for performing image inpainting vary in many fundamental aspects, a common drawback not covered by most inpainting systems is that they require the user to manually "carving" the targets to be edited. Selecting those targets consists of a meticulous process that demands great effort from users to precisely separate the targets from the image background. In fact, there are a few methods that address the problem of selecting the regions to be inpainted in an automatic or semi-supervised way. The techniques proposed by [90], [91], [92] automatically detect defects easily to identify visually but difficult to be segmented by hand. However, those algorithms can only handle a limited class of defects, thus constraining their application to specific problems. A method that covers a broader number of cases was proposed by [93], which introduces user knowledge into the pipeline of inpainting. The authors provide a user interface that allows users to guide the restoration by drawing straight lines on target regions. A similar user-steered interface was proposed by [94], which relies on the nearest neighbor correspondences among parts of the image. Their algorithm has recently been introduced in the *Adobe Photoshop Engine* as an interactive tool to perform image retouching. Since a great amount of user intervention is required to drive the whole inpainting process, the time for generating a pleasant result is considerable, specially when object to be inpainted is large.

Aiming at overcoming the issues mentioned above while providing a friendly and intuitive interface to select the regions to be recovered, we propose a novel framework (fist reported in [21]) that generates pleasant results with a reduced number of user interventions. Moreover, the non-interactive version of our technique, firstly reported in [15], outperforms existing inpainting methods in several aspects, as we show hereafter.

1) Pipeline Overview: In order to combine the accuracy and the high-adherence on image contours of the Laplacian Coordinates approach with the efficiency of the proposed inpainting technique, we design a new tool that allows users to easily brush the objects to be inpainted. As illustrated in Figure 28, our pipeline comprises four basic steps: *User-steered Selection of the Inpainting Domain*, *Pixel Filling Order Assignment*, and *Patch-based Pixel Replication*.

User-steered Selection of the Inpainting Domain. Let f be the target image, $\Omega \subset \mathbf{R}^2$ the region to be inpainted and $\partial\Omega$ its boundary. In our approach, entire objects can be easily segmented so as to avoid the meticulous election of the boundary pixels employed by tradicional approaches. The user selects a target region Ω by brushing on the object of interest and the marked pixels are used as seeds for the segmentation process. The image background must also be roughly marked to properly settle optimization constraints for the Laplacian Coordinates. Managing multiple regions is also allowed, since Laplacian Coordinates enables multiple segmentations simultaneously. Moreover, user can recursively steer the resulting partition towards reaching a higher quality result. Figure 28 (left) illustrates the above-mentioned scheme.

Filling Order Assignment. Our inpainting procedure starts by computing a mapping image u (cartoon image) from f using the anisotropic diffusion equation proposed in [95]. Similar to [54], [18], component u is obtained by numerically solving the following partial differential equation:

$$\frac{\partial f^{(t)}}{\partial t} = g|\nabla f^{(t)}|div\left(\frac{\nabla f^{(t)}}{|\nabla f^{(t)}|}\right) - (1-g)(f^{(t)} - f), \quad (25)$$

where $f^{(t)}$ is the scaled version of f, $g = g(|\nabla G_\sigma * f^{(t)}|)$ is an edge detection function, G_σ represents the gaussian function and σ is a tuning parameter. Figure 29 shows the original image, its cartoon version and the gradient fields computed from the rectangular samples highlighted in both images. As one can observe, the field derived from the cartoon image is better behaved than the one computed directly from f.

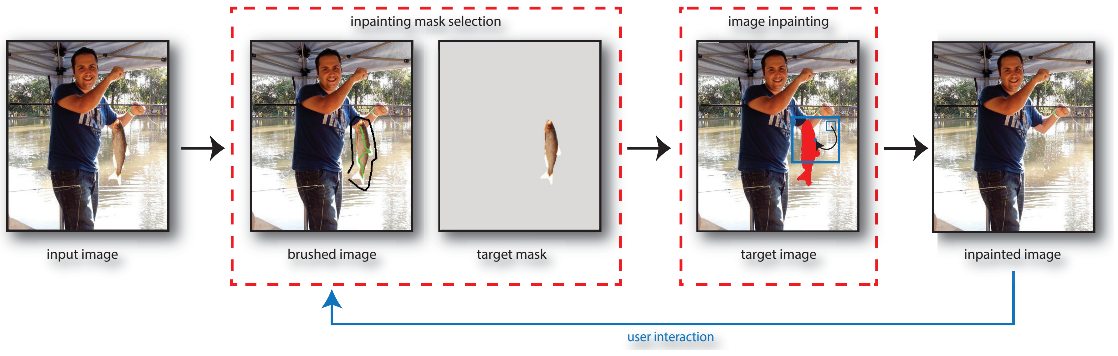


Fig. 28. Pipeline of our interactive inpainting framework.

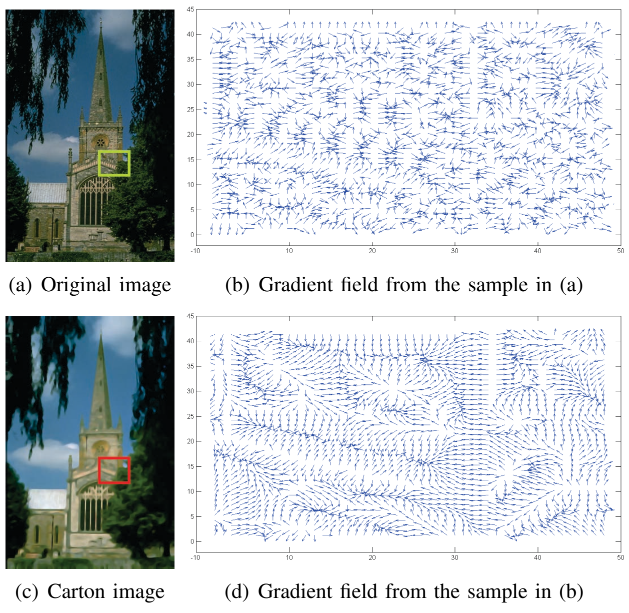


Fig. 29. Representation of the gradient field for an illustrative image.

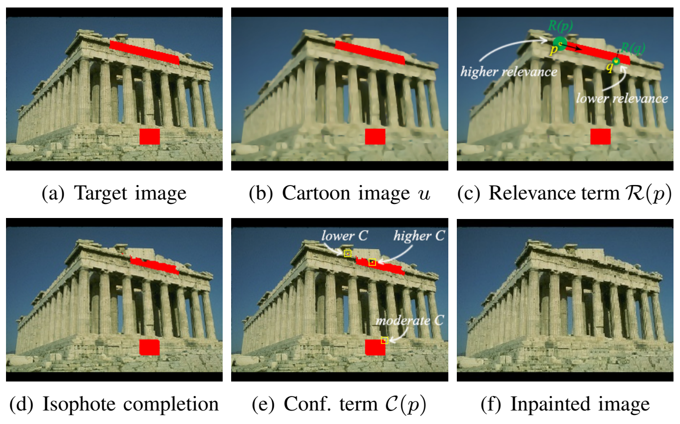


Fig. 30. Illustration of the priority filling order mechanism (Equation (26)).

The use of the cartoon image u allows us to embed image isophotes as well as image structures into the mechanism that computes the filling order of the damaged pixels. This mechanism is computed in terms of a new priority measure:

$$\mathcal{P}(p) = \mathcal{R}(p) \cdot \mathcal{C}(p), \quad p \in \partial\Omega, \tag{26}$$

where $\mathcal{R}(p)$ and $\mathcal{C}(p)$ represent the *Relevance* and *Biased Confidence* terms:

$$\mathcal{R}(p) = |\nabla(\Delta u_p) \cdot \overrightarrow{d}_p|, \qquad \overrightarrow{d_p} = \frac{\nabla^{\perp} u_p}{|\nabla^{\perp} u_p|}, \tag{27}$$

$$\mathcal{C}(p) = \left(\frac{\sum\limits_{q \in H_m(p) \cap (D-\Omega)} C(q)^{\frac{1}{k}}}{|H_m(p)|} \right)^k, \tag{28}$$

with $|H_m(p)|$ denoting the size of a squared block $m \times m$ centered at pixel p, and $k > 0$ is the bias parameter. Relevance term $\mathcal{R}$ computes the isophote direction from $\partial\Omega$. In fact, Expression (27) is similar to the transport equation proposed in [96], since it takes into consideration the directional derivative of a nontextured image for isophote propagation: $\mathcal{R}(p) = \left|\frac{\partial(\Delta u_p)}{\partial \overrightarrow{d_p}}\right|$. Biased confidence term (28) allows us to balance the filling order mechanism according to k parameter. For an illustration of $\mathcal{R}$ and $\mathcal{C}$, see Figure 30.

Block-based pixel replication. In this stage we are able to allocate the most suitable patch of pixels from the sampling region $\Lambda\Omega_p$ to the neighborhood of $p \in \partial\Omega$. Given a pixel $p \in \partial\Omega$, we define the region $H_L(p)$ from $\Lambda\Omega_p$ (see Fig.31(a)). Our algorithm then makes use of a cartoon-based metric to compare the fixed patch $H_n(p)$ with all candidate patches $H_n(q)$ inside $\Lambda\Omega_p$. More precisely, the optimal patch $H_n(\widehat{q})$ is the one which minimizes the distance between $H_n(p)$ and $H_n(q)$ w.r.t. a given metric. A smaller patch $H_m(\widehat{q})$ is then selected from $H_n(\widehat{q})$ and its valid pixels $H(\widehat{q})$ are placed in corresponding pixels in the neighborhood of p, namely $H(p)$ (see Fig.31(b)-(c)).

Let $\mathbf{p} = (f_{p_1}, f_{p_2}, ..., f_{p_l})$, $\mathbf{q} = (f_{q_1}, f_{q_2}, ..., f_{q_l})$ be the column vectors in $\mathbb{R}^l$, $l < n^2$, containing the intensities of the given pixels on $H_n(p)$ and in corresponding positions on $H_n(q)$. Aiming at measuring the distance between $H_n(p)$ (target) and $H_n(q)$ (candidate) blocks, we propose the following metric:

$$d(\mathbf{p},\mathbf{q}) = \frac{||\mathbf{p}-\mathbf{q}||_{\Delta U}}{\sqrt{||\mathbf{p}||^2_{\Delta U} + ||\mathbf{q}||^2_{\Delta U}}}, ||\mathbf{p}||_{\Delta U} := \sqrt{\mathbf{p}^T \Delta U \mathbf{p}}, \tag{29}$$

with ΔU being a diagonal matrix defined by the Laplacian of cartoon u: $\Delta U_{ii} = \Delta u_{p_i}, p_i \in H_n(p) \cap \Lambda\Omega_p$. Metric

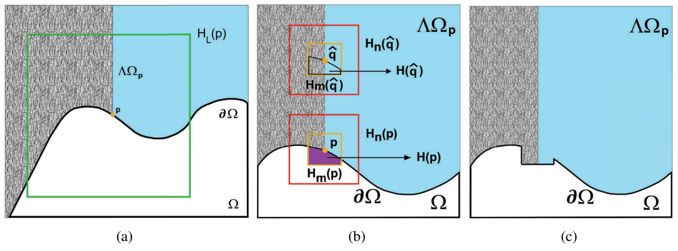


Fig. 31. Illustrative sketch of the dynamic sampling and the completion process. (a) $\Lambda\Omega_p$ (gray and blue parts) is the region inside $H_L(p)$ (green square) which provides candidate pixels. (b) Comparison between content of patches $H_n(p)$ and $H_n(\hat{q})$ (optimal patch) and (c) result after copying the information of interest.

(29) assigns higher weights for pixels located on the edges of the Laplacian operator of u. The weights of the valid pixels in Equation (29) are defined from the Laplacian of u, Δu, and then embedded into the distance computation of the pixel blocks.

2) Experimental Results and Comparisons: In this section we present a comparative study against state-of-the-art inpainting methods when taking the inpainting domain as input data. Best parameters were tuned according to original papers as well as the author's implementation available at their personal websites for the following methodologies: *Texture synthesis* [97], *Cartoon/texture decomposition-based inpainting* [98], *Exemplar-based inpainting* [99], *Optimization-based inpainting* [100] and *Sparsity-based inpainting* [101], [102], [103], [14], [104].

Comparison with Pure Texture Synthesis. Figure 32(a) reproduces a synthetic image made up of six different groups of textures, in which the inpainting region comprises a large amount of pixels. The challenge is to precisely reconstruct the boundary regions between distinct textures. It is clear in Fig. 32(b)-(c) that our framework outperforms the texture synthesis-based approach [97].

Comparison with Cartoon-Texture Inpainting. The experiment depicted in Figure 32(d)-(f) presents comparative results with the cartoon-texture inpainting algorithm [98]. Fig. 32(d) shows the input image and Fig. 32(e) depicts the resulting inpainting by [98]. Notice that some linear artifacts (inside the water area) were generated during the progress of the restoration. Our framework (Fig.32(f)), however, better recovers the natural aspect of the image.

Comparison with Exemplar-based Inpainting. Figure 32(g) shows a high-quality image. The result obtained by [99] (Fig. 32(h)) exhibits some artifacts creating inconsistent color tonalities. Our technique (Fig. 32(i)), however, captures fine details such as the layers of the "human eye".

Comparison with Optimization-based Inpainting. Object removal problem is investigated in this experiment. Fig. 32(j) presents a classical example of occluded image, where the goal is to accurately recover the gate parts hidden by the statue (Fig. 32(k)) while maintaining the regular aspect of the image. One can see that the inpainting method by [100] (Fig. 32(l)) results in a non-realistic image, as some parts of the image are overly smoothed. The proposed method, however, presents a more natural outcome (Fig. 32(m)), reconstructing the broken lines in a more satisfactory manner.

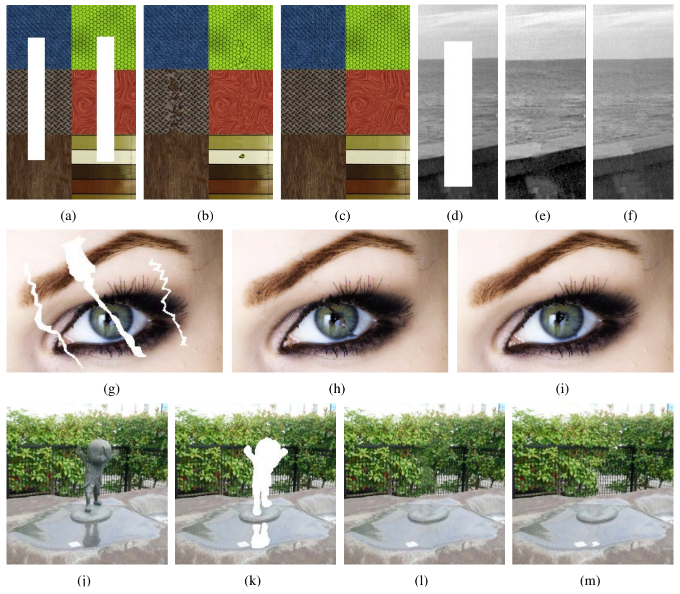


Fig. 32. Comparison with [97] (*pure texture synthesis*), [98] (*cartoon/texture inpainting*), [99] (*exemplar-based*) and [100] (*optimization-based*).

Comparison with Missing Block Completion Techniques. Finally, we finish this section performing both qualitative and quantitative evaluations against a variety of methods that address the problem of missing block completion. From left-to-right, we show in Figure 33 the input images and the inpainting results obtained by [101], [102], [103], [14], [104]. All examples are tricky to handle due to the predominance of structures and textures around the region to be filled. Notice that the algorithms [102] and [103] produce blurring effect on the images while the method [101] generates some artifacts in the outputs. The results reached by [14] as well as the method [104] are visually better than others but they still suffer from smoothing effect. In contrast, our technique leads to non-blurred completion and accurately recovers isophotes and pure texture regions, producing a pleasant and more realistic result.

For sake of quantitative comparison, PSNR (*Peak Signal-to-Noise Ratio*) between recovered and original images from Figure 33 have been considered (see Table III). Individual as well as the mean of PSNR values show that the proposed method quantitatively outperforms the others.

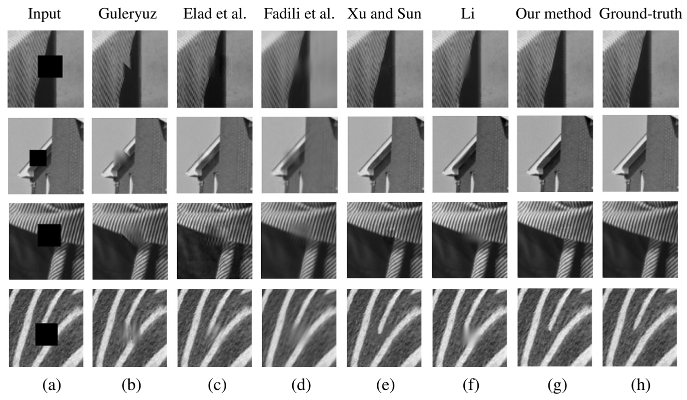


Fig. 33. Comparison with sparse representation-based inpainting methods.

TABLE III. QUANTITATIVE EVALUATION USING PSNR (IN DB) FOR ALL COMPARATIVE IMAGES FROM FIGURE 33.

Image	[102]	[101]	[103]	[14]	[104]	**Ours**
Tissue	20.41	22.43	22.16	23.53	22.21	25.02
Eaves	16.15	22.85	17.86	28.45	26.77	29.30
B.part	18.39	19.20	17.85	23.07	23.61	24.43
Fur	16.46	19.49	20.67	20.87	18.43	21.55
Mean	**17.85**	**20.99**	**19.64**	**23.98**	**22.75**	**25.08**

B. Interactive Photo Colorization via Laplacian Coordinates

Colorization is a computer-supervised process by which colors are imparted to grayscale images or to black-and-white films. It has been widely used in photo editing and scientific illustration, to modernize old films and to enhance the visual appear of an image. Traditionally, colorization is tedious, time consuming and requires artistic skills to precisely add appropriate colors to a grayscale image.

Aiming at making the colorization process simpler and less laborious, several computational systems have been proposed in the last decade, which can be roughly divided into two classes: *Example-based* [105], [106], [16], [17] and *Scribble-based* [107], [108], [109], [110], [111], [17]. Example-based methods accomplish the colorization process by matching the luminance of the grayscale image with the luminance of a reference color image used to drive the colorization. In scribble-based methods, the user guides the colorization by defining colored strokes onto the grayscale image. Due to the flexibility to operate arbitrary colorizations and the non-requirement for a reference image, scribble-based strategy has performed better than the example-based one in recent years [112]. This trend has been observed especially due to the simplicity of scribble-based methods which basically relies on an interactive interface in order to work.

The classical work by [107] is a good representative of scribble-based approach. Levin's method aims at optimizing the color of all image pixels using the scribbles as constraints. Although it shows good results for various types of images, Levin's method tends to propagate colors beyond the texture boundaries, thus resulting in unpleasant colorizations. The technique proposed by [108] employs adaptive edge detection so as to prevent colors from going beyond region boundaries. Further improvements have been proposed by [109], who present a faster scribble-based color optimization technique that relies on chrominance blending to perform the colorization. [110] and [111] employ texture continuity to colorize manga-cartoons and natural images, respectively. Despite good results, most existing scribble-based approach require intensive user involvement, especially when the image contains complex structures or has different texture patterns, which can demand lots of scribbles until acceptable outcomes are reached. In a more recent work, [17] has introduced an innovative user-based interface namely *ProjColor* that relies on a simple drag-and-drop manipulation of the badly colorized pixels using multidimensional projection as an recursive tool.

In this work we propose a new approach for colorizing grayscale images that relies on a scribble-based interface to replace the excessive provision of user strokes typically employed by existing scribble-driven methods. Moreover, the proposed approach holds the good segmentation properties

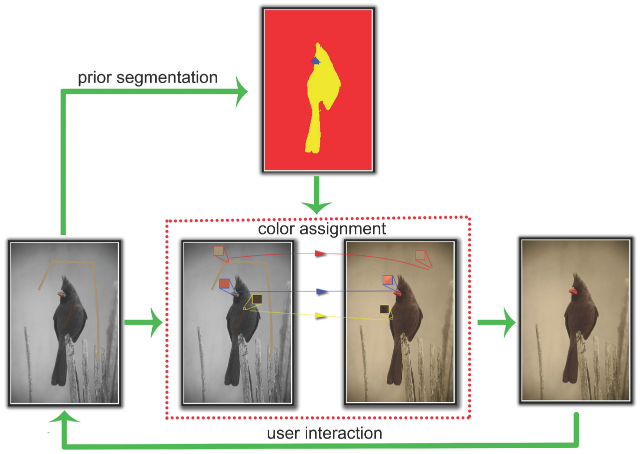


Fig. 34. Pipeline of our colorization framework.

derived from Laplacian Coordinates [3]. Since Laplacian Coordinates is used to precompute a prior segmentation of the input monochromatic image, our framework leads to pleasant results and requires just a few user interventions to colorize the image. As we shall show, by one-shot stroking the image, the user can colorize complex textured areas quite easily, preserving region edges and preventing the addition of new scribbles. In summary, the main contributions of this work are:

1) A novel interactive image colorization technique that combines the accuracy and the effectiveness of the Laplacian Coordinates approach with a fast color matching scheme to colorize images.
2) An efficient system that allows for recursively colorizing the image by reintroducing new seeds to reach more pleasant results.
3) The new method is easy-to-implement and it requires just a small amount of user intervention to quickly reach a good result.

1) Pipeline Overview: As illustrated in Figure 34, the proposed colorization pipeline comprises three main steps, namely, *User-driven Prior Segmentation*, *Color Assignment* and *Interactive Recolorization*. First, color scribbles given by the user are taken as constraints to the Laplacian Coordinates approach aiming at generating a prior segmentation. The partitioning obtained is then used to promote color transfer between input scribbles and image segments. Badly colored regions can be updated by interacting with the Laplacian Coordinates segmentation interface. Details of each stage of the pipeline are presented in the following sections.

User-driven Prior Segmentation. In this stage we use Laplacian Coordinates to assist the colorization process by fragmenting the image into multiple regions. Color labels are manually chosen by the user and freely spread inside representative image regions. Those labels are then designed to condition the linear system of equations formed by the Laplacian Coordinates approach. After solving the constrained linear system, the partitions obtained are used to support the next stage, *Color Assignment*. Figure 34 (up step) illustrates the procedure described above.

Color Assignment. The color assignment modulus is responsible for propagating the colors chosen by the user to the partitions generated by the LC segmentation. The propagation mechanism is accomplished as follows: given the set of color labels provided during the segmentation stage, we first convert those labels to $L\alpha\beta$ color system. Next, coordinates α and β are then copied to the uncolored pixels in the specified image segment. Similar procedure is performed until colorizing the remaining partitions (see the middle step in Figure 34).

Interactive Manipulation of the Colorization Result. One of the main contributions of the proposed framework is to exploit the flexibility provided by the LC scheme to interactively modify the colorization. Laplacian Coordinates enables an interactive tool that allows for repartitioning data by inserting new seeded pixels. In fact, if the result is not satisfactory, the user can select badly colored pixels, turning them into a different color label that can be reintroduced into the LC linear system to partition the image and, thereby, improve the resulting colorization. Figure 35 illustrates the need for user intervention. Notice that a few pixels close to the edge between the panther and the background grass were not colored properly as highlighted in Fig. 35(b). User can then provide an additional color scribble to the region with badly colorized pixels creating new constraints for the Laplacian Coordinates and, thus, generating a better result as shown in Fig. 35(c).

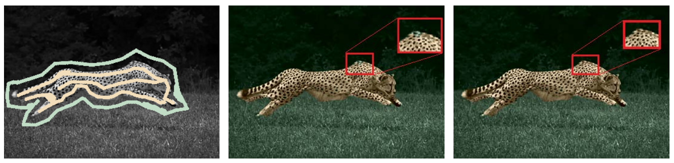

(a) Initial scribbles (b) The resulting colorization (c) Colorization after user interaction

Fig. 35. The use of our framework when allowing for user intervention.

2) Results and Comparisons: In order to confirm the effectiveness of our methodology, we provide comparisons against the well-established scribble techniques [107], [109], [112].

Figure 36 illustrates the capability of our method to perform colorization from a few user intervention. The seeding interface provided by Laplacian Coordinates approach is simpler than the traditional scribble-based employed by [107], as the user does not need to spread an excessive number of scribbles in the whole image to reach a reasonable result. In addition to its simplicity and ease of use, the colorization mechanism based on the Laplacian Coordinates produces pleasant results.

The experiment presented in Figure 37 establishes comparisons between the proposed technique and scribble-based methods [107], [109], [112]. Colorizations produced by [107] and [109] smoothed the images considerably almost all cases while the outcomes obtained by [112] and our technique have produced more refined results. By reintroducing just a small amount of seeds in the reference images in Figs.(g)-(o)-(w), one can see that our approach is quite flexible in capturing intrinsic details of the image such as pieces surrounded by image segments, a characteristic not present in the algorithm [112]. For sake of quantitative comparison, PSNR between the colorizations and original images in Fig. 37 were computed and summarized in Table IV. Notice from the average PSNR in the last row of Table IV that our approach outperforms others.

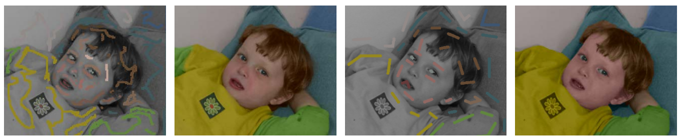

(a) Scribbles used by Levin's method (b) Levin's result from (a) (c) Scribbles used by our method (d) Our result from (c)

Fig. 36. Comparison between [107] and our framework.

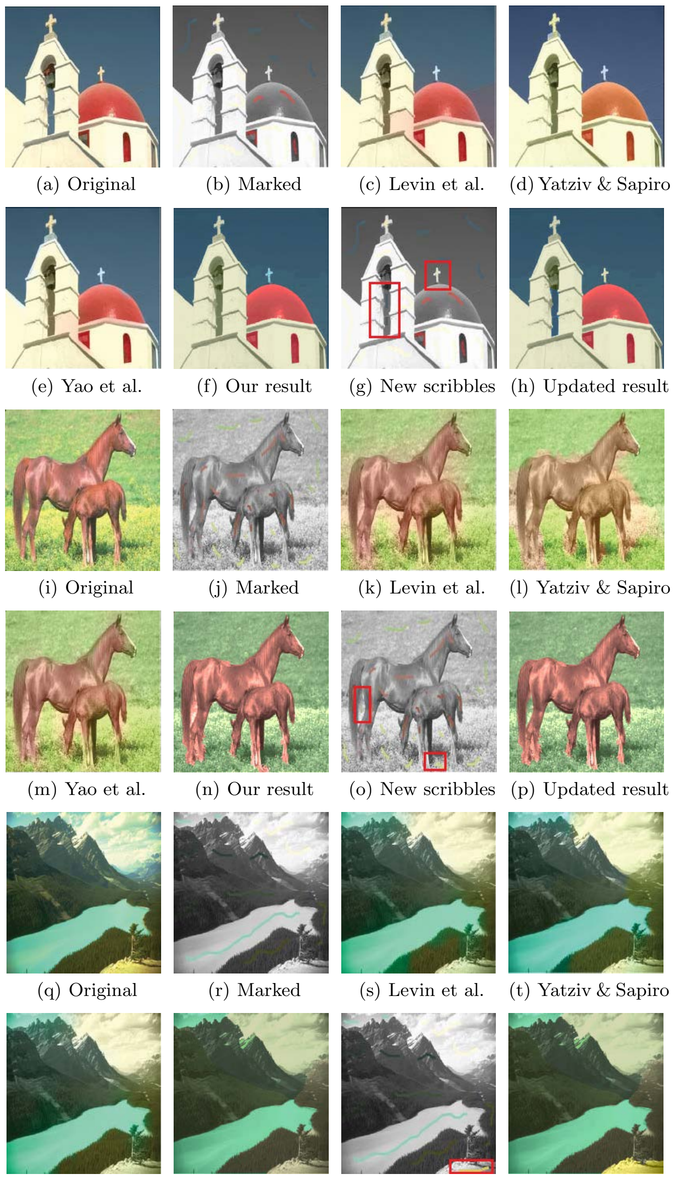

(a) Original (b) Marked (c) Levin et al. (d) Yatziv & Sapiro
(e) Yao et al. (f) Our result (g) New scribbles (h) Updated result
(i) Original (j) Marked (k) Levin et al. (l) Yatziv & Sapiro
(m) Yao et al. (n) Our result (o) New scribbles (p) Updated result
(q) Original (r) Marked (s) Levin et al. (t) Yatziv & Sapiro
(u) Yao et al. (v) Our result (w) New scribbles (x) Updated result

Fig. 37. Comparison with [107], [109], [112] and our method.

TABLE IV. PSNR COMPUTATION FOR IMAGES FROM FIG. 37.

Image	[107]	[109]	[112]	**Ours**	**Updated**
Church	35.06	34.38	35.54	37.00	37.43
Horse	29.67	28.62	30.52	31.84	31.87
River	31.82	31.48	32.18	32.95	33.44
Average	**32.18**	**31.49**	**32.75**	**33.93**	**34.25**

V. Conclusion

This thesis has addressed the fundamental problem of image segmentation from different perspectives. More specifically, we focus our attention on studying spectral methods in the context of graph-based image representation. We have also presented a novel Laplacian-based energy minimization for image segmentation and two new algorithms that depend on segmentation to perform image inpainting and colorization.

The combination cartoon-texture decomposition and spectral cut turned out to be a quite efficient methodology for image segmentation. Moreover, the proposed inner product-based weight assignment mechanism has produced more accurate results than the exponential weighting function used by other spectral clustering methods. The qualitative and quantitative analysis clearly showed the effectiveness of the proposed technique, surpassing, in terms of accuracy and smoothness, representative spectral cut-based methods. Furthermore, the flexibility as to user intervention is an important trait of our method, which enables the user to fix the segmentation locally.

The Laplacian Coordinates introduced in Section III is a novel seed-based image segmentation technique which has several advantages when compared with other methods. Besides its simple mathematical formulation, Laplacian Coordinates is easy to implement, guarantees a unique solution, and outperforms existing methods with respect to well established quantitative measures popularly used in the context of image segmentation. Laplacian Coordinates also holds high accuracy in terms of image boundary fitting capability, rendering it an interesting and compelling seed-based segmentation method.

The new inpainting methodology presented in the last Section operates by copy-to-paste blocks to recover real and synthetic images containing a large variety of textures and structure parts. The combination between interactive image segmentation, image decomposition and transport equation were rearranged so as to provide a robust interactive interface that allows for user involvement while managing the hierarchy of the restoration task. Images of different complexity levels were evaluated against state-of-the-art methods with the purpose of assessing the efficiency of the proposed approach. Our segmentation apparatus has also been successfully used as a basic tool for the photo colorization application. In fact, besides enabling a local modification of badly colored regions, the proposed method turned out to be robust when dealing with real-world images. In summary, flexibility and effectiveness render the proposed method one of the most attractive alternatives in the context of image colorization.

Acknowledgment

Special thanks for Prof. Luis Gustavo Nonato (advisor, ICMC-USP), Prof. Gabriel Taubin (co-adivor, Brown University) and Prof. Alex J. Cuadros Vargas (UCSP) for their valuable comments and ideas to improve the quality of this PhD work. This research has been funded by FAPESP (grants #09/17801-0 and #13/07375-0), CNPq and NSF-USA.

References

[1] F. Chung, *Spectral Graph Theory*. CBMS, Reg. Conf. S. in Mathematics, Ameciran Mathematical Society, 1997.

[2] W. Casaca, A. Paiva, E. Gomez-Nieto, P. Joia, and L. G. Nonato, "Spectral image segmentation using image decomposition and inner product-based metric," *Journal of Mathematical Imaging and Vision*, vol. 45, no. 3, pp. 227–238, 2013.

[3] W. Casaca, L. G. Nonato, and G. Taubin, "Laplacian coordinates for seeded image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 384–391.

[4] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient n-d image segmentation," *International Journal of Computer Vision*, vol. 70, no. 7, pp. 109–131, 2006.

[5] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1768–1783, 2006.

[6] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Minimum spanning forests and the drop of water principle," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 8, pp. 1362–1374, 2009.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 888–905, 2000.

[8] S. Maji, N. Vishnhoi, and J. Malik, "Biased normalized cuts," in *IEEE Conference on Computer Vision and Patter Recognition (CVPR)*, 2011, pp. 2057–2064.

[9] C. J. Taylor, "Towards fast and accurate segmentation," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 1–8.

[10] W. Casaca, A. Paiva, and L. G. Nonato, "Spectral segmentation using cartoon-texture decomposition and inner product-based metric," in *24th Conference on Graphics, Patterns and Images (SIBGRAPI), Best paper award*. IEEE Computer Society, 2011, pp. 266–273.

[11] F. Yi and I. Moon, "Image segmentation: A survey of graph-cut methods," in *International Conference on Systems and Informatics (ICSAI)*, 2012, pp. 1936–1941.

[12] B. Peng, L. Zhang, and D. Zhang, "A survey of graph theoretical approaches to image segmentation," *Pattern Recognition*, vol. 46, no. 3, pp. 1020–1038, 2013.

[13] W. Casaca, "Graph laplacian for spectral clustering and seeded image segmentation," 161 pgs, PhD Thesis, University of São Paulo, São Carlos, Brazil, 2014.

[14] Z. Xu and J. Sun, "Image inpainting by patch propagation using patch sparsity," *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1153–1165, 2010.

[15] W. Casaca, M. P. Almeida, M. Boaventura, and L. G. Nonato, "Combining anisotropic diffuison, transport equation and texture synthesis for inpainting textured images," *Pattern Recognition Letters (PRL)*, vol. 36, pp. 36–45, 2014.

[16] R. Irony, D. Cohen-Or, and D. Lischinski, "Colorization by example," in *Proc. of the Eurographics Symp. on Rendering*, 2005, pp. 201–210.

[17] W. Casaca, E. Gomez-Nieto, C. de O.L. Ferreira, G. Tavares, P. Pagliosa, F. Paulovich, L. G. Nonato, and A. Paiva, "Colorization by multidimensional projection," in *25th Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2012, pp. 32–38.

[18] W. Casaca, M. Boaventura, and M. P. de Almeida, "Classifying texture and free-textured content from images using nonlinear anisotropic diffusion," in *5th Conf. on Mathematical Analysis*, 2011, pp. 210–214.

[19] W. Casaca, M. P. Almeida, and M. Boaventura, "Denoising textured images via regularized anisotropic diffusion," in *An Introductory Guide to Image and Video Processing*, Z. N. Akshaya Mishra and Z. Shahid, Eds. Brisbane, Australia: Iconcept Press Ltd., 2013, pp. 48–71.

[20] W. Casaca, M. Colnago, and L. G. Nonato, "Interactive image colorization using laplacian coordinates," in *16th Int. Conf. on Computer Analysis of Images and Patterns (CAIP)*, 2015, pp. 1–12.

[21] W. Casaca, D. Motta, G. Taubin, and L. G. Nonato, "A user-friendly interactive image inpainting framework using laplacian coordinates," in *IEEE International Conference on Image Processing (ICIP) (accepted for publication)*. IEEE Computer Society, 2015, pp. 1–5.

[22] P. Joia, E. Gomez-Nieto, J. Batista Neto, W. Casaca, G. Botelho, A. Paiva, and L. Gustavo Nonato, "Class-specific metrics for multidimensional data projection applied to cbir," *The Visual Computer*, vol. 28, no. 10, pp. 1027–1037, 2012.

[23] E. Gomez-Nieto, W. Casaca, L. G. Nonato, and G. Taubin, "Mixed integer optimization for layout arrangement," in *26th Conference on*

Graphics, Patterns and Images (SIBGRAPI), Best paper award in graphics and visualization, 2013, pp. 115–122.

[24] E. Gomez-Nieto, F. S. Roman, P. Pagliosa, W. Casaca, E. S. Helou, M. C. F. de Oliveira, and L. G. Nonato, "Similarity preserving snippet-based visualization of web search results," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 457–470, 2014.

[25] E. Gomez-Nieto, W. Casaca, I. Hartmman, and L. G. Nonato, "Understanding large legal datasets through visual analytics," in *28th Conference on Graphics, Patterns and Images (SIBGRAPI, WVis)*. IEEE Computer Society, 2015, pp. 1–4.

[26] B. Mohar, "Some applications of laplacian eigenvalues of graphs," *Graph Symetric: Algebraic Methods and Applications*, vol. 497, pp. 225–275, 1997.

[27] D. A. Spielman, "Spectral graph theory and its applications," in *Proc. of the 48th Annual IEEE Symposium on Foundations of Computer Science*, 2007, pp. 29–38.

[28] W. Tao, H. Jin, and Y. Zhang, "Color image segmentation based on mean shift and normalized cuts," *IEEE Systems, Man, and Cybernetics*, vol. 37, pp. 1382–1389, 2007.

[29] M. Carvalho, A. Costa, A. Ferreira, and R. C. Junior, "Image segmentation using component tree and normalized cut," in *23th Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2010, pp. 317–322.

[30] M. Carvalho, A. Ferreira, and A. Costa, "Image segmentation using quadtree-based similarity graph and normalized cut," *Lectures Notes in Computer Science*, vol. 6419, pp. 329–337, 2010.

[31] X. Ma, W. Wan, and J. Yao, "Texture image segmentation on improved watershed and multiway spectral clustering," in *IEEE Int. Conf. on Audio, Language and Image Processing*, 2008, pp. 1693–1697.

[32] A. Sáez, C. Serrano, and B. Acha, "Normalized cut optimization based on color perception findings: A comparative study," *Machine Vision and Applications*, pp. 1–11, 2014.

[33] S. Yu, "Segmentation using multiscale cues," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. I–247–I–254.

[34] T. Cour, F. Bénézit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 1124–1131.

[35] W. Cai and A. C. S. Chung, "Shape-based image segmentation using normalized cuts," in *Proc. of IEEE Int. Conference on Image Processing (ICIP)*, 2006, pp. 1101–1104.

[36] F. Sun and J.-P. He, "A normalized cuts based image segmentation method," in *International Conference on Information and Computing Science (ICIC)*, 2009, pp. 333–336.

[37] M. Maire and S. X. Yu, "Progressive multigrid eigensolvers for multiscale spectral segmentation," *IEEE International Conference on Computer Vision (ICCV)*, vol. 0, pp. 2184–2191, 2013.

[38] L. Grady, "Multilabel random walker image segmentation using prior models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 763–770.

[39] L. Grady and A. K. Sinop, "Fast approximate random walker segmentation using eigenvector precomputation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[40] S. Andrews, G. Hamarneh, and A. Saad, "Fast random walker with priors using precomputation for interactive medical image segmentation," in *Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2009, pp. 9–16.

[41] D. Tolliver and G. L. Miller, "Graph partitioning by spectral rounding: Applications in image segmentation and clustering," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2006, pp. 1053–1060.

[42] T. H. Kim, K. M. Lee, and S. U. Lee, "Learning full pairwise affinities for spectral segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1690–1703, 2013.

[43] I. Koutis, G. L. Miller, and D. Tolliver, "Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing," in *International Symposium on Visual Computing (ISVC)*, 2009, pp. 1067–1078.

[44] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *IEEE International Conference on Computer Vision (ICCV)*, 2001, pp. 416–423.

[45] L. Demanet and L. Ying, "Wave atoms and sparsity of oscillatory patterns," *Applied and Computational Harmonic Analysis*, vol. 23, pp. 368–387, 2007.

[46] ——, "Curvelets and wave atoms for mirror-extended images," in *Proc. of SPIE Wavelets XII*, vol. 6701, 2007, p. 67010J.

[47] L. Vese and S. Osher, "Modeling textures with total variation minimization and oscillating patters in image processing," *Journal of Scientific Computing*, vol. 19, pp. 553–572, 2003.

[48] Y. Meyer, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations*. AMS, 2001.

[49] L. Vese and S. Osher, "Color texture modeling and color image decomposition in a variational-pde approach," in *Proc. of the 8th Int. Symposium on Symbolic and Numeric Algorithms for Scientific Comp.*, 2006, pp. 103–110.

[50] Y. Shuai, A. Masahide, T. Akira, and K. Masayuki, "High accuracy bicubic interpolation using image local features," *Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 8, pp. 1611–1615, 2007.

[51] Y. Zhang, D. Zhao, J. Zhang, R. Xiong, and W. Gao, "Interpolation-dependent image downsampling." *IEEE Transactions on Image Processing*, vol. 20, no. 11, pp. 3291–3296, 2011.

[52] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, 1990.

[53] W. Casaca and M. Boaventura, "A regularized nonlinear diffusion approach for texture image denoising," in *22th Conference on Computer Graphics and Image Processing (SIBGRAPI)*. IEEE Computer Society, 2009, pp. 164–171.

[54] ——, "A decomposition and noise removal method combining diffusion equation and wave atoms for textured images," *Mathematical Problems in Engineering*, vol. 2010, pp. 1–21, 2010.

[55] F. J. Estrada and A. D. Jepson, "Benchmarking image segmentation algorithms," *International Journal of Comput. Vision*, vol. 85, no. 2, pp. 167–181, 2009.

[56] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011.

[57] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[58] S. Vicente, V. Kolmogorov, and C. Rother, "Graph cut based image segmentation with connectivity priors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[59] P. Miranda and A. Falcão, "Links between image segmentation based on optimum-path forest and minimum cut in graph," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 128–142, 2009.

[60] K. C. Ciesielski, J. K. Udupa, A. X. Falcão, and P. Miranda, "Fuzzy connectedness image segmentation in graph cut formulation: A linear-time algorithm and a comparative analysis," *Journal of Mathematical Imaging and Vision*, vol. 44, no. 3, pp. 375–398, 2012.

[61] L. Grady, "Targeted image segmentation using graph methods," in *Image Processing and Analysis with Graphs*, O. Lézoray and L. Grady, Eds. CRC Press, 2012, pp. 111–135.

[62] B. Wang and Z. Tu, "Affinity learning via self-diffusion for image segmentation and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 2312–2319.

[63] A. Angelova and S. Zhu, "Efficient object detection and segmentation for fine-grained recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 811–818.

[64] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest," in *IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 731–738.

[65] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watershed: A unifying graph-based optimization framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 7, pp. 1384–1399, 2011.

[66] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images," in *IEEE International Conference on Computer Vision*, 2001, pp. 105–112.

[67] D. Hower, V. Singh, and S. C. Johnson, "Label set perturbation for mrf based neuroimaging segmentation," in *IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 849–856.

[68] A. K. Sinop and L. Grady, "A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm," in *IEEE International Conference on Computer Vision*, 2007, pp. 1–8.

[69] F. Monteiro and A. Campilho, "Watershed framework to region-based image segmentation," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2008, pp. 1–4.

[70] J. Coustry, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Thinnings, shortest path forests, and topological watersheds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 925–939, 2010.

[71] A. Falcão, J. Stolfi, and R. Lotufo, "The image foresting transform: Theory, algorithms, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 19–29, 2004.

[72] F. P. Bergo, A. X. Falcão, P. A. Miranda, and L. M. Rocha, "Automatic image segmentation by tree pruning," *J. Math. Imaging Vis.*, vol. 29, pp. 141–162, 2007.

[73] A. Criminisi, T. Sharp, and A. Blake, "Geos: Geodesic image segmentation," in *Proceddings of European Conference on Computer Vision: Part I (ECCV)*, 2008, pp. 99–112.

[74] L. M. Rocha, F. A. Cappabianco, and A. Falcão, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging System and Technology*, vol. 19, no. 2, pp. 50–68, 2009.

[75] X. Bai and G. Sapiro, "Geodesic matting: A framework for fast interactive image and video segmentation and matting," *International Journal of Computer Vision (IJCV)*, vol. 82, no. 2, pp. 113–132, 2009.

[76] N. T. N. Anh, J. Cai, J. Zhang, and J. Zheng, "Robust interactive image segmentation using convex active contours," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3734–3743, 2012.

[77] O. Sorkine, "Differential representations for mesh processing," *Computer Graphics Forum*, vol. 25, no. 4, pp. 789–807, 2006.

[78] K. Xu, H. Zhang, D. Cohen-Or, and Y. Xiong, "Dynamic harmonic fields for surface processing," *Computer and Graphics*, vol. 33, no. 3, pp. 391–398, 2009.

[79] K.-J. Yoon and I.-S. Kweon, "Locally adaptive support-weight approach for visual correspondence search," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 924–931.

[80] Z. Zhu, C. C. Loy, and S. Gong, "Constructing robust affinity graphs for spectral clustering," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 550–558.

[81] V. Jankovic, "Quadratic functions in several variables," *The Teaching of Mathematics*, vol. VIII, no. 1, pp. 53–60, 2005.

[82] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts," in *Proceedings of Seventh International Symposium on Mathematical Morphology*, 2007, pp. 301–312.

[83] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "Toward objective evaluation of image segmentation algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 929–944, 2007.

[84] M. Meilă, "Comparing clusterings: an axiomatic view," in *Proceedings of the 22nd international Conference on Machine Learning*, ser. ICML '05, 2005, pp. 577–584.

[85] M. Mignotte, "A label field fusion model with a variation of information estimator for image segmentation," *Information Fusion*, vol. 20, no. 1, pp. 7–20, 2014.

[86] M. Meilă, "Comparing clusterings: An information based distance," *Journal of Multivariate Analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[87] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "A measure for objective evaluation of image segmentation algorithms," in *Workshops of the IEEE Conf. on Computer Vision and Pattern Recognition*, ser. CVPR '05, 2005, pp. 34–39.

[88] M. Subramanyam, D. K. Nallaperumal, P. P., and D. S., "Analysis of image inpainting techniques with exemplar, poisson, successive elimination and 8 pixel neighborhood methods," *International Journal of Computer Applications*, vol. 9, no. 11, pp. 15–18, 2010.

[89] W. Casaca, "Restauração de imagens digitais com texturas utilizando técnicas de decomposição e equações diferenciais parciais (in portuguese)," 192 pgs, Master Dissertation, São Paulo State University (UNESP), IBILCE, São José do Rio Preto, SP, Brazil, 2010.

[90] T. Tamaki, H. Suzuki, and M. Yamamoto, "String-like occluding region extraction for background restoration," in *IEEE International Conference on Pattern Recognition (ICPR)*, 2006, pp. 615–618.

[91] T. Amano, "Correlation based image defect detection," in *IEEE Int. Conf. on Pattern Recognition*, 2006, pp. 163–166.

[92] M. G. Padalkar, M. A. Zaveri, and M. V. Joshi, "Svd based automatic detection of target regions for image inpainting," in *13th Asian Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 7729, 2013, pp. 61–71.

[93] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image completion with structure propagation," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 861–868, 2005.

[94] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graphics*, vol. 28, no. 3, pp. 24:1–24:11, 2009.

[95] C. A. Z. Barcelos, M. Boaventura, and E. C. S. Jr., "A well balanced flow equation for noise removal and edge detection," *IEEE Transactions on Image Processing*, vol. 12, no. 7, pp. 751–763, 2003.

[96] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Annual Conf. on Comp. Graphics*, 2000, pp. 217–226.

[97] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *IEEE International Conference on Computer Vision (ICCV)*, 1999, pp. 1033–1038.

[98] M. Bertalmío, L. A. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," *IEEE Transactions on Image Processing*, vol. 12, pp. 882–889, 2003.

[99] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on Image Processing*, vol. 13, pp. 1200–1212, 2004.

[100] N. Kawai, T. Sato, and N. Yokoya, "Image inpainting considering brightness change and spatial locality of textures and its evaluation," in *Proceedings of the 3rd Pacific Rim Symp. on Adv. in Img. and Video Technology*, ser. PSIVT '09, 2009, pp. 271–282.

[101] M. Elad, J.-L. Starck, P. Querre, and D. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis," *App. Comp. Harmonic Anal.*, vol. 19, pp. 340–358, 2005.

[102] O. G. Guleryuz, "Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising-part ii: adaptive algorithms," *IEEE Transactions on Image Processing*, vol. 15, no. 3, pp. 555–571, 2006.

[103] M. Fadili, J.-L. Starck, and F. Murtagh, "Inpainting and zooming using sparse representations," *The Comp. J.*, vol. 52, no. 1, pp. 64–79, 2009.

[104] X. Li, "Image recovery via hybrid sparse representations: a deterministic annealing approach," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 953–962, 2011.

[105] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. Salesin, "Image analogies," in *ACM Trans. Graphics*, 2001, pp. 327–340.

[106] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 277–280, 2002.

[107] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 689–694, 2004.

[108] Y.-C. Huang, Y.-S. Tung, J.-C. Chen, S.-W. Wang, and J.-L. Wu, "An adaptive edge detection based colorization algorithm and its applications," in *Proc. of the 13th ACM Intl. Conf. on Multimedia*, 2005, pp. 351–354.

[109] L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," *IEEE Transactions on Image Processing*, vol. 15, no. 5, pp. 1120–1129, 2006.

[110] Y. Qu, T.-T. Wong, and P.-A. Heng, "Manga colorization," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1214–1220, 2006.

[111] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Natural image colorization," in *Proc. of the Eurographics Symposium on Rendering*, 2007, pp. 309–320.

[112] C. Yao, X. Yang, L. Chan, and Y. Xu, "Image colorization using bayesian nonlocal inference," *Journal of Electronic Imaing*, vol. 20, no. 2, pp. 023 008–023 008–6, 2011.

Breve Paraguay

Diego Pinto[1]
[1]Facultad Politécnica. Asunción Paraguay
email: *dpinto@pol.una.py*

Schedule: Tue 20th@18:45, **Room:** C

Se presenta brevemente los criterios para evaluación de calidad de las Carreras de Ingeniería en el área Informática la cual fue desarrollado por la Agencia Nacional de Evaluación y Acreditación de la Educación Superior (ANEAES) del Paraguay en cooperación con las distintas Universidades Nacionales y Privadas del Paraguay. Estos criterios se basaron en documentos como el MEXA, ABET, Career Space, CONEAU, Proyecto Alfa Tuning América, ACM e IEEE. Los criterios fueron desarrollados en los años 2013 a 2014 y actualmente se encuentran las distintas Universidades en fase de autoevaluación.

USP Brasil: Redes de Colaboração na Área de Computação: perspectivas e ações

Adenilso Simao[1]
[1]University of Sao Paulo. São Carlos-SP Brazil
email: *adenilso@icmc.usp.br*

Schedule: Wed 21st@10:15, **Room:** D

Redes de Colaboração constituem um mecanismo efetivo e essencial para o desenvolvimento de soluções efetivos e de impacto para a sociedade. Nesta palestra promovida pelo ICMC-USP, por meio das Comissões de Relações Internacionais, de Pesquisa e de Pós-graduação, pretende-se motivar redes de colaboração em pesquisa, eventualmente explorando-se também a colaboração academia-indústria, na América Latina. Entende-se que as articulações poderiam ocorrer no âmbito do CLEI - centro e evento, que há décadas desenvolve atividades relevantes para a comunidade latino-americana. Essa colaboração deve ocorrer com o envolvimento de todos os agentes do sistema de ensino e pesquisa dos diversos países. Como ponto central será apresentada uma chamada para realização de pós-doutoramento no ICMC-USP, visando a promover e fortalecer a cooperação sul-sul. Outras iniciativas e propostas serão apresentadas e discutidas: programa de doutorado sanduiche; programa de professor visitante bilateral; dupla titulação entre instituições. Tem-se como objetivo de fundo promover a mobilidade de pesquisadores na América Latina. Um facilitador seria viabilizar um exame para ingresso em pós-graduação na américa latina coordenado pelo centro CLEI, o que facilitaria a mobilidade dos novos talentos para programas de mestrado e de doutorado.

UChile Chile: Programa de Doctorado en Ciencias de la Computación en la Universidad de Chile

Gonzalo Navarro[1]
[1]Universidad de Chile. Santiago de Chile Chile
email: *gnavarro@dcc.uchile.cl*

Schedule: Wed 21st@10:45, **Room:** D

Describiré el programa de doctorado que ofrecemos en el Departamento de Ciencias de la Computación de la Universidad de Chile, y los mecanismos de becas que existen para quienes postulan desde el extranjero.

Learning Dynamic Action Units for Three-dimensional Facial Expression Recognition

Pablo Arbelaez[1]
[1]Universidad de Los Andes. Bogotá Colombia
email: *pa.arbelaez@uniandes.edu.co*

Schedule: Thu 22st@10:15, **Room:** C

Automated understanding of facial expressions is a fundamental step towards high-level human-computer interaction. The goal of this proposal is to develop a solution to this problem by taking advantage of the color, depth and temporal information provided by an RGB-D video feed. We plan to model human facial expressions through the analysis of temporal variations in the pattern of activations of their natural constituents, three-dimensional Action Units. As starting point for algorithm development, we propose to build on our prior experience developing convolutional neural network architectures for fine-grained localization, RGB-D scene understanding and video analysis.

Interconnected Dual Biosensor for Type II Diabetes Mellitus

Mathieu Hautefeuille[1]
[1]Universidad Nacional Autónoma de México. Mexico City Mexico
email: *mathieu_h@ciencias.unam.mx*

Schedule: Thu 22st@11:15, **Room:** C

The main objective of the present project is to develop a biosensor for diagnostic and monitoring of type 2 diabetes mellitus. This platform will integrate transducers and electronics to measure glucose and insulin levels in real time, providing data universally accessible and useful for further analysis by physicians and clinicians, as well as for statisticians for a broader screening of this disease in Mexico and in certain local populations.

Urban Coordination of Autonomous Vehicles

Jorge Zapotécatl[1]
[1]Universidad Nacional Autónoma de México. Mexico City Mexico
email: *jzapotecatl@gmail.com*

Schedule: Thu 22st@11:45, **Room:** C

The massive deployment of autonomous vehicles in the coming years offers a great potential for improving urban mobility. An open question is how to coordinate selfdriving cars at intersections, which we will address in this project. Since traffic conditions change constantly, a promising alternative to develop adaptive coordination mechanisms is to use self-organization. We have already developed self-organizing algorithms for traffic light coordination with significant performance improvements over traditional coordination methods. We will follow a similar path exploring how autonomous vehicles can self-organize at intersections to maximize efficiency and safety without the need of traffic lights.

List of Authors (50)