UNIVERSIDADE FEDERAL FLUMINENSE

RUBEN INTERIAN

Contributions to network optimization problems: Polarization, routing and covering

NITERÓI

2019

UNIVERSIDADE FEDERAL FLUMINENSE

RUBEN INTERIAN

Contributions to network optimization problems: Polarization, routing and covering

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Algoritmos e Otimização.

Orientador:
Celso da Cruz Carneiro Ribeiro

NITERÓI

2019

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

```
I61c    Interian kovaliova, Ruben
          Contributions to network optimization problems :
        Polarization, routing and covering / Ruben Interian kovaliova
        ; Celso da Cruz Carneiro Ribeiro, orientador. Niterói, 2019.
          198 p. : il.

          Tese (doutorado)-Universidade Federal Fluminense, Niterói,
        2019.

        DOI: http://dx.doi.org/10.22409/PGC.2019.d.07346481119

          1. Otimização (Computação). 2. Produção intelectual.I.
        Ribeiro, Celso da Cruz Carneiro, orientador. II. Universidade
        Federal Fluminense. Instituto de Computação. III. Título.

                                              CDD -
```

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

RUBEN INTERIAN

CONTRIBUTIONS TO NETWORK OPTIMIZATION PROBLEMS:
POLARIZATION, ROUTING AND COVERING

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Algoritmos e Otimização.

Apresentada em Setembro de 2019.

BANCA EXAMINADORA

[illegible]

Prof. Celso da Cruz Carneiro Ribeiro - Orientador, UFF

[illegible]

Prof. Isabel Cristina Mello Rossetti, UFF

[illegible]

Prof. Jayme Luiz Szwarcfiter, UFRJ

[illegible]

Prof. Leonardo Gresta Paulino Murta, UFF

[illegible]

Prof. Nelson Maculan Filho, UFRJ

Uéverton dos Santos Souza

Prof. Uéverton dos Santos Souza, UFF

[illegible]

Prof. Virgilio Augusto Fernandes Almeida, UFMG

Niterói

2019

À memória daqueles que caíram vítimas pelo simples fato de pensar de maneira diferente, incomodando algum ditador empenhado em construir um 'admirável mundo novo'.

Agradecimentos

Agradeço a Deus por me dar a oportunidade de chegar aqui; por esta vida; por este país; por esta linda família que tenho. Isso tudo é muito mais do que eu mereço.

Agradeço aos meus pais: Ruben Interian e Svetlana Kovaliova, por seus conselhos; por me dar a educação que tenho e os valores que carrego comigo.

Agradeço o apoio inestimável de minha esposa, Isela Mendoza, durante a preparação desta tese. Amor, se não fosse por você, pode ter certeza - não teria chegado muito longe.

Aos meus filhos Anna e Dario - pelos momentos de felicidade que me deram forças para seguir. Agradeço à minha avó Liudmila, por sempre ter acreditado em mim.

Agradeço a toda a minha família, minha irmã Kristina, meus sogros, meus amigos, aqueles que me apoiaram e me ajudaram - Obrigado.

Agradeço ao professor Reinaldo Rodríguez por servir de exemplo, por sua garra e profissionalismo. Agradeço aos professores Leonardo Murta, Uéverton Souza, Anselmo Montenegro, Aura Conci, que gentilmente aceitaram em colaborar com um aluno desconhecido que chegou ao Brasil, sem nome nem publicações.

Agradeço imensamente a Henrique Cairus, que nos recebeu e nos deu as boas-vindas, a mim e a minha família, e nos fez sentir em casa. Obrigado por estar presente para nos apoiar em tempos difíceis.

Um agradecimento muito especial a Tatiana Ribeiro: sem dúvida, sem ela, essa tese não teria existido. Obrigado, acima de tudo, por ser forte na defesa daquilo que acredita. Madrinha: já não existe nem haverá mais *polarização* entre nós.

Por último, mas não menos importante, quero expressar minha gratidão ao meu orientador Celso Ribeiro: por seu olhar crítico, por ser exigente e sempre saber tirar o melhor de mim. Com você, aprendi o que significa ser um pesquisador, um orientador, um cientista.

A todos que colaboraram, apoiaram e ajudaram - Obrigado.

Resumo

Nesta tese, desenvolvem-se e apresentam-se contribuições para problemas de otimização em redes. A primeira e principal contribuição aborda dois aspectos diferentes do problema de polarização em redes sociais, de interação e de comunicação. Primeiro, uma abordagem probabilística é usada para derivar uma nova e melhor medida para o fenômeno da polarização. Em seguida, um novo problema de otimização é definido, abordando a questão da redução de polarização usando adições mínimas de arestas, de acordo com o princípio de intervenções externas mínimas. Prova-se que o problema é NP-difícil e formulações de programação inteira e heurísticas são propostas para resolvê-lo. Além disso, cinco outras contribuições são desenvolvidas nesta tese, correspondendo a abordagens para problemas de otimização representados em grafos e redes. A segunda contribuição aborda uma variante tipo Steiner do problema do caixeiro-viajante. A terceira descreve uma abordagem para calcular uma combinação ótima de métodos de verificação e validação que cobre as características de qualidade do software. A quarta contribuição apresenta um método para visualizar grandes grafos de histórico de commits, minimizando o número de nós exibidos e preservando a estrutura do grafo. O quinto tema tratado propõe um algoritmo que identifica o grau do polinômio implícito ótimo para ajustar curvas e superfícies. O sexto tema apresenta uma abordagem que deduz a estrutura de vizinhança e regras para um autômato celular baseado em redes regulares que aproxima o comportamento de certos tipos de tumores. Abordagens exatas (como programação inteira e algoritmos tratáveis por parâmetro fixo) e heurísticas (como GRASP, reconexão de caminhos e busca iterada gulosa) são usadas para resolver os problemas de otimização mencionados acima. Juntos, estes problemas abrangem uma ampla gama de áreas da Ciência da Computação, como análise de redes, roteamento e logística, engenharia de software e visão computacional.

Palavras-chave: Otimização em redes, otimização combinatória, métodos exatos, heurísticas, polarização, problema de roteamento, problema de Steiner, problema de cobertura, ajuste de curvas e superfícies, programação inteira, GRASP, reconexão de caminhos.

Abstract

In this thesis, we develop and present a number of contributions to network optimization problems. The first and major contribution addresses two different aspects of the polarization problem in social, interaction, and communication networks. First, a probabilistic approach is used to derive a new and improved measure of the polarization phenomenon. Next, a new optimization problem is defined, addressing the issue of polarization reduction using minimal edge additions in order to cope with a principle of minimal external interventions. The NP-hardness of the problem is proved, and integer programming formulations and heuristics are proposed to solve it. In addition, five other contributions are developed in this thesis, corresponding to approaches to optimization problems represented on graphs and networks. The second contribution addresses the Steiner traveling salesman problem. The third describes an approach for computing an optimal combination of verification and validation methods that covers the software quality characteristics. The fourth introduces a method to visualize large commit history graphs, minimizing the number of displayed nodes and preserving the structure of the graph. The fifth proposes an algorithm that identifies the optimum implicit polynomial degree for fitting curves and surfaces. The sixth presents an approach that deduces the neighborhood structure and rules for a regular, network-based cellular automaton that approximates the behavior of certain types of tumors. Exact approaches (such as integer programming and fixed-parameter tractable algorithms) and heuristics (such as GRASP, path-relinking, and iterated greedy) are used to solve the above-mentioned optimization problems. Altogether, they cover a broad range of areas of Computer Science, such as network analysis, routing and logistics, software engineering, and computer vision.

Keywords: Network optimization, combinatorial optimization, exact methods, heuristics, polarization, routing, Steiner problem, set covering problem, curve and surface fitting, integer programming, GRASP, path-relinking.

Contents

Chapter 1

Introduction

In this chapter, general optimization problem and the specific class of combinatorial optimization problems are introduced. The most frequently used solution approaches are discussed. The optimization problems covered in this thesis are presented, together with the solution approaches used to solve each of them and the associated publications.

1.1 Combinatorial optimization problems

An **optimization problem** is defined in the following way:

Definition 1 *Given a set F and a function $f : F \rightarrow \mathbb{R}$, an optimization problem seeks:*

- *an element $x^* \in F$ such as that $f(x^*) \leq f(x)$ $\forall x \in F$ (minimization); or*
- *an element $x^* \in F$ such as that $f(x^*) \geq f(x)$ $\forall x \in F$ (maximization).*

The element x^* in Definition 1 is named a *global optimum*, f is the *objective function* (selection criterion), and F is the set of *feasible solutions.*

In the case of a **combinatorial optimization problem**, the set F of feasible solutions is required to be finite or countably infinite.

A typical **network optimization problem** is an optimization problem defined over the structure of a graph or a network. The optimization may refer to the minimization or the maximization of the number of elements in some type of subset of nodes or edges, or to the cost of these elements. Nodes may represent e.g. sites in the Internet, cities in a map, or users in a social network. In this thesis, several contributions to network optimization problems are presented.

Four examples of typical network optimization problems are presented below:

Shortest path problem:

Let $G = (V, A)$ be a directed graph, where V is the set of nodes and A is the set of arcs. The origin s and destination t are two special nodes in V. For every pair of nodes $i, j \in V$ that are directly connected, let d_{ij} be the length of arc $(i, j) \in A$. Furthermore, let P_{st} be a path from s to t in G defined as a sequence of nodes $i_1, i_2, \ldots, i_{q-1}, i_q$ with $i_1 = s$ and $i_q = t$. The length of path P_{st} is given by $f(P_{st}) = \sum_{k=1}^{q-1} d_{i_k, i_{k+1}}$.

In the case of the shortest path problem, the feasible solutions set F is formed by all subsets of A that correspond to paths from s to t in G. The objective of the shortest path problem is to find a path $P^* \in F$ that minimizes the objective function $f(P)$ over all paths $P \in F$ from s to t in G [14]. ■

Minimum spanning tree problem

Let $G = (V, E)$ be a graph, where V is the set of nodes and E is the set of edges. Let d_{ij} be the weight associated with each edge $(i, j) \in E$. In addition, let $T = (V, E')$ be a spanning tree of graph G, i.e., a connected subgraph of G with the same node set V and with exactly $|V| - 1$ edges in set $E' \subseteq E$. The total weight of tree T is given by $f(T) = \sum_{(i,j) \in E'} d_{ij}$.

In the case of the minimum spanning tree problem, the set of feasible solutions F is formed by all subsets of edges that correspond to spanning trees of G. The objective of the minimum spanning tree problem is to find a spanning tree $T^* \in F$ such that $f(T^*) \leq f(T)$ for all $T \in F$. [30, 40]. ■

Maximum clique problem

Let $G = (V, E)$ be a graph, where V is the set of nodes and E is the set of edges. A clique is a subset $C \subseteq V$ such that $(i, j) \in E$ for every pair $i, j \in C$ with $i \neq j$. The size $f(C)$ of a clique C is defined to be its cardinality, i.e. $f(C) = |C|$.

In the case of the maximum clique problem, the feasible solutions set F is formed by all subsets of V in which all nodes are pairwise adjacent. The objective of the maximum clique problem is to find a clique $C^* \in F$ such that $f(C^*) \geq f(C)$ for every $C \in F$ [29]. ■

Traveling salesman problem

Let $G = (V, E)$ be a graph, where V is the set of nodes and E is the set of edges. For every pair of nodes $i, j \in V$ connected by an edge $(i, j) \in E$, let d_{ij} be the nonnegative length of that edge. Furthermore, let H be a Hamiltonian cycle in G, i.e., a cycle $(i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n), (i_n, i_1)$ where $(i_n, i_1) \in E$, $(i_k, i_{k+1}) \in E$ for $k = 1, \ldots, n-1$, and $i_j \neq i_k$ for every $j \neq k \in V$, therefore visiting every node exactly once. The total length of this cycle is given by $f(H) = \sum_{k=1}^{n-1} d_{i_k,i_{k+1}} + d_{i_n,i_1}$.

In the case of the traveling salesman problem, the feasible solution set F is formed by all subsets of edges corresponding to Hamiltonian cycles in G. The objective of the traveling salesman problem is to find a Hamiltonian cycle $H^* \in F$ such that $f(H^*) \leq f(H)$ for all $H \in F$ [29]. ■

1.2 Solution approaches

Exact methods for solving optimization problems are those that are guaranteed to produce, in a finite time, a global optimum and a proof of its optimality, in case one exists, or otherwise show that no feasible solution exists. For many combinatorial optimization problems such as the shortest path and the minimum spanning tree problems, there exist efficient exact algorithms running in polynomial time. However, efficient exact algorithms are not known (and are unlikely to exist) for a broad class of optimization problems classified as NP-hard [18]. These problems are often referred to as intractable. Examples of such problems are the maximum clique and the traveling salesman problems. Exact methods for solving these problems are, typically, integer programming techniques or other enumerative procedures running, in general, in superpolynomial-time. Developments in polyhedral theory, combined with efficient algorithm design and advances in computer hardware, have made it possible to solve even large instances of some NP-hard problems. Methods such as branch-and-bound and branch-and-cut are routinely applied to exactly solve these instances in affordable computational time. Such strategies are specially suitable for real-life, limited-size instances of NP-hard problems.

However, even though the size of the problems that can be exactly solved to optimality has been always increasing due to algorithmic and technological developments, there are problems (or problem instances) that are not amenable to be solved by exact methods. Other approaches, based on different paradigms, are needed to tackle such hard and large optimization problems [43].

Opposed to exact methods, approximate methods (or, simply, heuristics) provide feasible solutions that are not necessarily optimal. Approximate methods usually run faster than exact methods. As a consequence, they are capable of handling larger problem instances than exact methods. Very often, heuristics are among the most efficient strategies to handle in practice hard, intractable combinatorial optimization problems. A heuristic is, essentially, any algorithm that provides a feasible solution for a given problem, without necessarily providing a guarantee of performance in terms of solution quality or computational time. They are usually faster than exact methods and provide good, near-optimal solutions in reasonable running times.

The contributions of this thesis make use of exact and heuristic approaches for solving the combinatorial optimization problems that are presented and discussed in the next chapters and appear in the publications in the appendices.

1.3 Contributions to network optimization problems

The major contribution of this work came and evolved from discussions and concerns of the author and his advisor about the growing polarization of society, the lack of dialogue across groups with different political sympathies, and the total absence of debate in the face of the increasing subjectivity of the people and the media. These concerns led them to set out to study the polarization process happening in front of them and to propose analytical tools to model and resolve issues in social and polarized networks. These contributions will be described in the next chapter. The other subjects and problems treated in the thesis followed naturally from their agenda of research on metaheuristics and exact algorithms for combinatorial optimization problems, as well as from some collaborations with other authors.

A number of contributions to the solution of network optimization problems are first reported in this thesis.

1. The first, and certainly major, contribution addresses two different aspects of the polarization problem in networks. Polarization refers to the *division into sharply contrasting groups or sets of opinions or beliefs* [38]. The research is driven by a minimal intervention principle, according to which the strategy to reduce polarization should preserve as much as possible the original structure of the network.

First, a new measure of the polarization phenomenon in networks is presented [24]. The distribution of this measure over the nodes is used for evaluating the strength of polarization of groups of nodes or entire networks, and to estimate the impacts of interventions on polarization. An intervention can be seen as any externally-induced process that modifies the structure of the network, such as a manipulation that adds or removes nodes or edges of the network. The process of removing nodes or edges may be controversial, because it can be associated with the permanent exclusion or deletion of elements (e.g., users, sites, or posts) from a social network. This exclusion is often interpreted as an aggression against freedom of expression in digital environment. On the other hand, the addition of edges can be easily associated with friendship or post recommendations, that are able to connect nodes from different groups, and are more easily acceptable as externally induced changes in the network.

Next, a new optimization problem is defined addressing the issue of polarization reduction using edge additions [20]. The new problem consists in the minimization of the number of edges to be added to a polarized graph in order that any node in a proper node subset A can reach some node not in A by a path with a limited number of edges. This is named the Minimum-Cardinality Balanced Edge Addition Problem (MinCBEAP). The NP-completeness of MinCBEAP is proved by using a reduction from the Minimum-Cardinality Bounded-Eccentricity Edge Addition Problem, which in turn is NP-complete due to a reduction from the set covering problem.

In order to solve MinCBEAP, three integer programming formulations are proposed. All these formulations are applied after a specific instance transformation that reduces the number of nodes in the network. A restarted iterated greedy heuristic is also developed for solving MinCBEAP.

2. The second contribution refers to the selection of optimal routes in the Steiner traveling salesman problem (Steiner TSP), an essential practical modification of the original traveling salesman problem (TSP). Instead of searching for a Hamiltonian cycle visiting all nodes, a minimum-weight closed walk is requested that visits all required nodes.

A GRASP heuristic is proposed [22, 23] for solving the Steiner TSP, using a randomized extension of the nearest neighbor search algorithm in the construction phase. A variable neighborhood descent (VND) strategy exploring a reduced 2-opt neighborhood is used to optimize the best-improving local search scheme. Backward path-relinking and restart strategies are used to improve the efficiency of the GRASP algorithm.

3. The third contribution consists of an algorithm for solving specific instances of the set covering problem that arise from quality management tasks. The relation between software product quality characteristics and the verification and validation methods can be modeled as a bipartite graph [34], and finding the set of methods that properly addresses all quality characteristics can be seen as a set covering problem.

Although the set covering problem is hard to be solved [29], a fixed-parameter tractable algorithm (FPT-algorithm) [16] can solve in reasonable computational time instances with specific characteristics, where the size of some parameter of the input is fixed. FPT-algorithms are exponential in the size of this fixed parameter, while polynomial in the size of the rest of the input. In this case, the number of quality characteristics to be covered is considered as a fixed and small parameter. Following this strategy, an exact and efficient approach is developed for a specific group of instances of this problem.

4. The fourth contribution is related to an optimization problem that arises when visualizing commit history graphs (CHG) in a version control system [10]. Many of the commit nodes in CHG are unnecessary for comprehending the evolution of the project in time. Therefore, an optimal visualization (i.e., with the minimum number of nodes) that reflects the structure of the commit graph must be found.

In this problem, there are no clearly established criterion for deciding which solution will be considered as feasible, that is, the set F must be first defined. To achieve it, a modelling process is driven, identifying two common node structures that can be automatically collapsed: sequential and parallel, decreasing the number of commit nodes. An efficient iterative algorithm is described, which is able to benefit from both sequential and parallel collapsing strategies.

The above contributions correspond to optimization problems represented on graphs and networks. These networks may represent polarized groups of nodes, transportation routes, relations between software quality methods and characteristics, or topologies of the commit history of a project in version control systems. Either exact algorithms or heuristics are used to solve these optimization problems.

1.4 Contributions to other optimization problems

In addition to the previously described contributions to network optimization problems, the thesis offers other research contributions:

5. The fifth contribution addresses a problem that combines discrete and continuous optimization. Implicit polynomials [21] are used to obtain a compact representation of sets of bi- or three-dimensional observations. While most existing algorithms assume the knowledge of the degree of the implicit polynomial that best represents the points, a two-stage algorithm that is able to find the optimum degree is developed. In the first stage, it is assumed that the degree of the polynomial is fixed. A subproblem is solved using a continuous-space heuristic that finds a near-optimal solution of the fitting problem for that degree. In the second stage, the optimum degree is selected from a discrete set of possible values. A heuristic adaptive fitting algorithm is obtained.

6. The sixth contribution regards the use of a cellular automaton that approximates the behavior of certain types of tumors [26]. The automaton is used with the aim of studying the tumor growth in a new manner, more realistic and closer to the reality, where the tumor is neither perfectly regular nor circular. The cellular automaton employs a lattice in the form of a regular network and is defined from a continuous deterministic differential equation model. The neighborhood structure of the regular network is deduced, and specific rules for updating the states of the nodes are inferred from the continuous model using different model settings.

1.5 Structure of the thesis

This work is organized as follows. Chapter 2 presents the contributions to polarization problems in social, interaction, and communication networks. Chapter 3 addresses the Steiner traveling salesman problem. Chapter 4 describes an approach for computing an optimal combination of verification and validation methods that covers the software quality characteristics. Chapter 5 introduces a method to visualize large commit history graphs, minimizing the number of displayed nodes and preserving the structure of the graph. Chapter 6 proposes an algorithm that identifies the optimum implicit polynomial degree for fitting curves and surfaces. Chapter 7 presents an approach that deduces the neighborhood structure and rules for a regular-network based cellular automaton. Concluding remarks are drawn in the last chapter.

1.6 List of publications

The complete list of full articles, extended abstracts, and short abstracts in conference proceedings that resulted from the research developed in this thesis is presented below.

1.6.1 Full papers

1. Mendoza, I.; Souza, U.; Kalinowski, M.; Interian, R.; Murta, L. G. P., "An efficient algorithm for combining verification and validation methods". In *SOFSEM 2019: Theory* *and Practice of Computer Science (eds. B. Catania, R. Královic, J. Nawrocki, and G. Pighizzini)*, Lecture Notes in Computer Science 11376 (2019), 324–340.

2. Interian, R.; Ribeiro, C. C., "Minimum-cardinality balanced edge addition in polarized networks". *Journal of Combinatorial Optimization*, submitted for publication (2019).

3. Interian, R.; Ribeiro, C. C., "An empirical investigation of network polarization". *Applied Mathematics and Computation* 339 (2018), 651–662.

4. Interian, R.; Ribeiro, C. C., "A GRASP heuristic using path-relinking and restarts for the Steiner traveling salesman problem". *International Transactions in Operational Research* 24 (2017), 1307–1323.

5. Interian, R.; Otero, J. M.; Ribeiro, C. C.; Montenegro, A. A., "Curve and surface fitting by implicit polynomials: Optimum degree finding and heuristic refinement". *Computers & Graphics* 67 (2017), 14–23.

6. Interian, R.; Rodríguez-Ramos, R.; Valdés-Ravelo, F.; Ramírez-Torres, A.; Ribeiro, C. C.; Conci, A., "Tumor growth modelling by cellular automata". *Mathematics and Mechanics of Complex Systems* 5 (2017), 239–259.

7. Cesario, C.; Interian, R.; Murta, L. G. P., "DyeVC: An approach for monitoring and visualizing distributed repositories". *Journal of Software Engineering Research and Development* 5 (2017), 1–34.

1.6.2 Extended abstracts

8. Interian, R.; Ribeiro, C. C., "An iterated greedy heuristic for the minimum-cardinality balanced edge addition problem". *MIC 2019: 13th Metaheuristics International Conference*, Cartagena (2019), 195–198.

9. Interian, R.; Ribeiro, C. C., "GRASP with restarts heuristic for the Steiner traveling salesman problem". *MIC 2017: 12th Metaheuristics International Conference*, Barcelona (2017), 106–115.

1.6.3 Abstracts

10. Interian, R.; Moreno J. R.; Ribeiro, C. C. "Minimum-cardinality Balanced Edge Addition in Polarized Networks". *32nd Conference of the European Chapter on Combinatorial Optimization*, St. Julian's (2019), p.58.

11. Interian, R.; Moreno J. R.; Ribeiro, C. C. “Minimum-cardinality Balanced Edge Addition in polarized networks: Formulation and solution approaches”. *INFORMS Annual Meeting*, Seattle (2019) (to appear).

12. Ribeiro, C. C; Interian, R. “Minimum-cardinality Balanced Edge Addition in polarized networks”. Conferences Abstracts Book, *30th European Conference on Operational Research*, Dublin (2019), p.267.

13. Ribeiro, C. C.; Interian, R., “An empirical investigation of network polarization”. *29th European Conference on Operational Research*, Valencia (2018), p.346.

14. Interian, R.; Ribeiro, C. C. “An empirical investigation of network polarization”. Program Abstracts, *INFORMS Annual Meeting*, Phoenix (2018), p.216.

15. Interian, R.; Ribeiro, C. C., “A GRASP with restarts heuristic for the Steiner traveling salesman problem”. *31st Conference of the European Chapter on Combinatorial Optimization*, Koper (2017), p.21.

16. Ribeiro, C. C.; Interian, R., “Heuristics for the Steiner traveling salesman problem”. *INFORMS Annual Meeting*, Nashville (2016), p.290.

Chapter 2

Polarization issues in social networks

Polarization is a widely known phenomenon that has been discussed by politicians, media, and researchers in recent years [37, 47]. This subject has also attracted the attention of thinkers throughout history.

Since the 19th century, John Stuart Mill, an important philosopher and political theorist, claimed that dialogue across lines of political difference is a key prerequisite for sustaining a democratic citizenry [35]. Hannah Arendt also asseverated that debate is irreplaceable for forming enlightened opinions that reach beyond the limits of one's own subjectivity to incorporate the standpoints of others [2]. More recently, several world leaders have often expressed concern about polarization problems caused by social media [3, 48]. From sociologists to economists, many are interested in studying the behavior and interactions in social networks that rule the opinion formation process.

According to the Oxford Dictionaries, polarization is the *division into sharply contrasting groups or sets of opinions or beliefs* [38]. Among the negative effects caused by polarization, we can mention the proliferation of fake news and the rise of extremism and intolerance.

Recently, the causes of the proliferation of flat-earth believers, i.e., people who believe that the Earth is actually flat, were investigated by Landrum [31], revealing the role of the video-sharing platform YouTube on this proliferation. This work showed that the algorithms the platform uses to guide people to topics that might interest them makes it easier for a user to end up in a misinformation echo chamber. The study concludes that the most effective instrument to combat disinformation is to provide users of the platform with quality and plural information when watching videos on some subject, thereby breaking the polarization and the isolation of these groups.

2.1 A new measure of network polarization

In the article reproduced in Appendix A, Interian and Ribeiro [24] proposed and explored a new strategy for measuring the polarization phenomenon in networks. Considering that the nodes of a network are partitioned into a number of groups, the homophily of any node of the network is defined as the ratio between the number of its successors that belong to the same group and the total number of its successors. The distribution of the homophily over the network is used as an indicator of polarization.

Next, a probabilistic approach is used to define a new and improved polarization measure, which is based on the calculation, for each node, of the probability (p-value) of observing a number of "same-type successors" that is greater than or equal to the actual number of same-type successors observed for this node. A statistical test evaluates the significance of the homophily values for each node of the network. A one-tailed test is used, because the possible presence of heterophily (the opposite of homophily) in some nodes is irrelevant for evaluating the strength of polarization.

The distribution of the obtained p-values is used for evaluating the strength of polarization of the network. The empirical distribution function of the p-values can be used to compare, in a more informative way, the polarization of different groups of nodes and even of entire networks. Several real-life networks from different sources have been employed as case studies to illustrate the usefulness of the proposed tools.

The case studies included the network of books about U.S. politics sold by Amazon.com; the network of political blogs that emerged during the 2004 U.S. presidential election; the World trade network built using information about bilateral trade data between countries; and the network of interactions between characters of the novel "A Song of Ice and Fire", also called "Game of Thrones". Strong polarization is observed on most case studies.

The approach proposed in the article in Appendix A is generic and may be applied to a variety of real networks and situations. In particular, the p-value distributions can be used to estimate the impacts of interventions in the polarization of the nodes of a network. An intervention can be seen as any externally-induced process that modifies the structure of the network, such as a fact-checking campaign, a marketing campaign, a regulatory action or some direct manipulation that adds or removes nodes or edges of the network.

The investigation of the impact of external interventions is relevant in a world characterized by extreme political and ideological polarization. In the next section, we introduce an optimization problem associated with the idea of carrying out direct interventions in a network with the goal of reducing its polarization. This line of research is driven by a *minimal intervention* principle, according to which we aim to reduce polarization with minimal interventions that preserve, as much as possible, the original structure of the network.

2.2 Balanced edge addition to reduce polarization

Interian and Ribeiro showed in [24], the article reproduced in Appendix A, that in order to reduce polarization, networks can be treated by external interventions consisting of the addition or the removal of vertices and edges. The process of adding new vertices is often difficult to be performed in real networks. On the other hand, removing vertices or edges may be controversial, because it can be interpreted as the permanent exclusion or deletion of elements (e.g., users, sites, or posts) from a social network. This exclusion is often interpreted as an aggression against freedom of expression in digital environment.

2.2.1 Complexity and integer programming approaches

In the paper reproduced in Appendix B, Interian, Moreno and Ribeiro [20] introduced a new optimization problem addressing the issue of polarization reduction by edge additions. The new problem consists in the minimization of the number of edges to be added to a polarized graph $G = (V, E)$, in order that any vertex in a proper vertex subset $A \subset V$ could reach some vertex of $V \setminus A$ in the resulting graph by a path with at most D edges. The parameter D provides an upper bound to the number of edges in a path from any vertex in A to some (closest) vertex in $V \setminus A$. This problem was named the Minimum-Cardinality Balanced Edge Addition Problem (MinCBEAP).

Parameter D has a central role in ensuring the reduction of the level of polarization of the network. In the specific case when $D = 1$, the problem is easy to solve. Typically, D takes very small values: 2, 3, or maybe 4. This guarantees that every vertex in the set $A \subset V$ will have a "way out" from the set in a small number of steps. In practice, this means that every vertex has a fast way to access information from outside the group.

Interian, Moreno and Ribeiro [20] also showed that MinCBEAP is NP-complete for any $D \geq 2$ by using a reduction from the Minimum-Cardinality-Bounded-Eccentricity edge addition problem (MCBE), which in turn is NP-complete due to a reduction from the Set Covering problem.

Three integer programming formulations of MinCBEAP are developed in the article reproduced in Appendix B. The first and most general formulation uses the Miller-Tucker-Zemlin constraints to avoid cycles [36]. This formulation has integer and binary variables, and a quadratic number of variables and constraints.

The second formulation is a modification of the model proposed in [13] that solves the linear feasibility problem associated to MCBE. In this formulation, all variables are binary. The number of variables and constraints is also quadratic.

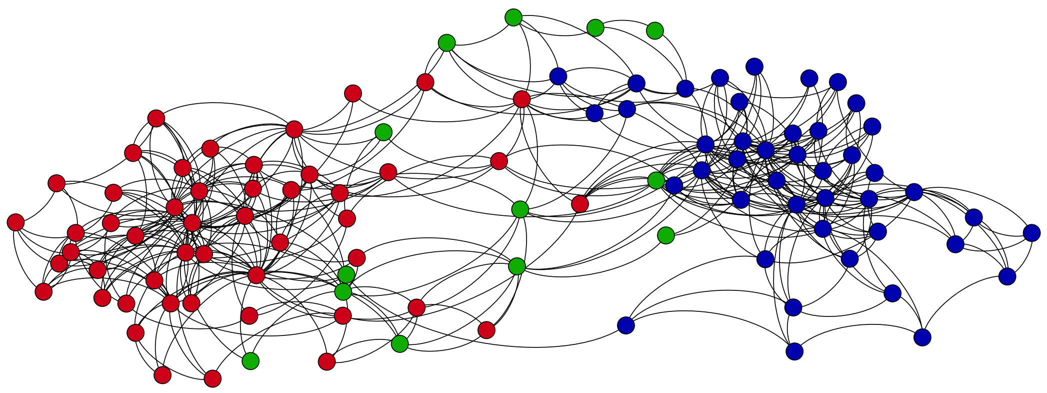
The third formulation proposed in this work is eventually the most efficient one. In this formulation, we take into account the fact that the current distance between some specific vertices on the graph can not increase after the addition of the new edges. A detailed description of this formulation can be found in the paper reproduced in Appendix B. All variables are binary. The number of variables and constraints is $O(nD)$, with $n = |A|$.

Computational results are presented and discussed for both randomly generated and real-life instances. On the real-life instances, we showed that polarization can be reduced to the desired threshold by the addition of a few edges, as established by the minimum intervention principle that guided the problem formulation.

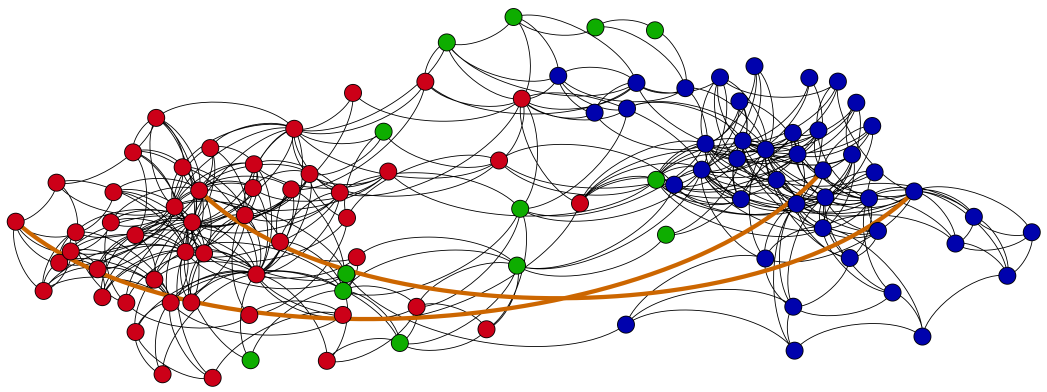
The political books instance from Interian, Moreno and Ribeiro [20] was used to illustrate the solution of problem MinCBEAP. The number of edge additions needed to solve the MinCBEAP problem for conservative and liberal groups is equal to one and two, respectively, as shown in Table 5 of [20]. This implies that there is one vertex (let it be v_1^c) in the conservative group that will be connected to some vertex from the liberal group, and that there are two vertices (let's say v_1^l and v_2^l) in the liberal group that will be connected to vertices in the conservative group. Consequently, the solution for the entire graph has two edges: those connecting v_1^l and v_2^l to two vertices from conservative group, one of which is v_1^c. This solution is shown in Figure 2.1, with conservative and liberal groups being represented by red and blue vertices, respectively.

In practice, this solution may be interpreted as a recommendation, for a reader that already bought some book v_i about politics, of buying another book v_j of different ideological orientation, corresponding to the edge (v_i, v_j) added to the graph.

An interesting conclusion is that in strongly polarized groups, there is often some easy way of spreading polarization-breaking information by the addition of few edges. This is a consequence of the fact that the higher is the density of a polarized group of vertices in a network, the smaller is the number of edges in the optimal solution.



(a)



(b)

Figure 2.1: Political books instance and its solution. (a) Original political books instance with vertex colors representing conservative (red), neutral (green), and liberal (blue) groups in terms of ideological orientation. (b) Political books instance with two new orange edges representing the solution of problem MinCBEAP for conservative and liberal groups.

2.2.2 Heuristic solution

The integer programming formulations developed in the article reproduced in Appendix B and mentioned in the previous section were unable to solve large MinCBEAP instances with more than 2000 vertices. Therefore, a restarted iterated greedy heuristic was developed and introduced in the article by Interian and Ribeiro [25] reproduced in Appendix C for solving large MinCBEAP instances. The iterated greedy heuristic is especially suitable for optimization problems without weights, in which the objective function is related to the cardinality of the solution. Local search procedures often do not perform satisfactorily in the context of optimization problems with such characteristics.

An iterated greedy heuristic starts from a greedy or a semi-greedy candidate solution, and generates a sequence of solutions using two main phases: destruction and reconstruction. During the destruction phase, some edges are removed from the current solution. The reconstruction procedure applies a greedy construction heuristic to reconstruct a complete solution [44].

In the Restarted Iterated Greedy (RIG) variant, an external loop is added to the original iterated greedy heuristic [44]. Instead of starting with a completely greedy solution, a semi-greedy solution is used and the iterated greedy construction is embedded into a multi-start procedure.

The polarization reduction methodology proposed in this work could be useful to decrease the intransigence and inflexibility in social and communication networks on different points of view on issues such as politics, traditions, and customs. Nowadays, issues such as capital punishment, abortion and extremist political ideologies cause a deep division in society. There is an almost total absence of dialogue between groups with different worldviews, which certainly can not explain, by themselves, the complexity of the real world. A minimal regulation of the social networks, such as proposed by the *minimum intervention principle* – which is completely opposed to any idea of censorship –, can contribute to giving to their users the opportunity to get out of the echo chambers created and reinforced by polarization.

Chapter 3

The Steiner travelling salesman problem

The traveling salesman problem (TSP) is one of the fundamental combinatorial optimization problems [18, 49] and has numerous real-life applications in transportation, logistics, vehicle routing, genome sequencing, and other areas. Given a set of nodes and the distances between them, it consists in finding the shortest route that visits each node exactly once and returns to the first. Its decision version is proven to be NP-complete by a reduction from the Hamiltonian cycle problem [18].

However, in many practical applications, it is more frequent to find the following variant of the TSP. A set $V_R \subseteq V$ of required nodes is given. Instead of searching for a Hamiltonian cycle visiting all nodes, a minimum-weight closed walk is requested that visits only the required nodes. Thus, nodes can be visited more than once and edges may be traversed more than once. The so-called Steiner traveling salesman problem (Steiner TSP) was first proposed in [11, 17], where its NP-hardness was also proved. The Steiner TSP is specially suitable to model network design [6], package delivery [49, 50], and routing [32] problems.

Most studies on the Steiner TSP focus on integer programming formulations and valid inequalities. The articles by Interian and Ribeiro [22, 23], reproduced in Appendices D and E, respectively, proposed an heuristic approach for solving Steiner TSP. In these articles, a GRASP heuristic with path-relinking and restarts was developed for solving the Steiner Traveling Salesman Problem.

The algorithm used in the construction phase is a randomized extension of the nearest neighbor heuristic for the Traveling Salesman Problem. A variable neighborhood descent (VND) strategy exploring a reduced 2-opt neighborhood is used to optimize the best improving local search scheme. Backward path-relinking and restart strategies are used to improve the efficiency of the GRASP algorithm.

Extensive computational results for a set of previously used instances are reported. It is shown that pure GRASP results are substantially improved by GRASP with path-relinking, and that GRASP with path-relinking results are slightly (but also systematically) improved when a restart strategy is used [22].

In addition, we also genetared and considered a set of larger test instances derived from real-life graphs for future benchmarking purposes. Since neither optimal values nor even upper bounds have been previously reported for these instances, the solutions obtained by the GRASP with path-relinking and restarts heuristic cannot be directly compared with other previous solutions.

As a step towards avoiding this difficulty and facilitating the research on this problem, we made all test instances and their best known solution values available at a public URL. It is hard to underestimate the importance of data sharing for the progress of science, but to a much greater extent this applies to computer science. Many of the published studies are irreproducible simply due to lack of access to instances or data sets. Recently, platforms like *Mendeley Data* have emerged to help researchers to store, share, publish and find data for their investigations. Unlike classical scientific venues (journals and conferences) that publish papers describing scientific methodologies and results, these data sharing tools offer an efficient way to provide access to benchmark instances for a large scope of fields and problems.

Chapter 4

Combining verification and validation methods to improve software quality

Verification is used to ensure that the software product is built in the correct way, complying with the previously defined specifications [7]. On the other hand, validation guarantees that the product is adherent to the user needs [7].

An appropriate combination of verification and validation (V&V) methods is important to improve software quality control throughout the development process and to reduce costs. There are eight well defined software product quality characteristics in the ISO 25010 standard [27]. There are dozens or even hundreds of V&V methods that have been proposed over the years.

Mendoza et al. [34], in the article reproduced in Appendix F, proposed a novel algorithm that efficiently combines V&V methods in order to properly cover a set of quality characteristics. The algorithm obtains an optimal combination with the smallest number of methods covering all the software quality characteristics in reasonable computational time.

The relation between the characteristics and the methods can be modeled as a bipartite graph. The methods and the characteristics are both disjoint independent sets. An edge between a method m and a quality characteristic c indicates that m covers c, that is, if m is used properly for V&V, then c will be satisfied. Therefore, finding a set of methods that together properly addresses all quality characteristics of interest can be seen as a set covering problem (SCP) [29].

SCP is a classic NP-hard problem in the computational complexity area, whose decision version belongs to Karp's list of 21 NP-complete problems [29]. This means that when the number of methods or quality characteristics increase, the performance of an algorithm that aims to combine them in an optimal way would drastically decrease.

Although set covering is considered hard to be solved, an FPT-Algorithm (fixed-parameter tractable algorithm) that effectively solves the problem is proposed through the theoretical framework of Parameterized Complexity [16], considering the number of quality characteristics k to be covered as a fixed, *and small*, parameter. The FPT-algorithm sacrifices the execution time, which can be exponential, but guarantees that the exponential dependency is restricted to the parameter k, which means that the problem can be solved efficiently for small values of this fixed parameter.

The algorithm proposed in [34] runs in time $O(f(k) \times n)$, where k is the number of quality characteristics, n is the number of methods, and $f(k)$ is some function of k. It is also shown that $f(k)$ has a closed-form expression, and that for $k = 8$, $f(k)$ is bounded by 3×10^5. Since the number of quality characteristics k is currently fixed, and will not increase significantly in time, it is possible in this case to consider $f(k)$ as a constant. This implies that the algorithm is efficient in the sense that it depends linearly on n.

Computational experiments show that the algorithm is capable of solving all the proposed instances in less than one second. The proposed algorithm provides the optimal combination of V&V methods that cover a set of chosen quality characteristics to be considered when developing a software product. The algorithm can be applied to instances of different sizes, making this approach scalable, i.e., suitable for larger case studies, with more V&V methods. Companies may choose to complement the minimum set of V&V methods we provided with others to further assure the quality of the product. In practice, other factors, such as the cost of applying each method, should be considered when taking the final decision.

Chapter 5

Optimizing the visualization of commit history graphs

Distributed version control systems have become more and more frequent during the software development process. Such systems bring more flexibility, but also increasing complexity to manage and monitor multiple repositories, as well as their branches [8, 12, 39].

Cesario, Interian, and Murta [10], in the article reproduced in Appendix G, proposed an approach to assist developers and repository administrators in identifying and visualizing dependencies among clones of distributed repositories. It allows understanding what is going on around one's clone and visualizing the relationships between existing clones. The tool called DyeVC that implements the approach was evaluated over open source projects, showing how they could benefit from using the above-mentioned features.

DyeVC collects information about different repositories and presents it visually to the user. One of the main visualizations that the user can take advantage of is the representation of the *commit history graph* (CHG). Formally, the CHG is a directed acyclic graph. Each of its nodes represents a known commit in the topology. There is an arc between two commits if and only if they are in a parent-child relationship, that is, commit A is parent of commit B if B is based on A. Additionally, a color is assigned to each commit reflecting the information about the presence or absence of this commit in the local repository and the peers' repositories.

The initial commit is usually unique, and in this case it is the only node without a parent. There may be commits with more than one parent, because of the non-fast-forward merge operations. Commits also may have more than one child.

The visualization of CHG can easily have thousands of nodes, one for each commit in the topology. Nevertheless, despite the high number of nodes, the user is often interested in understanding the structure of the project's commit history.

Many of the commit nodes in CHG are unnecessary for comprehending the evolution of the project in time. For instance, a linear sequence of 20 commit nodes that are ordinary revisions and that belong to all clones (i.e., all have the same color) could be collapsed in the visualization into just one single node. The following question arises: is there an optimal (with the minimum number of nodes) visualization that reflects the structure of the commit graph?

In this case, we are facing an optimization problem that is not completely defined: the number of nodes that are showed to the user is minimized, but there is no clearly established criteria for deciding which solution will be considered as feasible. This kind of problem is extremely common in practical applications of optimization methods. Therefore, modelling is necessary to formally define which solution will be considered as valid, i.e., define the conditions or restrictions that the solution graph must satisfy.

Modelling consists in the identification of two common node structures that can be automatically collapsed: sequential and parallel [10]. The former contains a sequence of commits of the same type, where each of them has just one successor and one ancestor, that is, simple paths. This kind of structure can be collapsed, because it does not represent any additional information besides the fact that some sequential work was performed. On the other hand, the latter contains one fork node and one merge node, with at most one (regular or collapsed) 2-degree node in each branch, between the fork and the merge nodes. This parallel structure represents a simple logical ramification in the project history that separates the work in two branches that are then joined again.

Note that collapsing parallel structures may lead to new sequential structures, and vice-versa. Cesario, Interian, and Murta [10], in the article reproduced in Appendix G, described an iterative algorithm that works in phases to benefit from both sequential and parallel collapse strategies. This algorithm is efficient because it performs a small number of iterations, each of them running in linear time.

To measure the impact of automatic collapsing algorithm on real-life repositories, several computational experiments were executed. With only two iterations of the algorithm, where each iteration applies sequential and parallel collapses, the number of nodes is reduced by an average of 73% compared to the original commit history graph. In some cases, the reduction in the number of nodes surpassed 90%.

Using this method, significantly lower running times and memory consumption values are obtained during the visualization, compared with values before the automatic collapsing. It was possible to visualize repositories with tens of thousands of nodes, which could not be represented before, without the application of the collapsing algorithm.

Chapter 6

A two-stage heuristic for curve and surface fitting

The problem of 2D and 3D object representation arises in different computer science areas. Modelling, 3D reconstruction and recognition tasks depend totally on a good representation of the observed objects. However, it is usual to receive noised or incomplete real world data. The models are obtained from images, videos, 3D scanners and other capture devices [19]. The nature of this devices allow to obtain a finite and discrete amount of data from the original object, commonly as a point set. The process of finding a model that better fits the observations is the goal of a lot of research work in recent years [4, 51].

An *implicit polynomial* (IP) model [45] is used to obtain a compact representation of sets of bi- or three-dimensional observations, appearing as a powerful tool for modelling real objects when compared to other representation types. An implicit polynomial can represent a curve (2D), or a surface (3D). Most existing IP fitting algorithms assume the knowledge of the degree of the implicit polynomial that best represents the points. In the paper by Interian et al. [21] reproduced in Appendix H, an IP fitting problem of finding the optimum degree needed for representing the data set is addressed.

This problem combines discrete and continuous optimization in the following way. The task of identifying the optimum degree needed for representing the data set is a non-trivial discrete optimization problem, since the degrees are represented by integer numbers. A too small implicit polynomial degree does not allow to represent complex objects; at the same time, the greater the degree, the higher the possibility of overfitting. Moreover, the problem of finding the implicit polynomial of some fixed degree that best fits the points is a continuous problem, since the coefficients of the polynomials can take any real value.

To solve the problem, a two-stage heuristic is proposed by Interian et al. [21]. Initially, it is assumed that the degree of the polynomium is a fixed integer g. The fixed degree fitting algorithm solves a subproblem using a continuous-space heuristic that finds a near-optimal solution of the fitting problem for the degree value equal to g. Two metaheuristics designed specifically for continuous problems are used: Particle Swarm Optimization (PSO) and Differential Evolution (DE).

Next, the optimum degree is selected from a (discrete) set of allowed degrees. Since the zero set of any odd degree polynomial is always unbounded [46], the proposed algorithm only uses even degrees. Additionally, the advantage of having a compact representation of the object is lost when very high degrees are used. Therefore, the set of feasible degrees contains only even integers, bounded by some constant n, which depends on several factors, such as the problem dimension (2D or 3D). The proposed algorithm is based on the idea of gradually increasing the degree, while there is an improvement in the smoothness of the solutions.

As a result, a heuristic adaptive fitting algorithm is obtained. This algorithm is better than the others existing [5], not only because it can automatically find the required degree of the implicit polynomial, but also because it can improve the quality of the solutions obtained by other fitting algorithms, such as the classical linear and 3L.

The experiments confirm the validity of the approach for the selected 2D and 3D data sets. The fits obtained by the proposed algorithm are interpretable. Additionally, the results for specifically generated noisy data sets indicate that the proposed algorithm is not very sensitive to noise, which allows it to be used for recognition tasks.

Chapter 7

Modelling tumor growth using regular graphs and cellular automata

Tumor growth is a complex process that requires mathematical and computational modeling approaches for studying real-life cancer behavior. Cell-based and cell-centered approaches for the study of biological soft tissues, and particularly tumors, have been widely used, with *cellular automata* (CA) being one of the most successful models. Cellular automata models have been used to investigate avascular tumor growth [15, 28], tumor cell invasion [1], and tumor interactions with various environmental factors [42].

In the article by Interian et al. [26] reproduced in Appendix I, cellular automata are used for studying the process of tumor growth, approximating the behavior of certain types of tumors, specially in the avascular stage. The automaton is used with the aim of studying tumor growth in a new manner, more realistic and closer to the reality, where the tumor is neither perfectly regular nor circular. The cellular automaton employs a lattice in the form of a regular graph and is defined from a continuous deterministic differential equation model developed in [41]. The neighborhood structure of the regular graph is deduced, and specific rules for updating the states of the nodes are inferred from the continuous model using different model settings.

In this work, a simple two-state cellular automaton is considered, where the state 0 represents a normal cell and the state 1 represents a tumoral cell. The automaton is defined in a square lattice represented by a regular graph. First, from the differential equation model, it is known that the increments of the tumor cell coordinates should be similar, for some fixed moment of time, in all directions. Then, an appropriate neighborhood structure is determined using the fact that the influence zone (neighbors) of any cell c must have cells with the same distance to c, leading to a well known von Neumann neighborhood.

To infer the behavior of the CA from the continuous model, a new stochastic rule is created [26]. The rule, also called transition function, receives a current state of some cell c and its neighbors, and returns the state of c in the next moment of time. Since the rule is stochastic, it also depends on a random variable that expresses the probability of appearance of a new tumor cell during a time unit step.

In some cases, the closed-form of the specific stochastic rule is obtained analyzing the relationship between the tumor radius growth speed and the radius itself, making it possible to identify the average number of new tumor cells created from one tumor cell per unit of time, for each radius value. In other cases, a closed-form expression can not be found for the referred relationship, making it necessary to use numerical methods.

To evaluate the performance of the model, the evolution of the tumor radius with time for several runs of the discrete CA model was compared with the continuous model. The differences in the tumor radius between the models are small and are actually due to their different nature. The consistency of the model was also evaluated using the coefficient of variation of the tumor radius variable. Validation tests confirmed that the CA model accurately captures the hypothesis of the described phenomena.

The methodology exposed in this work can be applied to other continuous models in order to represent the growth processes in a nonidealized and nondeterministic way.

Chapter 8

Concluding remarks

The academic formation of the author of this thesis began at *Universidad de La Habana*, an institution with 290 years of history, where he obtained a bachelor degree (five years of studies) in Computer Science with a work titled "Classification of molecules represented by graphs" in 2011. Next, he earned a master degree in Mathematics, in the area of Optimization, with a dissertation titled "Curve and surface fitting by implicit polynomials" in 2015.

The author was accepted as a PhD student at *Universidade Federal Fluminense* in 2015 with a CNPq doctorate scholarship, where he found a very favorable environment to continue his graduate studies in Computer Science. The excellence of the faculty of the Institute of Computing greatly contributed to this, as well as the inauguration between 2014 and 2016 of the two new buildings of the Institute, which considerably improved the infrastructure used for research.

During the first year of the doctoral studies, the author took six graduate courses, achieving 9.4 out of a maximum of 10.0 grade point average: Computer Systems, Treatment of Uncertainty, Graph Theory, Computational Intelligence, Optimization in Graphs, and Supervised Research.

During the very first year of the doctorate, the author developed and completed joint collaborations with his advisor and two professors of the Institute of Computing. Together with professor Anselmo A. Montenegro, the study of implicit polynomial fitting problem emerged as a continuation of the author's master dissertation, leading to the publication of the article Interian, Otero, Ribeiro and Montenegro [21], "Curve and surface fitting by implicit polynomials: Optimum degree finding and heuristic refinement", which appears in Appendix H. In addition, in collaboration with professor Reinaldo Rodríguez-Ramos (one of the most cited scientists from *Universidad de La Habana*, who was visiting the Institute of Computing) and other co-authors, we worked on a research project that lead to the article Interian, Rodríguez-Ramos, Valdés-Ravelo, Ramírez-Torres, Ribeiro and Conci [26], "Tumor growth modelling by cellular automata", presented in Appendix I. Both papers appeared in 2017.

In parallel, the author and his advisor, professor Celso C. Ribeiro, began to have some concerns when observing the increasing polarization of society, the lack of dialogue across groups with different political sympathies, and the total absence of debate in the face of the increasing subjectivity of the people and the media. These concerns led them to set out to study the polarization process happening in front of them, investigation that led to the major contribution of this doctoral thesis.

As a direct result of this work, we developed a framework for measuring the polarization phenomenon and evaluating the strength of polarization of groups of nodes, or of entire networks, and to estimate the impacts of external interventions on the polarization of a network. The paper Interian and Ribeiro [24], "An empirical investigation of network polarization", presented in Appendix A, condensed this research and was published in the highly cited Elsevier's journal *Applied Mathematics and Computation*, in 2018. Another paper by Interian, Moreno and Ribeiro [20], "Polarization reduction by minimum-cardinality balanced edge additions: Formulations, complexity, and integer programming approaches", which appears in Appendix B, addressed a practical solution of the polarization problem using the strategy of connecting people with different points of view. This article was completed and submitted to Springer's *Journal of Combinatorial Optimization.*

The author and his advisor participated at several meetings, workshops and conferences presenting their research methods and results, as enumerated in the first chapter of this thesis. In particular, the subject of this research was presented as an invited plenary lecture at the 13th Metaheuristics Intenational Conference (MIC 2019) in July 2019 with the title "Polarization reduction by minimum edge additions", illustrating the relevance of the subject for the optimization and metaheuristics communities. In this lecture, in particular, we used as a driving motivation the existing polarization – sometimes hidden, sometimes explicit – in the optimization community between researchers more biased towards metaheuristics and others that rely exclusively on exact methods.

During the time he was involved with the research on polarization issues, the author continued his interactions with other professors of the Institute of Computing of *Universi-*

dade Federal Fluminense. In collaboration with professor Leonardo Murta, he published the article Cesario, Interian and Murta [10], "DyeVC: an approach for monitoring and visualizing distributed repositories", which appears in Appendix G. One of the main contributions of this article addresses the visualization of commit history graphs. This article is related to the master's dissertation "Awareness over distributed version control systems" [9] of Cristiano Cesário presented at *Universidade Federal Fluminense* in 2015, for which the main contribution of the author was to propose and develop the algorithm for optimal graph visualization.

Together with professors Uéverton Souza and Leonardo Murta, the study of verification and validation methods led to a novel algorithm that efficiently combines these methods in order to properly cover a set of quality characteristics. This algorithm was the result of a collaboration that culminated with the master's dissertation "Combining verification and validation methods to cover software quality characteristics" [33] of Isela Mendoza at *Universidade Federal Fluminense* in 2018, and appeared in the article Mendoza, Souza, Kalinowski, Interian and Murta [34], "An Efficient Algorithm for Combining Verification and Validation Methods", which appears in Appendix F.

Altogether, this research and these publications cover a broad range of areas of Computer Science, such as network analysis, routing and logistics, software engineering, and computer vision. Exact approaches (such as integer programming and fixed-parameter tractable algorithms) and heuristics (such as GRASP, path-relinking and iterated greedy) are used to solve the optimization problems, showing that there is no universal technique to solve all kinds of problems. Instead, each of them must be treated differently, exploiting its own characteristics.

The publication of test problems and algorithm parameters should be encouraged by journal editors and they all should be made available to other researchers to support their investigations. In other cases, test data are available, but not the detailed results obtained in computational experiments. As a result, many times previously published algorithms can not be even even compared with new approaches, simply due to lack of access to test instances, algorithm parameters, and detailed results obtained in computational experiments. We experienced such difficulties, as reported in Section 5 of article Interian and Ribeiro [22], reproduced in Appendix D. Recently, Mendeley Data – and other repositories – has emerged to assist researchers to store, share, publish and find data for their investigations. Unlike classical scientific venues (such as journals and conferences) that publish papers describing scientific methodologies and results, these data sharing tools offer an efficient way to provide access to benchmark instances for a large scope of fields and problems. As an example, our data set "Polarization case studies" [24] providing the data used in the computational experiments reported in our articles [20, 24] has had 185 views and 48 downloads since its recent publication in 2018.

Finally, the author considers that the main contribution of this thesis, regarding polarization evaluation and reduction, could be a useful tool to decrease the intransigence and inflexibility in social and communication networks on different points of view on issues such as politics, traditions, and customs. In particular, issues such as capital punishment, abortion and extremist political ideologies cause a deep division in society. Today, we observe an almost total absence of dialogue between different worldviews, that certainly can not explain, by themselves, the complexity of the real world. A minimal regulation on social networks, such as proposed by the *minimum intervention principle* – which guided part of this research and is completely opposed to any idea of censorship –, can contribute to giving to their users the opportunity to get out of the echo chambers created and reinforced by polarization. After all, quoting Hannah Arendt [2], *"I form an opinion by considering a given issue from different viewpoints, by making present to my mind the standpoints of those who are absent. The more people's standpoints I have present in my mind while I am pondering a given issue, . . . the more valid my final conclusions."*

References

[1] Anderson, A.; Rejniak, K. Microenvironment driven invasion: a multiscale multimodel investigation. *Journal of Mathematical Biology 58* (2009), 579–624.

[2] Arendt, H. *Between Past and Future.* Viking Press, New York, 1968.

[3] BBC News. Angela Merkel: The big internet platforms have become an eye of a needle. Last access on 2017-09-26. http://www.bbc.com/news/technology-37798762.

[4] Ben-Yaacov, H.; Malah, D.; Barzohar, M. Recognition of 3d objects based on implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32* (2010), 954–960.

[5] Blane, M. M. The 3l algorithm for fitting implicit polynomial curves and surfaces to data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22* (2000), 298–313.

[6] Borne, S.; Mahjoub, A. R.; Taktak, R. A branch-and-cut algorithm for the multiple Steiner TSP with order constraints. *Electronic Notes in Discrete Mathematics 41* (2013), 487–494.

[7] Bourque, P.; Fairley, R. E. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, 3rd ed. IEEE Computer Society Press, Los Alamitos, 2014.

[8] Brun, Y.; Holmes, R.; Ernst, M. D.; Notkin, D. Proactive detection of collaboration conflicts. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering* (New York, 2011), ACM, pp. 168–178.

[9] Cesário, C. Awareness over distributed version control systems. Master's thesis, Institute of Computing, Universidade Federal Fluminense, Niterói, 2015.

[10] Cesario, C.; Interian, R.; Murta, L. DyeVC: an approach for monitoring and visualizing distributed repositories. *Journal of Software Engineering Research and Development 5* (2017), 1–34.

[11] Cornuéjols, G.; Fonlupt, J.; Naddef, D. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming 33* (1985), 1–27.

[12] da Silva, I. A.; Chen, P. H.; Van der Westhuizen, C.; Ripley, R. M.; van der Hoek, A. Lighthouse: Coordination through emerging design. In *Proceedings of the 2006 OOPSLA Workshop on Eclipse Technology eXchange* (Portland, 2006), ACM, pp. 11–15.

[13] Demaine, E. D.; Zadimoghaddam, M. Minimizing the diameter of a network using shortcut edges. In *Proceedings of the 12th Scandinavian Workshop on Algorithm Theory* (Berlin, 2010), Springer, pp. 420–431.

[14] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271.

[15] Dormann, S.; Deutsch, A. Modeling of self-organized avascular tumor growth with a hybrid cellular automaton. *In Silico Biology 2* (2002), 393–406.

[16] Downey, R. G.; Fellows, M. R. *Parameterized Complexity.* Springer, 1999.

[17] Fleischmann, B. A cutting plane procedure for the travelling salesman problem on road networks. *European Journal of Operational Research 21* (1985), 307–317.

[18] Garey, M. R.; Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, 1979.

[19] Gomes, A.; Voiculescu, I.; Jorge, J.; Wyvill, B.; Galbraith, C. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*, 1st ed. Springer, 2009.

[20] Interian, R.; Moreno, J. R.; Ribeiro, C. C. Polarization reduction by minimum-cardinality balanced edge additions: Formulations, complexity, and integer programming approaches. *Journal of Combinatorial Optimization* (2019). Submitted.

[21] Interian, R.; Otero, J. M.; Ribeiro, C. C.; Montenegro, A. A. Curve and surface fitting by implicit polynomials: Optimum degree finding and heuristic refinement. *Computers & Graphics 67* (2017), 14–23.

[22] Interian, R.; Ribeiro, C. C. A GRASP heuristic using path-relinking and restarts for the Steiner traveling salesman problem. *International Transactions in Operational Research 24* (2017), 1307–1323.

[23] Interian, R.; Ribeiro, C. C. A GRASP with restarts heuristic for the Steiner traveling salesman problem. In *Proceeding of the MIC and MAEB 2017 Conferences* (Barcelona, 2017), pp. 106–115.

[24] Interian, R.; Ribeiro, C. C. An empirical investigation of network polarization. *Applied Mathematics and Computation 339* (2018), 651–662.

[25] Interian, R.; Ribeiro, C. C. An iterated greedy heuristic for the minimum-cardinality balanced edge addition problem. In *Proceeding of the Metaheuristics International Conference 2019* (Cartagena, 2019), pp. 195–198.

[26] Interian, R.; Rodríguez-Ramos, R.; Valdés-Ravelo, F.; Ramírez-Torres, A.; Ribeiro, C. C.; Conci, A. Tumor growth modelling by cellular automata. *Mathematics and Mechanics of Complex Systems 5* (2017), 239–259.

[27] ISO 25000 Official site. ISO25000 Software Product Quality, ISO/IEC 25010. Last access on 2018-07-17. http://iso25000.com/index.php/en/iso-25000-standards/iso-25010.

[28] Kansal, A.; Torquato, S. Emergence of a subpopulation in a computational model of tumor growth. *Journal of Theoretical Biology 207* (2000), 431–441.

[29] Karp, R. M. *Reducibility among Combinatorial Problems*. Springer, Boston, 1972.

[30] Kruskal, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society* (1956), vol. 7, American Mathematical Society, pp. 48–50.

[31] Landrum, A. YouTube as a primary propagator of flat Earth philosophy. In *American Association for the Advancement of Science Annual Meeting* (Washington, 2019). https://aaas.confex.com/aaas/2019/meetingapp.cgi/Session/21893.

[32] Letchford, A. N.; Nasiri, S. D.; Theis, D. O. Compact formulations of the Steiner traveling salesman problem and related problems. *European Journal of Operational Research 228* (2013), 83–92.

[33] Mendoza, I. Combining verification and validation methods to cover software quality characteristics. Master's thesis, Institute of Computing, Universidade Federal Fluminense, Niterói, 2018.

[34] Mendoza, I.; Souza, U.; Kalinowski, M.; Interian, R.; Murta, L. G. P. An efficient algorithm for combining verification and validation methods. In *SOFSEM 2019: Theory and Practice of Computer Science* (Cham, 2019), Springer, pp. 324–340.

[35] Mill, J. S. *On liberty*. J. W. Parker and Son, London, 1859.

[36] Miller, C. E.; Tucker, A. W.; Zemlin, R. A. Integer programming formulation of traveling salesman problems. *Journal of the ACM 7* (1960), 326–329.

[37] New York Times. How we became bitter political enemies. Last access on 2017-11-13. https://www.nytimes.com/2017/06/15/upshot/how-we-became-bitter-political-enemies.html.

[38] Oxford Dictionaries. Definition of polarization. Last access on 2017-09-06. https://en.oxforddictionaries.com/definition/polarization.

[39] Perry, D. E.; Siy, H. P.; Votta, L. G. Parallel changes in large scale software development: An observational case study. In *Proceedings of the 20th International Conference on Software Engineering* (Kyoto, 1998), IEEE Computer Society, pp. 251–260.

[40] Prim, R. C. Shortest connection networks and some generalizations. *The Bell Systems Technical Journal 36* (1957), 1389–1401.

[41] Ramírez-Torres, A.; Valdés-Ravelo, F.; Rodríguez-Ramos, R.; Bravo-Castillero, J.; Guinovart-Díaz, R.; Sabina, F. J. Modeling avascular tumor growth via linear elasticity. In *Proceedings of the 24th International Congress of Theoretical and Applied Mechanics* (Montreal, 2016), pp. 1739–1740.

[42] Rejniak, K.; McCawley, L. Current trends in mathematical modeling of tumor-microenvironment interactions: a survey of tools and applications. *Experimental Biology and Medicine 235* (2010), 411–423.

[43] Resende, M.; Ribeiro, C. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures.* Springer, 2016.

[44] Ruiz, R.; Stützle, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research 177* (2007), 2033–2049.

[45] Taubin, G. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13* (1991), 1115–1138.

[46] Taubin, G. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16* (1994), 287–303.

[47] The Economist. Not persuaded – political polarization. Last access on 2017-11-13. https://www.economist.com/blogs/democracyinamerica/2015/05/political-polarisation.

[48] The New Yorker. Obama reckons with a Trump presidency. Last access on 2016-11-28. http://www.newyorker.com/magazine/2016/11/28/obama-reckons-with-a-trump-presidency.

[49] Zhang, H.; Tong, W.; Xu, Y.; Lin, G. The Steiner traveling salesman problem with online edge blockages. *European Journal of Operational Research 243* (2015), 30–40.

[50] Zhang, H.; Tong, W.; Xu, Y.; Lin, G. The Steiner traveling salesman problem with online advanced edge blockages. *Computers & Operations Research 70* (2016), 26–38.

[51] Zheng, B.; Takamatsu, J.; Ikeuchi, K. An adaptive and stable method for fitting implicit polynomial curves and surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32* (2010), 561–568.

APPENDIX A – An empirical investigation of network polarization

Applied Mathematics and Computation 339 (2018) 651–662

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

An empirical investigation of network polarization

Ruben Interian*, Celso C. Ribeiro

Institute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-346, Brazil

A R T I C L E   I N F O

MSC:
05C78
62F03
91F99

Keywords:
Polarization
Homophily
Network
Graph
p-Value

A B S T R A C T

This paper proposes and explores a new quantitative characterization of the polarization phenomenon in networks. New tools for evaluating the polarization of a network are presented. We first characterize the homophily of each node individually. We depart from the definition of a new measure of the homophily of the nodes of a network and we consider the homophily distribution over the nodes as a primary indicator of the strength of polarization. Next, to address the polarization of the network as a whole, a probabilistic approach is developed. The approach is based on the straightforward computation of empirical cumulative distribution functions of sampled data from the network. These empirical distributions provide a more insightful understanding of the status of the network. They may be used not only to compare the polarization of groups of nodes or entire networks, but also to estimate the impacts of external interventions in terms of node polarization. The usefulness of the approach is illustrated on several case studies associated with real-life data sets from different sources.

1. Introduction

Polarization is a widely known phenomenon that has been discussed by politicians, media, and researchers in recent years [1,2]. This subject has also attracted the attention of thinkers throughout history.

Since the 19th century, Mill, an important philosopher and political theorist, claimed that dialogue across lines of political difference is a key prerequisite for sustaining a democratic citizenry [3]. Arendt also asseverated that debate is irreplaceable for forming enlightened opinions that reach beyond the limits of one's own subjectivity to incorporate the standpoints of others [4]. More recently, several world leaders have often expressed concern about polarization problems caused by social media [5,6]. From sociologists to economists, many are interested in studying the behavior and interactions in social networks that rule the opinion formation process.

According to the Oxford Dictionaries, polarization is the *division into sharply contrasting groups or sets of opinions or beliefs* [7].

It is not a widely accepted fact that social media increase polarization. Although there are many arguments supporting this thesis [8,9], there are also opposing views [10]. However, it is known that social networks and mass media, like newspapers and blogs, are the place where this phenomenon manifests itself in a more strong way. Even if social and mass media do not contribute to increase polarization in modern societies, it is important to identify the mechanisms by which polarization arises, as well as the characteristics and the peculiarities of polarized networks.

* Corresponding author.
E-mail addresses: rinterian@ic.uff.br, rubenus@yandex.ru (R. Interian), celso@ic.uff.br (C.C. Ribeiro).

https://doi.org/10.1016/j.amc.2018.07.066
0096-3003/© 2018 Elsevier Inc. All rights reserved.

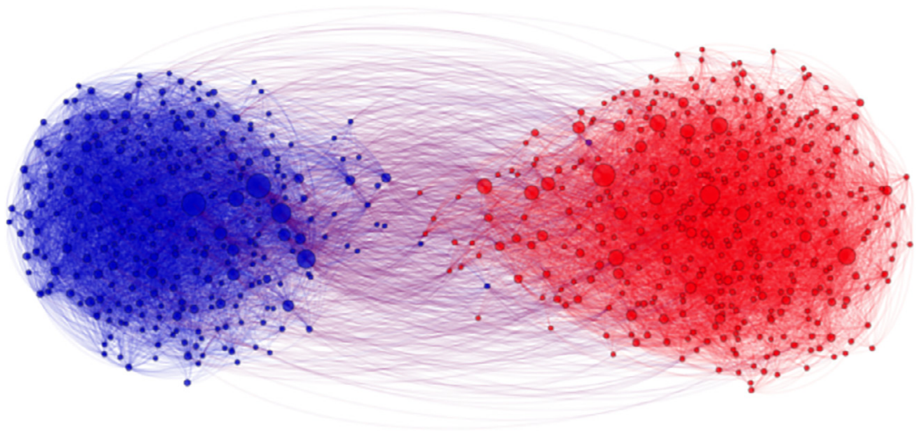


Fig. 1. Network of political blogs during the 2004 U.S. presidential election. Democratic and republican blogs are represented by blue and red circles, respectively. It is clear that this network is strongly polarized, since there are a huge number of edges connecting democratic blogs among themselves, a huge number of edges connecting republican blogs as well, and relatively few edges connecting a democratic blog and a republican blog. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Polarization is closely related to homophily (from Ancient Greek: *homo* = "self" and *philia* = "love", love to oneself), the tendency of individuals to associate with others that are similar to themselves. To avoid misunderstandings, we regard polarization as an extreme expression of homophily.

Homophily, also called assortativity [11,12], has been widely studied by researchers. Several measures and models exist to characterize homophily. However, most of the work in the literature make use of a single numerical measure to define the homophily of some network in a given moment of time. An assortativity coefficient is proposed in [12] and applied to several networks, showing that homophily is a universal phenomenon. The probability of creating a new link between two individuals as a function of their similarity is studied in [13]. The inbreeding homophily measure that reflects the amount of bias towards same-type relationships is mentioned in [14]. The Pearson correlation coefficient is used in [11] as a measure of the preference of high-degree nodes to attach to other high-degree nodes.

The characterization and the properties of polarized networks are very helpful to provide a better understanding of the behavior of individuals and societies. In addition, they can also help policymakers to define policies based on these characteristics. Although quantitative methods are often difficult to use in real-life situations because they tackle models that are abstractions of concrete cases, they are very useful in moderation systems used to detect suspicious events or users, as well as for economists and marketing experts when dealing with social and complex networks [15].

In this work, we develop tools that can be used for evaluating the polarization of a network in a more deep way than that offered by single valued measures. We first characterize the homophily of each node individually and, then, the polarization of the network as a whole. Section 2 introduces some real-life test instances that will be used as case studies. The homophily of each node is defined in Section 3 and we consider the distribution of the homophily values over the nodes of the network as a primary indicator of the strength of polarization, which is analyzed and illustrated with some case studies. The probability of a node to be influenced by homophily is derived in Section 4 as an improved measure of network polarization. This new measure is used to assess the statistical relevance of the homophily value. Section 5 develops a probabilistic approach to compare the polarization of groups of nodes or entire networks. The approach is based on the straightforward computation of empirical cumulative distribution functions of sampled data from the network, which provide a more insightful understanding of the status of the network. The usefulness of the approach is illustrated on several case studies associated with real-life data sets. Concluding remarks are drawn in the last section.

2. Case studies

We consider as case studies a number of test data sets selected from very different sources. We assume that some of them may be polarized to some extent.

- *Books* – A network of books about U.S. politics sold by Amazon.com [16]. Edges between books represent frequent copurchasing of those books by the same buyers. Most of the books are classified as conservative or liberal, and a small number of them as neutral.
- *Blogs* – A network of political blogs that emerged during the 2004 U.S. presidential election [17]. Blogs are divided into two groups: republican and democratic. Fig. 1 represents democratic and republican blogs by blue and red circles, respectively [18].
- *Trade* – A world trade network, built using information about bilateral trade data between countries. The data compiled by Aller et al. [19] was obtained from the United Nations COMTRADE database. We consider that two countries are connected by an edge if the amount of trade between them is at least 5% of the total traded by any two countries. In other words, we discard the edges that do not represent a significant trade for either country. We analyzed two groups

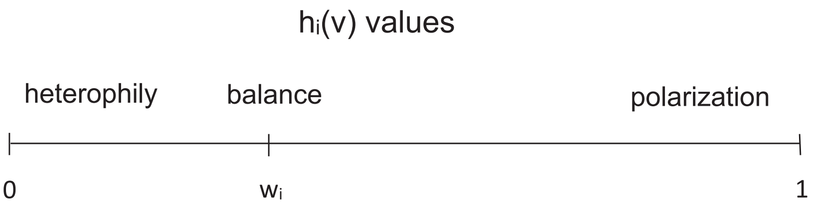


Fig. 2. Range of variation of the homophily value. The extreme case $h_i(v) = 0$ corresponds to the absence of polarization (heterophily). As the homophily value increases it reaches the other extreme case $h_i(v) = 1$, which corresponds to extreme polarization.

of countries: those that formerly belonged to Eastern bloc (full COMECON members in 1990) and those who were part of Western bloc (NATO members in 1990, mainly the same countries that participated in the Marshall Plan). Most of the available data refers to 2010 and we analyze whether there are still strong ties between countries in both groups, about 20 years after the end of the Cold War.

- *Game of Thrones* – A network of interactions between characters of the well known novel “A Song of Ice and Fire”, sometimes called “Game of Thrones” (GoT), by the name of the first book and the television series. A more detailed description of this instance can be found in [20]. Characters are classified according to the house (clan) to which they belong or of which they are vassals. There is also a classification of the characters according to their gender (to see the role of gender in network organization, see [21]).

All data for these four case studies are available in [22].

3. Node homophily

Let $G = (V, A)$ be a directed graph, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes and $A = \{a_1, a_2, \ldots, a_m\} \subseteq V \times V$ is the set of arcs. Moreover, let $\mathcal{A} = \{A_1, A_2, \ldots, A_q\}$ be a set of node groups defined on V, i.e., each $A_i \subseteq V$ for any $i = 1, \ldots, q$. Each node $v \in V$ has an associated state $S(v) = \{i = 1, \ldots, q \ : \ v \in A_i\}$, that reflects the set of groups to which node v belongs.

The groups may be interpreted as communities (as in social networks) or classes defined by some criterion. In case the groups define a partition of the node set V, i.e., $A_1 \cup A_2 \cup \cdots \cup A_q = V$ and $A_i \cap A_j = \emptyset$ for any $i, j = 1, \ldots, q : i \neq j$, each node belongs to one single group and its state is also referred to as its type.

For any node $v \in V$, we denote by $\mathcal{N}(v)$ the set of successors of v in G. We consider that any successor $u \in \mathcal{N}(v)$ wins influence on v (and, therefore, v is influenced by u), in the sense that v receives information from u.

The cardinality of $\mathcal{N}(v)$ is the out-degree $d(v)$ of node v. For any group $A_i \in S(v)$, the successors of v that belong to A_i form the set $\mathcal{N}_i(v) = \mathcal{N}(v) \cap A_i \subset \mathcal{N}(v)$. The cardinality of $\mathcal{N}_i(v)$ is the i-degree $d_i(v)$ of node v. The homophily of any node $v \in V$, with $d(v) > 0$, with respect to any group A_i, for $i = 1, \ldots, q$, is defined by

$$h_i(v) \equiv \frac{d_i(v)}{d(v)}. \tag{1}$$

In other words, the homophily of a node $v \in V$ with respect to a group A_i, for $i = 1, \ldots, q$, is the ratio between the number of successors of v that belong to the same group A_i as v and the number of successors of v. Of course, this definition only makes sense if $d(v) > 0$. The value of the homophily is a real number in the [0,1] interval, where 0 suggests heterophily (preference for the opposite), while 1 indicates extreme homophily. If a node v belongs to only one group, i.e., $|S(v)| = 1$ and $S(v) = \{i\}$ for some $i = 1, \ldots, q$, then the index i can be omitted and we denote $h(v) = h_i(v)$.

The homophily measure $h_i(v)$ is similar to the homophily index H_i defined by Currarini et al. [14], which denotes the average i-degree of the nodes in A_i, divided by the average degree of all nodes in this same group. Following our notation,

$$H_i = \frac{\sum_{j=1}^{|A_i|} d_i(v_j)}{\sum_{j=1}^{|A_i|} d(v_j)}. \tag{2}$$

The newly proposed node homophily measure $h_i(v)$ is more useful to fully describe the polarization of a network than the single value H_i. Therefore, we will use the $h_i(v)$ values for a node-centered homophily analysis of the network.

Let $w_i = |A_i|/|V|$ be the fraction of the nodes of the graph G that belong to group A_i, for $i = 1, \ldots, q$. If $h_i(v)$ is close to w_i for a node $v \in V$, then $d_i(v) \approx w_i d(v)$, i.e., $d_i(v)$ is proportional to the number of nodes in group A_i. This means that if we randomly choose a node u from among the successors of v, then the probability that u belongs to A_i is approximately w_i, which is the same probability that a randomly selected node of the network belongs to A_i. In this case, we say that node v is *balanced*.

In case $d_i(v)$ is much greater than $w_i d(v)$, then the homophily of v is much larger than w_i and there is polarization of v. Finally, if $d_i(v)$ is significantly smaller than $w_i d(v)$, then we are in the presence of heterophily. Fig. 2 illustrates these concepts and the range of variation of the homophily value.

This analysis can be made for each node in the network. Therefore, the distribution of the homophily over the nodes of the network (or over some subset of them, such as one of the groups $A_1, A_2, \ldots, A_p$) can be used to characterize its polarization.

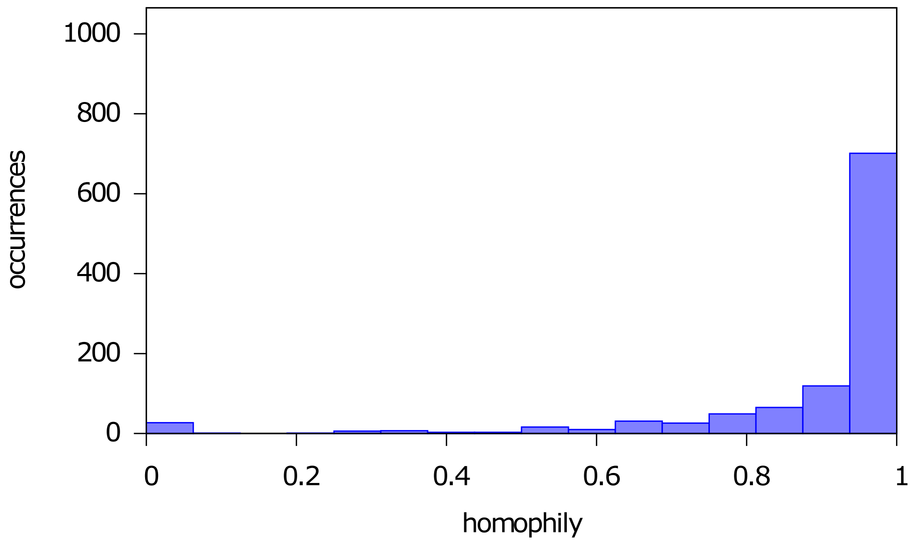


Fig. 3. Histogram representing the distribution of homophily values over the nodes of the political blogs network. The vertical axis indicates the number of occurrences of the homophily values in each interval of the horizontal axis.

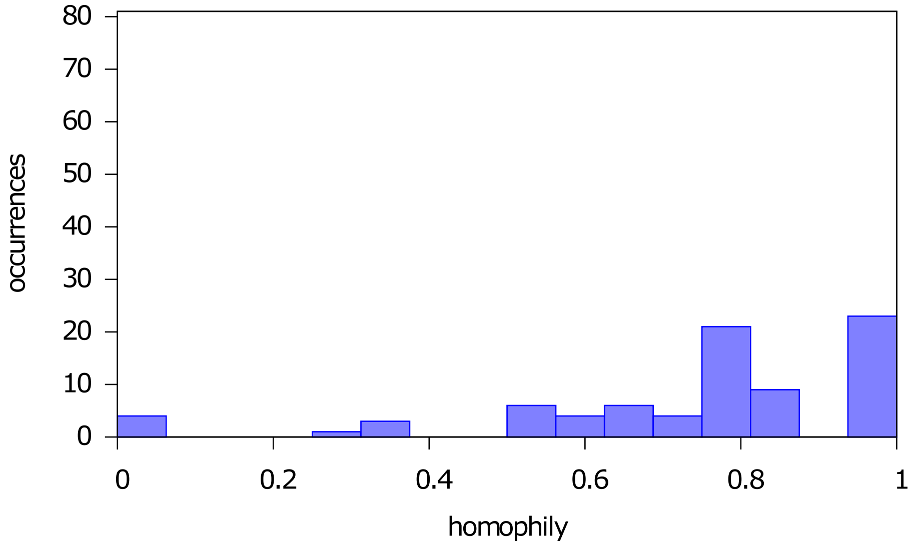


Fig. 4. Histogram that represents the distribution of homophily values over the male character nodes of the Game of Thrones network. The vertical axis indicates the number of occurrences of the homophily values in each interval of the horizontal axis.

The distribution of the homophily values over the nodes of the network can be represented by a histogram, with the frequency of occurrences in each interval displayed in the vertical axis. Fig. 3 shows the case for the political blogs data set [17]. In this network, both groups (republican-affiliated and democrat-affiliated) form a partition of the node set and are of similar size ($w_{republican} = 0.491$ and $w_{democrat} = 0.509$). Homophily values in the histogram are mostly clustered near the value 1, far from 0.5. Therefore, the neighborhoods of a significant number of nodes of this network are mainly formed by same-type nodes, indicating polarization.

Next, as a second example, we consider the case of homophily by gender in the Game of Thrones network. There are two groups in this network, formed by male and female characters. We analyze the homophily values exclusively for the male characters. We are interested in knowing if male characters are more prone to interact with other male characters or not. Fig. 4 illustrates the occurrences of the homophily values in this network. If the network is not polarized, then the homophily values should average approximately $w_{males} = 0.757$, corresponding to the proportion of male characters in the node set V. Although we observe some clusterization of the histogram around this value, there is also a concentration of nodes with a high homophily value leaning towards 1, in the right extremity of the histogram. We abstain for now from assuming any further hypothesis in this case, and we return to this example in the next section.

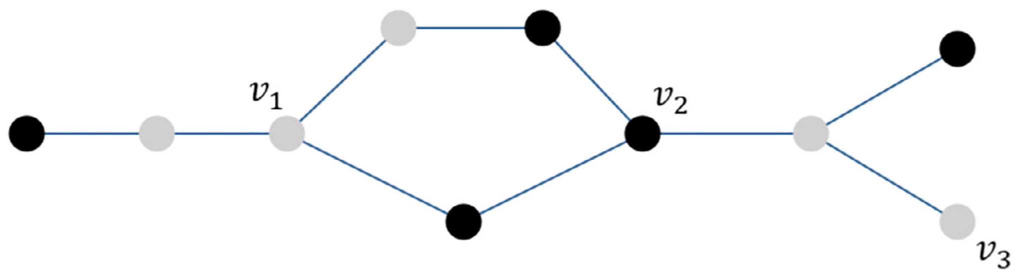


Fig. 5. Example of a graph with five black and five grey nodes for which high values of the homophily measure for some specific nodes do not necessarily indicate polarization. For example, the fact that $h(v_3) = 1$ does not indicate "polarization" of node v_3.

4. Polarization test

We note that although the homophily values are relevant, they may somehow be incomplete. Let us consider the graph in Fig. 5, with five black and five grey nodes. Each edge represents two arcs, one in each direction. The number of arcs between same-type nodes is equal to the number of arcs between different-type nodes. Therefore, this graph is not polarized. However, many nodes of the graph have different numbers of black and grey neighbors. In this example, both nodes v_1 and v_2 have two same-type neighbors and one neighbor of the other type. Therefore, $h(v_1) = h(v_2) = \frac{2}{3}$. Since v_3 has only one neighbor, $h(v_3) = 1$. The homophily value can be high for some nodes simply by chance, either because of their small degrees, or because it is impossible to divide an odd number by two, often lacking a greater meaning.

Therefore, "high" homophily values for some specific nodes do not necessarily indicate polarization.

In order to avoid the misuse of the raw homophily values we propose to use the binomial test to assess the statistical significance of $d_i(v)$ and $d(v)$ values as indicators of the polarization of node v.

Two definitions are relevant to establish whether a network is polarized or not. We assume that the networks considered in this work are such that, whenever a new arc (v, u) originating from node v is added to the graph, there is a probability $p_i \in [0, 1]$ that arc (v, u) is formed by same-type neighbors, where p_i depends only on the group A_i, $i = 1, \ldots, q$, to which node v belongs to. More formally:

Definition 4.1 (Linking probability)**.** Let $G = (V, A)$ be a directed graph with a set $\mathcal{A} = \{A_1, A_2, \ldots, A_q\}$ of node groups defined on V, with $A_i \cap A_j = \emptyset$ for any $i, j = 1, \ldots, q : i \neq j$. We say that $p_i \in [0, 1]$ is the linking probability of group A_i if, for each $v \in A_i$ and for any arc (v, u) that is added to the graph, the probability that node u also belongs to A_i is equal to p_i.

We observe that the linking probability p_i is the same for all nodes in group A_i. The concept of linking probabilities can be illustrated and better understood with some examples. Consider e.g. the case of the political blogs in the internet, associated with some ideological or political groups that establish new hyperlinks reflecting the information sources that exert influence on them. If a blog is biased to some extent, it is more likely (i.e., it happens with higher probability) to establish links with other same-type blogs. If we consider the trade network, we should expect that countries often establish commercial ties with other countries that are ideologically more similar to themselves. Therefore, we made the assumption that, in each case, there is a linking probability of establishing a same-type link from any node of a given group, and this probability is somehow related to the strength of the bias of this group.

Definition 4.2 (Balanced graph)**.** A directed graph $G = (V, A)$ with a set $\mathcal{A} = \{A_1, \ldots, A_q\}$ of node groups defined on V, with $A_i \cap A_j = \emptyset$ for any $i, j = 1, \ldots, q : i \neq j$, is said to be balanced if, for each A_i, the linking probability p_i is equal to w_i, where $w_i = |A_i|/|V|$.

Now, let $G = (V, A)$ be a directed graph with a set $\mathcal{A} = \{A_1, A_2, \ldots, A_q\}$ of node groups defined on V, with $A_i \cap A_j = \emptyset$ for any $i, j = 1, \ldots, q : i \neq j$. Moreover, let $v \in V$ be a node of type $i \in \{1, \ldots, q\}$ with $n = d(v)$ successors and p_i be the linking probability of group A_i, $i = 1, \ldots, q$. Then, the number $d_i(v)$ of same-type successors of v follows a binomial distribution with parameters n and p_i:

$$P(d_i(v) = k) = \binom{n}{k} \cdot p_i^k (1 - p_i)^{n-k}.$$

We observe that if $G = (V, A)$ is a balanced graph, then arcs are formed indistinctly: in other words, the probability that a successor of node $v \in A_i$ be a same-type node is $w_i = |A_i|/|V|$. Therefore, the null hypothesis for the binomial test is

$$H_0 : \quad p_i = w_i.$$

The alternative hypothesis is

$$H_1 : \quad p_i > w_i$$

and, therefore, G is not a balanced graph. Consequently, node v is polarized to some extent.

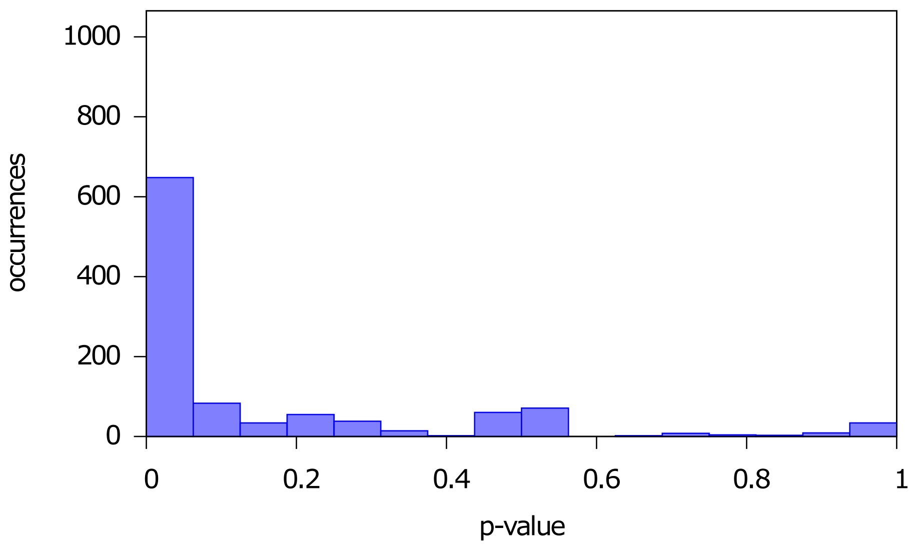


Fig. 6. Histogram that represents the distribution of *p*-values over the nodes of the political blogs network. The vertical axis indicates the number of occurrences of the *p*-values in each interval of the horizontal axis.

We do not consider the case $p_i < w_i$, since in typical networks it is not expected to find a significantly large number of arcs with different-type extremities. Even if for some node we find a disproportionately large number of such inter-group arcs, and a too small number of intra-group arcs, the goal of the *p*-value measure is to evaluate the strength of the polarization of that node. Therefore, we consider that the possible presence of heterophily in some nodes is irrelevant to our analysis. Consequently, the one-tailed binomial test applies in this case.

If we choose some level of significance, such as 5%, we can decide if the null hypothesis is rejected by calculating the probability $p\text{-value}(v)$ of observing a number of same-type successors that is greater than or equal to $d_i(v)$. If $p\text{-value}(v)$ is less than or equal to the significance level, then the null hypothesis should be rejected. If $p\text{-value}(v)$ is greater than the significance level, then the null hypothesis cannot be rejected.

In the example of Fig. 5, the probability of having two or more same-type neighbors among three nodes is 0.5. Therefore, we cannot reject the null hypothesis at the 5% significance level. There is not enough information to conclude that the bias we observe is due to polarization and not to chance. However, if instead there was a node with 30 successors, with 20 of them having its same type, then the *p*-value would be approximately 0.04937 and we could reject the null hypothesis at the 5% level.

Therefore, we can see that two nodes with the same homophily measure 2/3 computed by Eq. (1) can have very different probabilities of being truly influenced by homophily. The homophily measure $h(.)$ shows that some bias is present, but the *p*-value is a stronger measure that indicates how significant the observed bias is.

This is partially explained by the fact that the measure $h(v)$ summarizes two informative variables in a single one, normalizing the number of same-type successors by dividing it by the total number of successors, thus losing usable information.

4.1. Distribution of p-values

Given the null hypothesis, it is well known [23] that the *p*-values are uniformly distributed in the interval [0,1]. Since the number of nodes in real-life networks is finite and some node degrees are small, the *p*-value distribution may be not perfectly uniform. Given their interpretation, the occurrence of small *p*-values indicate that the network is unbalanced. Therefore, a significant concentration of *p*-values towards zero means that the graph is polarized.

Fig. 6 shows the distribution of *p*-values over the nodes of the political blogs data set [17]. The values are mostly clustered close to zero, indicating polarization.

In contrast, Fig. 7 illustrates the distribution of *p*-values for the male characters of the Game of Thrones network, which is much more uniform than the precedent. The complete absence of small *p*-values (lower than any of the commonly used significance levels, such as 5% or even 10%) is striking, meaning that there is no statistically relevant homophily values that justify the rejection of the null hypothesis, therefore indicating the absence of polarization.

4.2. Evaluating the homophily of a network

The distribution of *p*-values summarizes the information about polarization of some group of nodes or network. In addition to the *p*-value histograms, we can present this information by choosing some values for the significance level α and counting the number of nodes (or the fraction of nodes) whose *p*-values are smaller than each value of α. A large fraction

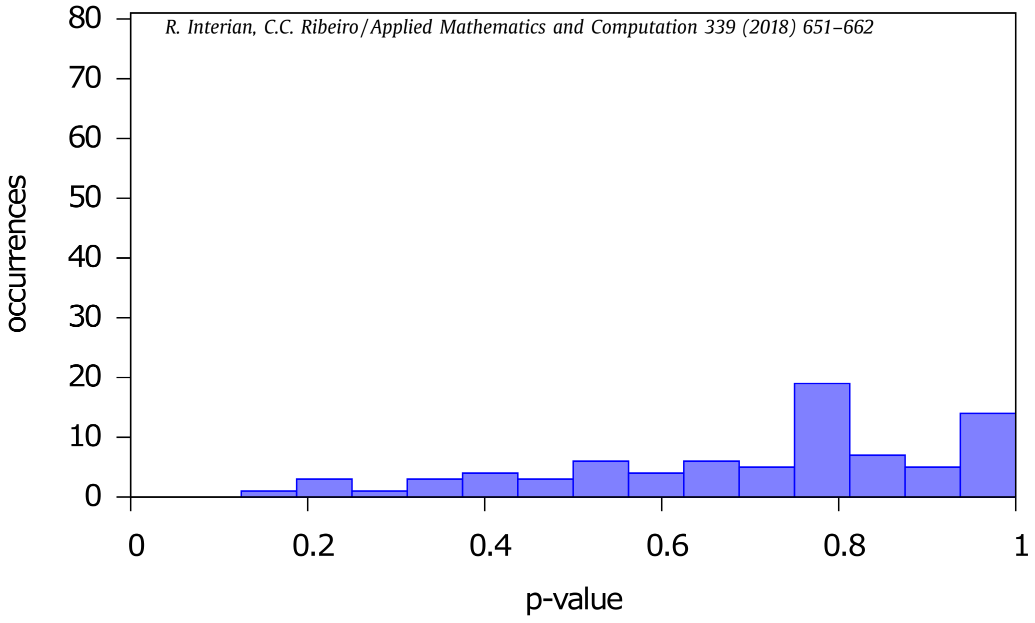


Fig. 7. Histogram that represents the distribution of p-values over the male character nodes of the Game of Thrones network. The vertical axis indicates the number of occurrences of the p-values in each interval of the horizontal axis.

Table 1
Fraction (in percent) of the nodes in each group whose p-value is below the significance level α, corresponding to the more polarized nodes.

Instance name	Group	Significance level, α			
		0.01	0.02	0.05	0.10
Political books	Conservative	40.8	49.0	73.5	73.5
	Liberal	53.5	69.8	81.4	83.7
	Neutral (*)	0.0	0.0	15.4	23.1
	All groups	41.0	51.4	69.5	71.4
Political blogs	Republican	53.8	57.4	62.3	68.6
	Democratic	44.2	48.5	53.2	58.5
	All groups	49.2	53.1	57.9	63.8
World trade	Eastern bloc	60.0	68.0	76.0	80.0
	Western bloc	68.8	75.0	81.3	87.5
Game of Thrones (houses)	Stark-Arryn	25.0	58.3	62.5	79.2
	Lannister	41.7	50.0	54.2	66.7
Game of Thrones (gender)	Male (*)	0.0	0.0	0.0	0.0
	Female (*)	0.0	0.0	0.0	7.7
	All groups (*)	0.0	0.0	0.0	1.9

of nodes with p-values smaller than the significance level α corresponds to a large number of polarized nodes. The higher the former is, the more polarized the network is.

Table 1 shows the fraction of the nodes of the network that satisfy this condition for the most common significance level values 1%, 2%, 5%, and 10%. If the groups cover the entire set of nodes, then every node in V has an associated p-value. In this case, we also show the 'all groups' row, that contains the fraction p-values that are smaller than the significance level α in the whole network.

We observe that, in most cases in this table, a substantial part of the network is polarized. The only exceptions are the groups in the instance corresponding to gender classification in the Game of Thrones network and the small neutral group in the political books network (both marked with an '*'). This means, for the case of the Game of Thrones network, that there is no gender assortativity over the characters, and this can be interpreted as the absence of any sexist pose of the author. Similarly, for the case of the political books network, buyers of neutral books often buy books of different tendencies besides the neutral ones.

5. Comparing polarized groups and networks

We have shown that the p-values are a good indicator of how likely it is to observe polarization and that smaller p-values point to more polarization. In this section, we use the tools proposed in this work to compare the strength of polarization of groups and networks. In order to compare two groups of nodes in terms of their polarization, we make use of the empirical distribution function [24] of the p-values over the nodes.

Table 2
Probability that the p-value of some group X be smaller than the p-value of another group Y.

Instance name	X	Y	$P(p\text{-value}(X) < p\text{-value}(Y))$
Political books	Liberal	Conservative	0.575
	Conservative	Liberal	0.425
Political blogs	Republican	Democratic	0.567
	Democratic	Republican	0.432
World trade	Eastern bloc	Western bloc	0.435
	Western bloc	Eastern bloc	0.565
Game of Thrones (houses)	Stark-Arryn	Lannister	0.536
	Lannister	Stark-Arryn	0.453
Game of Thrones (gender)	Male	Female	0.449
	Female	Male	0.538

The empirical distribution function (CDF) of a sample $X_1, \ldots, X_n$ of a real-valued random variable X is defined as

$$F_n(x) = \frac{1}{n} \cdot \sum_{i=1}^{i=n} 1 \cdot \{X_i \leq x\}, \quad \forall x \geq 0. \tag{3}$$

The empirical distribution function $F_n(x)$ of the random sample of size n is an estimator of the unknown distribution function $F_X(x)$ of the random variable X. This estimator has good statistical properties: it is unbiased and consistent, among other properties [24].

In the following, we explore this concept for comparing the polarization of groups of nodes in the same network, as well as the polarization of two different networks.

5.1. Statistical comparison of p-values

For each pair of groups X and Y of the networks considered in our case studies, Table 2 displays the probability $P(p\text{-value}(X) < p\text{-value}(Y))$ that p-value(x) of a randomly selected node $x \in X$ be smaller than p-value(y) of a randomly selected node $y \in Y$.

To compute $P(p\text{-value}(X) < p\text{-value}(Y))$, we simply divide the number of pairs of nodes (x, y): $x \in X$, $y \in Y$ such that p-value$(x) < p$-value(y) by the total number $|X| \cdot |Y|$ of pairs (x, y): $x \in X$, $y \in Y$. Considering e.g. the case of the political books, let X be the set formed by 43 liberal books and Y be the set of 49 conservative books, amounting to $43 \times 49 = 2107$ (x, y) pairs. Since there are 1212 (x, y) pairs with p-value$(x) < p$-value(y), then $P(p\text{-value}(X) < p\text{-value}(Y)) = \frac{1212}{2107} = 0.575$. Probabilities in the same instance do not necessarily add up to one, since there are pairs (x, y): $x \in X$, $y \in Y$ of nodes with p-value$(x) = p$-value(y).

For any of the instances in Table 2, we observe that $P(p\text{-value}(X) < p\text{-value}(Y)) \neq P(p\text{-value}(Y) < p\text{-value}(X))$. In principle, we could take the fact that $P(p\text{-value}(X) < p\text{-value}(Y)) > P(p\text{-value}(Y) < p\text{-value}(X))$ as an indication that group X is more polarized than group Y. However, we do not know a priori whether this lack of symmetry is sufficient or conclusive to compare the strength of polarization of the two groups X and Y. This issue will be addressed in the next section.

5.2. Comparing the polarization of two groups

In this section, we present and discuss the empirical distribution functions of the p-values associated to the groups in which some of our data sets are clustered.

Fig. 8 displays the p-value empirical CDFs (3) for the nodes of the groups representing conservative and liberal books. We observe that the conservative books p-value CDF first-order stochastically dominates [25] the liberal books p-value CDF, i.e., conservative books are less likely to be polarized. However, both groups are extremely polarized, since more than 70% of the p-values are below the 0.1 threshold. In fact, the CDF at 0.1 is approximately 0.735 for the conservative books and approximately 0.837 for the liberal books.

Similarly, Fig. 9 shows the p-value empirical CDFs for the nodes of the groups representing democratic and republican political blogs. In this case, the democratic blogs appear to be somewhat less polarized than the republican blogs although, once again, both are strongly polarized.

Fig. 10 compares the p-value CDFs of the two largest clans in the Game of Thrones network. There is no first-order stochastic dominance between them. However, most of the Lannister p-value CDF is below the Stark-Arryn p-value CDF, indicating more polarization on the Stark-Arryn clan, as already indicated by Table 2.

Finally, Fig. 11 presents the p-value CDFs for the gender groups in the Game of Thrones network. Although the values $P(p\text{-value}(X) < p\text{-value}(Y))$ and $P(p\text{-value}(Y) < p\text{-value}(X))$ in Table 2 are different for this instance, we do not observe first-order stochastic dominance. Instead, we observe that the plots are much closer to uniform distributions (corresponding

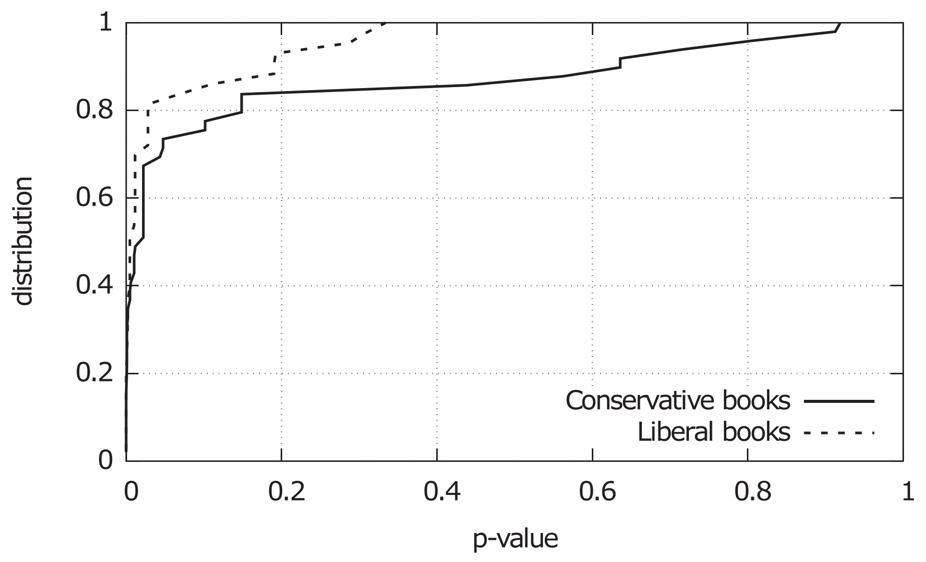


Fig. 8. Polarization in the network of conservative and liberal books: comparison of the p-value cumulative distribution functions. Both groups are extremely polarized, although the conservative books group less likely to be polarized.

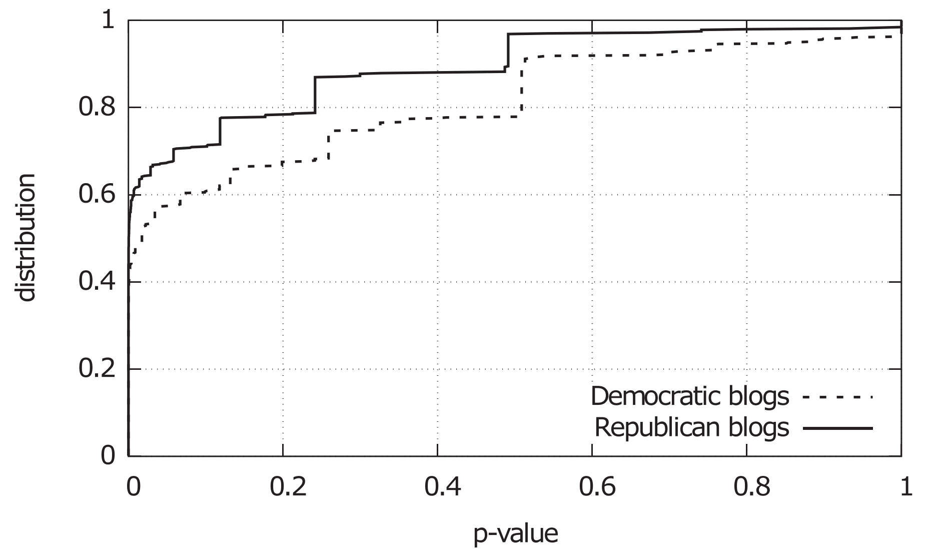


Fig. 9. Polarization in the network of democratic and republican blogs: comparison of the p-value cumulative distribution functions. Both groups are extremely polarized, although the democratic blogs appear to be somewhat less polarized than the republican blogs.

to straight lines through the origin) and the p-vales are greater than those in the previous figures, corresponding to the expected absence of polarization.

5.3. World trade case study

It is well established that there was very little trade between countries in the West and East countries before the nineties. We can use the tools proposed in this work to analyze how things have changed ever since.

We summarize in tabular form some relevant p-value information for the World trade network. We use this particular representation to explicitly represent p-values in order to show the five most open and the five least independent economies in 2010, with regard to these groups. Table 3 presents the results.

The results indicate that, in 2010, Russia and Ukraine remained commercially intertwined with other former Soviet Union states. Cuba and Vietnam, which were on the periphery of the Eastern bloc, no longer have any relevant commercial links with other former COMECON states. In the Western bloc, United States and Canada have more diversified economies, in the sense that they do not depend on other countries from the same group.

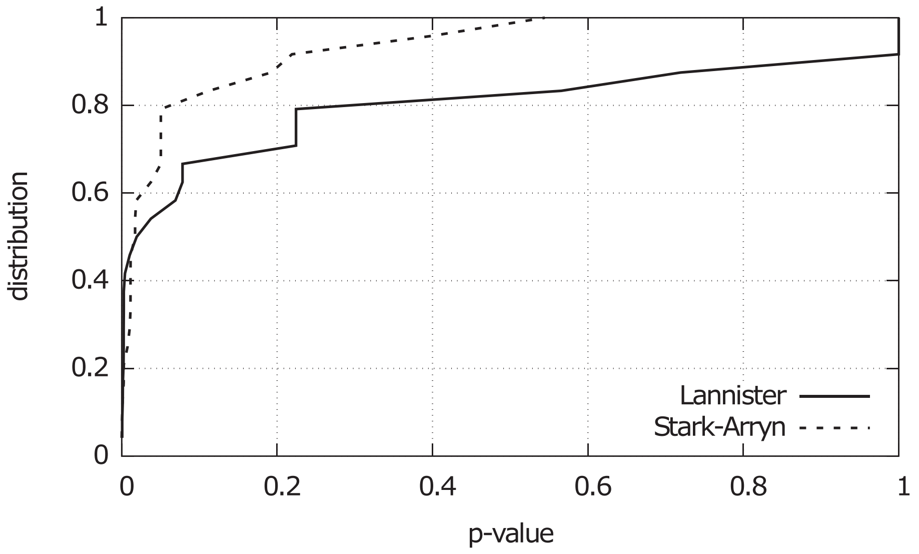


Fig. 10. Polarization in the Lannister and Stark-Arryn clans in the Game of Thrones network: comparison of the *p*-value cumulative distribution functions. Most of the Lannister *p*-value CDF is below the Stark-Arryn *p*-value CDF, indicating more polarization on the Stark-Arryn clan.

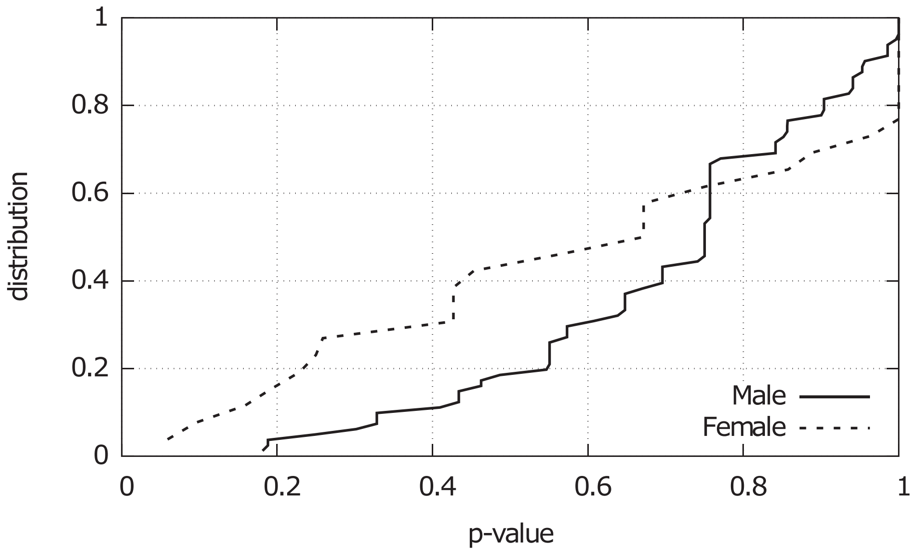


Fig. 11. Polarization by gender in the Game of Thrones network: comparison of the *p*-value cumulative distribution functions. There is no first-order stochastic dominance between the two plots, that are much closer to uniform distributions, corresponding to the expected absence of polarization.

5.4. Comparing the polarization of two networks

In the same way as we compared the polarization of two groups of nodes in Section 5.2, in some situations it is possible to compare the polarization of two different networks. In this case, it is required that the groups define a partition of the node set corresponding to each network: every node should belong to exactly one group and the *p*-value CDFs can be computed for each entire network.

The usefulness of this type of comparison can be illustrated with practical and realistic applications in two scenarios. First, consider a social network that is subject to some intervention (such as a marketing campaign) or, alternatively, severely modified by some external circumstances. If complete information on the network before and after the intervention (or external circumstance) is available, the *p*-value CDF provide a tool to compare the polarization of the networks representing the two states of the same physical entity, before and after the intervention (or external circumstance) and to evaluate its impacts. A second interesting scenario of application involves the comparison of the polarization of two different networks that represent collective behavior of the same group of people, such as two different social networks in the same country at the same time.

To illustrate this application, once again we consider the political books data set. Its node set V is partitioned into three groups: $A_{conservative}$, $A_{liberal}$, and $A_{neutral}$. Their average different-type out-degrees are, respectively, $\overline{d}_{conservative} = 0.94$,

Table 3
Openness of economies in terms of their p-values. The table presents the countries with the five largest and the five lowest p-values for each bloc.

Bloc	Feature	Country	p-value
East	Smaller p-values	Russia	3.681×10^{-11}
		Ukraine	1.459×10^{-6}
		Lithuania	1.106×10^{-5}
		Moldova	6.073×10^{-5}
		Latvia	1.084×10^{-4}
	Larger p-values	Turkmenistan	0.118
		Azerbaijan	0.155
		Mongolia	0.194
		Cuba	1.000
		Vietnam	1.000
West	Smaller p-values	Iceland	5.126×10^{-7}
		Norway	5.483×10^{-6}
		Germany	4.514×10^{-5}
		Luxembourg	8.619×10^{-5}
		Denmark	2.265×10^{-4}
	Larger p-values	Greece	0.017
		Turkey	0.022
		Italy	0.059
		Canada	0.491
		USA	0.563

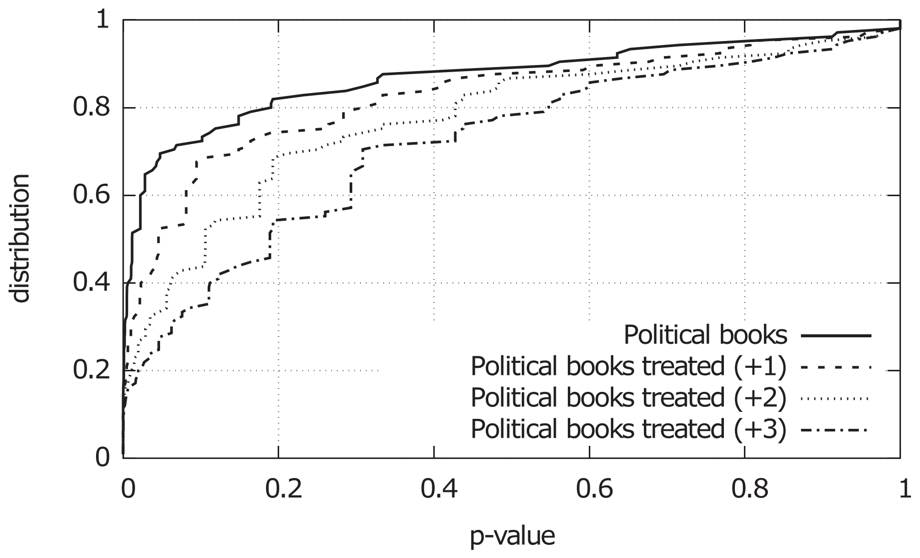


Fig. 12. Comparison of the p-value cumulative distribution functions of the original and treated political books instances with three levels of intervention. The strength of polarization decreases as the intervention level increases.

$\bar{d}_{liberal} = 0.84$, and $\bar{d}_{neutral} = 4.5$, with the overall average of different-type out-degrees being 1.30 and the average node degree being 8.40. This instance was previously shown to be strongly polarized in Section 5.2.

We assume now that this network is subject to some intervention and additional arcs connecting different-type nodes are inserted. Fig. 12 displays the p-value CDFs of the results observed by the simulation of three levels of intervention: with a low-level intervention, one new arc connecting each node to a randomly selected different-type node is added to the network. In the case of medium- or high-level interventions, the network is treated by the addition of two or three new arcs in each case, respectively.

We observe that as the level of the intervention increases and more arcs connecting nodes of different groups are added, the p-values CDF quickly becomes more uniform and the network becomes balanced, eliminating polarization. Considering e.g. the medium level intervention, the overall average of different-type out-degrees raises to 3.30 and the average node degree to 10.40. Since the fraction $2/10.40 \approx 19.2\%$ of new arcs in the resulting network is relatively low, we observe that a medium-level intervention might be sufficient to eliminate polarization. These results show that the p-value CDF is a powerful tool not only to evaluate the polarization of actual networks, but also the impact of interventions.

6. Discussion and concluding remarks

In this study, we proposed and explored a new quantitative characterization of the polarization phenomenon in networks. First, we defined the homophily of the node of a network and we analyzed the distribution of the homophily in a directed graph as a primary indicator of polarization.

Next, we used a probabilistic approach to define a new and improved polarization measure, which is based on the calculation, for each node, of the probability (p-value) of observing a number of same-type successors that is greater than or equal to the actual number of same-type successors observed for this node. We used the distribution of the obtained p-values for evaluating the strength of polarization of the network. The empirical distribution function of the p-values can be used to compare, in a more informative way, the polarization of different groups of nodes and even of entire networks. Several real-life networks from different sources have been used as case studies to illustrate the usefulness of the proposed tools.

The approach proposed in this work is generic and may be applied to a variety of real networks and situations. In particular, the p-value distributions can be used to estimate the impacts of interventions in the polarization of the nodes of a network. Also, the tool developed in this work could be applied into the identification of influential spreaders in networks [15] and for the promotion of collective cooperation within a regular cooperation network [26].

We consider that these tools are relevant in a world characterized by extreme political and ideological polarization.

Competing interests

The authors declare no competing interests.

Acknowledgments

Work of Ruben Interian was sponsored by a CAPES doctorate scholarship. Work of Celso C. Ribeiro was partially supported by CNPq research grants 303958/2015-4 and 425778/2016-9 and by FAPERJ research grant E-26/202.854/2017. We are also grateful for the constructive remarks of one anonymous referee.

References

[1] New York Times, How we Became Bitter Political Enemies, 2017, https://www.nytimes.com/2017/06/15/upshot/how-we-became-bitter-political-enemies.html, last access on 2017-11-13.
[2] The Economist, Not Persuaded – Political Polarization, 2017, https://www.economist.com/blogs/democracyinamerica/2015/05/political-polarisation, last access on 2017-11-13.
[3] J.S. Mill, On Liberty, J. W. Parker and Son, London, 1859.
[4] H. Arendt, Between Past and Future, Viking Press, New York, 1968.
[5] B. News, Angela Merkel: The Big Internet Platforms Have Become an Eye of a Needle, 2017, http://www.bbc.com/news/technology-37798762, last access on 2017-09-26.
[6] T.N. Yorker, Obama Reckons with a Trump Presidency, 2016, http://www.newyorker.com/magazine/2016/11/28/obama-reckons-with-a-trump-presidency, last access on 2017-06-01.
[7] Oxford Dictionaries, Definition of Polarization, 2017, https://en.oxforddictionaries.com/definition/polarization, last access on 2017-09-06.
[8] V. Almeida, D. Doneda, Mecanismos Invisíveis de Polarização Política, 2016, http://www.valor.com.br/opiniao/4768549/mecanismos-invisiveis-de-polarizacao-politica, last access on 2017-11-13.
[9] S. Hong, S.H. Kim, Political polarization on twitter: implications for the use of social media in digital governments, Gov. Inf. Q. 33 (4) (2016) 777–782.
[10] P. Barberá, How social media reduces mass political polarization. Evidence from Germany, Spain, and the U.S., in: Proceedings of the 2015 American Political Science Association Annual Meeting, San Francisco, 2015.
[11] M.E.J. Newman, Assortative mixing in networks, Phys. Rev. Lett. 89 (20) (2002) 208701.
[12] M.E.J. Newman, Mixing patterns in networks, Phys. Rev. E 67 (2) (2003) 026126.
[13] G. Kossinets, D.J. Watts, Empirical analysis of an evolving social network, Science 311 (2006) 88–90.
[14] S. Currarini, M.O. Jackson, P. Pin, An economic model of friendship: homophily, minorities, and segregation, Econometrica 77 (2009) 1003–1045.
[15] C. Li, L. Wang, S. Sun, C. Xia, Identification of influential spreaders based on classified neighbors in real-world complex networks, Appl. Math. Comput. 320 (2018) 512–523.
[16] M.E.J. Newman, Network Data – Books About US Politics, 2017, http://www-personal.umich.edu/~mejn/netdata/, last access on 2017-12-11.
[17] L.A. Adamic, N. Glance, The political blogosphere and the 2004 U.S. election: divided they blog, in: Proceedings of the Third International Workshop on Link Discovery, ACM, New York, 2005, pp. 36–43.
[18] C. Jones, Visualizing Polarization in Political Blogs, 2018, http://allthingsgraphed.com/2014/10/09/visualizing-political-polarization/, last access on 2018-01-13.
[19] C. Aller, L. Ductor, M.J. Herrerias, The world trade network and the environment, Energy Econ. 52 (2015) 55–68.
[20] A. Beveridge, J. Shan, Network of thrones, Math Horiz. Mag. 23 (4) (2016) 18–22.
[21] I. Psylla, P. Sapiezynski, E. Mones, S. Lehmann, The role of gender in social network organization, PLOS One 12 (12) (2017) 1–21.
[22] R. Interian, C.C. Ribeiro, Data for Case Studies in 'An Empirical Investigation of Network Polarization', 2017, https://data.mendeley.com/datasets/xxwtvd92pb/1. 10.17632/xxwtvd92pb.1
[23] C. Blocker, J. Conway, L. Demortier, J. Heinrich, T. Junk, L. Lyons, G. Punzig, Simple facts about p-values, Technical Report, Laboratory of Experimental High Energy Physics, Rockefeller University, New York, 2006.
[24] A.W. van der Vaart, Asymptotic Statistics, first, Cambridge University Press, 1998.
[25] J. Hadar, W.R. Russell, Rules for ordering uncertain prospects, Am. Econ. Rev. 59 (1969) 25–34.
[26] C. Xia, S. Ding, C. Wang, J. Wang, Z. Chen, Risk analysis and enhancement of cooperation yielded by the individual reputation in the spatial public goods game, IEEE Syst. J. 11 (2017) 1516–1525.

APPENDIX B – Polarization reduction by minimum-cardinality balanced edge additions: Formulations, complexity, and integer programming approaches

Journal of Combinatorial Optimization manuscript No.
(will be inserted by the editor)

Polarization reduction by minimum-cardinality balanced edge additions:

Formulations, complexity, and integer programming approaches

Ruben Interian · Jorge R. Moreno · Celso C. Ribeiro

Received: date / Accepted: date

Abstract Real-world networks are often extremely polarized, because the communication between different groups of vertices can be weak and, most of the time, only vertices within the same groups or sharing the same beliefs communicate to each other. In this work, we introduce the Minimum-Cardinality Balanced Edge Addition Problem (MinCBEAP) as a strategy for reducing polarization in real-world networks based on a principle of minimum external interventions. We present the problem formulation and discuss its complexity, showing that its decision version is NP-complete. We also propose three integer linear programming formulations for the problem and discuss computational results on artificially generated and real-life instances. Randomly generated instances with up to 1000 vertices are solved to optimality. On the real-life instances, we show that polarization can be reduced to the desired threshold with the addition of a few edges. The minimum intervention principle and the methods developed in this work are shown to constitute an effective strategy for tackling polarization issues in practice in social, interaction, and commu-

Work of Ruben Interian and Jorge R. Moreno was supported by CNPq scholarships. Work of Celso C. Ribeiro was partially supported by CNPq research grant 303958/2015-4 and by FAPERJ research grant E-26/201.198/2014, and concluded during a visit to the State Key Laboratory of Scientific and Engineering Computing of the Chinese Academy of Sciences, China. This work was also partially sponsored by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001.

R. Interian
Universidade Federal Fluminense, Institute of Computing, Niterói, RJ 24271-270, Brasil.
E-mail: rinterian@ic.uff.br

J. R. Moreno
Universidade Federal Fluminense, Institute of Computing, Niterói, RJ 24271-270, Brasil.
E-mail: jmoreno@ic.uff.br

C. C. Ribeiro
Universidade Federal Fluminense, Institute of Computing, Niterói, RJ 24271-270, Brasil.
E-mail: celso@ic.uff.br

nication networks, which is a relevant problem in a world characterized by extreme political and ideological polarization.

Keywords Polarization · Minimum-cardinality balanced edge addition problem · Polarized networks · Complexity · Integer programming

1 Motivation

The issue of polarization has been discussed by politicians, media, and researchers [8,25]. This subject has also attracted the attention of thinkers throughout history. John Stuart Mill, an important philosopher and political theorist, claimed that dialogue across lines of political difference is a key prerequisite for sustaining a democratic citizenry [16]. Hannah Arendt also asseverated that debate is irreplaceable for forming enlightened opinions that reach beyond the limits of one's own subjectivity to incorporate the standpoints of others [4]. From sociologists to economists, many are interested in studying the behavior and interactions in social networks that rule the opinion formation process.

According to the Oxford Dictionaries, polarization is the division into sharply contrasting groups or sets of opinions or beliefs [7]. Academic articles, newspapers, and the media in general constantly report the growth of fake news, misinformation spreading, and polarization of increasingly isolated groups of individuals. These phenomena are closely interrelated with each other. Fake news spread faster in polarized networks or groups [24]. At the same time, fake and tendentious news can accentuate polarization within already existing echo chambers in the social networks.

Recently, the causes of the proliferation of flat-earth believers, i.e., people who believe that the Earth is actually flat, were investigated by Landrum [14], revealing the role of the video-sharing platform YouTube on this proliferation. This work showed that the algorithms the platform uses to guide people to topics that might interest them makes it easier for a user to end up in a misinformation echo chamber. The study concludes that the most effective instrument to combat disinformation – i.e., false information spread deliberately to deceive – is to provide (or even "to flood") users of the platform with quality information, to ensure that the public also receives accurate, scientific or simply plural information when watching videos on some subject.

Interian and Ribeiro [13] have shown that many case-study real-world networks are extremely polarized. A polarized network is divided into two or more strongly connected groups, with few edges between vertices belonging to different groups. Communication between different groups is weak: there are many vertices for which all or most of its neighbors belong to the same group. In practice, this corresponds to a situation where, most of the time, only same-group vertices communicate to each other and most of the information that a vertex can receive comes from inside the same group to which it belongs. These groups may correspond to large cliques or quasi-cliques [1,2,22,23,26,27].

Interian and Ribeiro also showed in [13] that in order to reduce polarization, networks can be treated by external interventions. An intervention can be seen as any externally-induced process that modifies the structure of the network, such as a fact-checking campaign, a marketing campaign, a regulatory action or some direct manipulation that adds or removes vertices or edges of the network. The process of adding new vertices is often difficult to be performed in real networks. On the other hand, removing vertices or edges may be controversial, because it can be interpreted as the permanent exclusion or deletion of elements such as users, sites, or posts from a social network. This kind of intervention has been widely used in moderation systems for inspecting or removing objectionable content at the discretion of the moderator and such exclusions are often seen as aggressions against freedom of expression in the digital environment.

Suppose that we have a network formed by a set of vertices V partitioned into disjoint subsets $V_1, V_2, \ldots, V_k$. Two vertices that belongs to the same subset V_i are called same-type vertices, while two vertices that belong to different subsets V_i and V_j, $i \neq j$, are called different-type vertices. We consider the addition of edges between different-type vertices of the network as a less invasive treatment method. A typical example of the use of this kind of treatment in real networks is the suggestion of new friendship relations in social networks. By adding edges between vertices of different groups, a super-graph containing the original graph is built. There are more connections between different-type vertices inside this super-graph and, consequently, inter-group communication is improved.

A new optimization problem addressing the issue of polarization reduction by edge additions is presented in this work. Other optimization problems have already used the idea of adding edges to a graph with the goal of improving specific performance measures. Constant-factor approximation algorithms were developed in [6] for the problem of adding k shortcut edges to the graph in order to minimize its diameter. A game that models the creation of a network by selfish agents that benefit from shortest paths to all destinations is analyzed in [9], considering that the agents pay for the links they establish. Two variants of the diameter minimization problem are studied in [15]: the minimum-cardinality-bounded-diameter and the Bounded-cardinality-minimum-diameter edge addition problems, where it is shown that both problems are NP-hard even if the value of the diameter is fixed to 2. Improved approximation ratios of $O(\log n)$ and 2 were proposed in [5] for both problems, respectively. Some results were also extended to the edge-weighted versions of the problems.

Other works in the area of analysis of social networks explored the idea of adding edges to a graph in order to improve its ability to disseminate information. A problem addressing the minimization of the average shortest path distance between all pairs of vertices was studied in [20], adding a limited number of additional "ghost edges" with the objective of improving the network efficiency of information propagation. This approach prioritizes the shortest path distance between each pair of vertices, while in the present work the connectivity between groups of vertices that represent different opinions, ideas, or beliefs is analyzed.

A measure called characteristic path length was minimized in [19]. The characteristic path length is another name for the average shortest path distance between all pairs of vertices. Some properties of the problem are proved and methods for computing the utility of all candidate edges in large graphs are described and evaluated.

Another edge recommendation problem was introduced in [11]. In this case, the goal of the recommendation is to reduce the "controversy score" of the graph, using a metric based on random walks. The controversy score relies on how controversial a topic is or, in other words, on how much polarization it generates. The probability of acceptance of the recommended edge is also evaluated.

In this article, we propose the minimal intervention principle, which consists in assuming that the lowest number of changes should be made in the original network in order to attend any proposed condition for polarization reduction. We formulate the minimum-cardinality balanced edge addition problem and discuss integer programming formulations for its solution. This work is organized as follows. In the next section, we present the problem formulation and its complexity. Integer programming models are presented in Section 3. Computational results on randomly generated and real-life instances are discussed in Section 4. Concluding remarks are drawn in the last Section 5.

2 Problem formulation and complexity

Let $G = (V, E)$ be a graph defined by a set $V = \{v_1, \ldots, v_n\}$ of vertices and a set $E \subseteq V \times V$ of edges. We use the term *group* to refer to any subset of the vertex set V.

We assume that graph G is polarized to some extent and it is necessary to solve the issue of insufficient communication between the groups. The reduction of the polarization in a polarized graph can be treated and formulated as a mathematical optimization problem is discussed next.

2.1 Minimum-cardinality balanced edge addition problem

Interian and Ribeiro [13] observed that, in many graphs, there may be an important number of vertices that are not connected to other groups, i.e., there may be only intra-group edges adjacent to these vertices. Consider, for example, a network of books about U.S. politics sold by Amazon.com [18]. Edges between books represent frequent copurchasing of those books by the same buyers. Most of the books are classified as conservative or liberal, and a small number of them as neutral. There are 105 vertices in this instance and 56 of them are adjacent only to neighbors of the same group, as shown in Figure 1. Another example is that of a network of political blogs that emerged during the 2004 U.S. presidential election [3]. Blogs are divided into two groups:

republican and democratic. Among the 1065 non-isolated vertices in this instance, there are 572 blogs with links exclusively to blogs of the same political orientation, as shown in Figure 2.

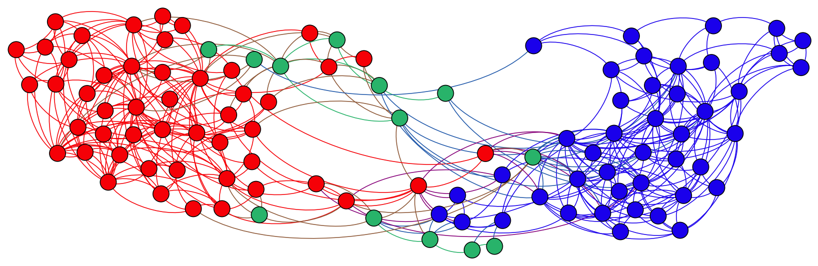


Fig. 1: Network of books about U.S. politics sold by Amazon.com. Red, green, and blue vertices represent, respectively, conservative, neutral and liberal books.

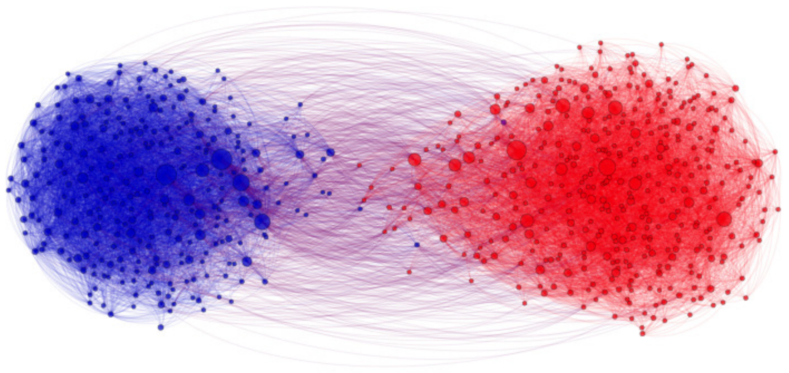


Fig. 2: Network of political blogs during the 2004 U.S. presidential election. Red and blue vertices represent republican and democratic blogs, respectively.

In practice, it can be unrealistic to add an expressive number of edges for each vertex, since this kind of intervention should be minimal. We refer to this assumption as the minimal intervention principle. These statement led us to consider the following optimization problem, in which we seek to minimize the number of edges to be added to a polarized graph in order that any vertex in a proper vertex subset $A \subset V$ can reach some vertex of $V \setminus A$ in the resulting graph by a path with a limited number of edges. If we denote by $d_G(v, V')$ the minimum number of edges in a path from a vertex v of graph G to any vertex in $V' \subseteq V$, then this problem can be formulated as:

Minimum-cardinality Balanced Edge Addition Problem (MinCBEAP)
Instance: Graph $G = (V, E)$, subset $A \subset V$, integer D.
Goal: Find a minimum-cardinality set $E' \subseteq (V \times V) \setminus E$ such that $d_{G'=(V,E\cup E')}(v, V \setminus A) \leq D,\ \forall v \in A$.

Given an integer L as an additional parameter, the decision version of MinCBEAP amounts to the question: "Is there a set $E' \subseteq (V \times V) \setminus E$ with at most L edges such that $d_{G'=(V,E\cup E')}(v, V \setminus A) \leq D,\ \forall v \in A$?"

To prove that MinCBEAP is NP-complete, we first define the eccentricity $\epsilon(v)$ of a vertex $v \in V$ as the longest of the shortest paths in G from v to all other vertices in V [12].

Bearing this definition in mind, we introduce the Minimum-cardinality-bounded-eccentricity edge addition problem [6] (MCBE), which consists in reducing the eccentricity of some vertex v by adding edges to the graph the vertex belongs. More formally, its decision version can be stated as:

Minimum-cardinality-bounded-eccentricity edge addition problem (MCBE)
Instance: Graph $G = (V, E)$, source vertex $s \in V$, integer p, integer B.
Question: Is there a supergraph $G' = (V, E \cup E')$ of G with $E' \subseteq (V \times V) \setminus E$ such that $|E'| \leq p$ and $\epsilon_{G'}(s) \leq B$?

Lemma 1 *There is a concise certificate for MCBE with all edges incident to vertex s.*

Proof Let E' be any concise certificate for MCBE. Consider the shortest-path tree T in graph $G' = (V, E \cup E')$ rooted at s. Each edge in the tree is traversed in the direction of the shortest path to s. Any edge (u, v) in $E' \cap T$ used in this direction can be replaced by edge (u, s), since all vertices that use edge (u, v) in their shortest paths to s will not have their their distance to s increased, therefore creating a new concise certificate with all edges incident to the source vertex s.

Although the NP-completeness of MCBE has been suggested by some authors [6][21], to the best of our knowledge a formal proof does not exist. We give a proof using a polynomial reduction from the set covering problem [10]:

Set covering problem (SC)
Instance: Collection $C = \{S_1, \ldots, S_m\}$ of subsets of a finite set $S = \{x_1, \ldots, x_n\}$, integer k.

Question: Is there a cover $C' \subseteq C$ such that each element of S belongs to at least one member of C' and $|C'| \leq k$?

Theorem 1 *MCBE is NP-complete.*

Proof MCBE is in NP, since for any of its instances defined by a graph $G = (V, E)$, a source vertex $s \in V$, and integers p and B, the eccentricity of the source vertex s in a supergraph $G' = (V, E \cup E')$ of G can be calculated in polynomial time, where $E' \subseteq (V \times V) \setminus E$.

We show that any instance of set covering problem can be transformed into an instance of MCBE with $B = 2$. Consider an instance of the set covering problem defined by subsets $S_1, \ldots, S_m$, with $|S_1 \cup \ldots \cup S_m| = n$, and by an integer k that indicates the size of the target cover C'. Build an instance of MCBE as follows. Let G be a graph with vertex set $V = \{u_1, \ldots, u_m, v_1, \ldots, v_n, s, s'\}$. There is an edge between vertices u_j and v_i if element x_i belongs to S_j. Vertices s and s' are connected and vertex s' is connected with vertices $u_1, \ldots, u_m$. In addition, set $B = 2$ and $p = k$.

Figure 3 illustrates an example of the construction of an instance of MCBE with $B = 2$ and $p = 3$. Note that $\epsilon(s) = 3$ and let $E' \subseteq (V \times V) \setminus E$ be a set with at most p edges such that $\epsilon(s) \leq 2$ in $G' = (V, E \cup E')$, i.e., E' is a concise certificate for MCBE for this instance.

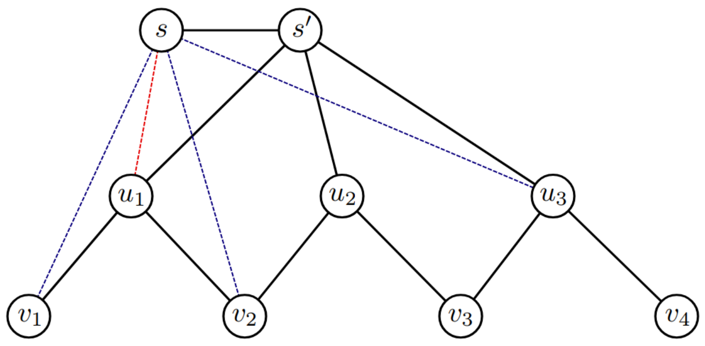


Fig. 3: Example of instance used in the proof of the NP-completeness of MCBE. The concise certificate $E' = \{(s, v_1), (s, v_2), (s, u_3)\}$, highlighted in blue, is replaced by the certificate $\overline{E} = \{(s, u_1), (s, u_3)\}$, with edges (s, v_1) and (s, v_2) replaced by edge (s, u_1).

The distance from vertex s to any vertex $v_1, \ldots, v_n$ in G is greater than 2. From Lemma 1, without loss of generality, we may pick the certificate E' in such a way that all its edges are incident to s. The other extremities of the edges in E' necessarily belong to either $\{v_1, \ldots, v_n\}$ or $\{u_1, \ldots, u_m\}$.

To build another set $\overline{E}$ with at most p edges such that all of them are incident to $\{u_1, \ldots, u_m\}$, we replace every edge $(s, v_i), i = 1, \ldots, n$, in E' by an edge (s, u_j) in $\overline{E}$, with $j : x_i \in S_j$. $\overline{E}$ remains a concise certificate for MCBE, because the distance from s to vertex v_i in $\overline{G} = (V, E \cup \overline{E})$ is still less than 3 for any $i = 1, \ldots, n$ for which there is an edge $(s, v_i) \in E'$. Therefore, $\epsilon(s)$ in $\overline{G}$ is also at most 2.

To conclude, we note that for each vertex v_i there is a vertex u_j such that there is an edge in $\overline{E}$ from s to u_j, because v_i is at most at distance 2 from s in $\overline{G}$. In consequence, the edges in $\overline{E}$ are incident to at most k vertices, each one associated with a set S_j. These k sets represent a concise certificate for the set covering instance.

In order to prove the NP-completeness of MinCBEAP, a polynomial transformation from MCBE is used:

Theorem 2 *MinCBEAP is NP-complete.*

Proof The problem is in NP, since the distance from any vertex $v \in A$ to any vertex in $V \setminus A$ can be calculated in polynomial time.

Now, consider an instance of MCBE defined by graph G, vertex s and integers p and B, and build an instance of MinCBEAP by setting $A = V \setminus \{s\}$ as the proper vertex subset of V. Then, $V \setminus A = \{s\}$. Set $D = B$ and $L = p$.

Let $E' \subseteq (V \times V) \setminus E$ be a set with at most L edges such that all vertices in A are at a distance of at most D from s in $G' = (V, E \cup E')$, i.e., E' is a concise certificate to MinCBEAP. Then, adding E' to G reduces the eccentricity of s to at most $B = D$ using at most $p = L$ edges, since the graph $G' = (V, E \cup E')$ is undirected. Consequently, E' is also a concise certificate to MCBE.

Exact integer programming approaches for the Minimum-cardinality balanced edge addition problem are developed in the next section.

3 Integer programming formulations

Given a non-oriented graph $G = (V, E)$, a vertex subset $A \subset V$, and a non-negative integer D, the optimization version of MinCBEAP amounts to finding a minimum cardinality set $S_G \subseteq (V \times V) \setminus E$ such that $d_{G'=(V, E \cup S_G)}(v, V \setminus A) \leq D$, $\forall v \in A$.

3.1 Instance transformation

There are no edges in an optimal solution S_G to MinCBEAP with both extremities in $V \setminus A$, because adding edges with both extremities in $V \setminus A$ would not affect the distance from any vertex in A to those in $V \setminus A$.

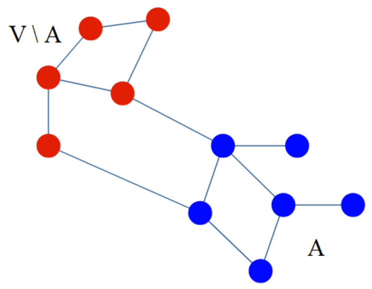
The following proposition holds:

Proposition 1 *Let S_G be a solution to MinCBEAP. Let $(u, v) \in S_G$ be an edge with $u \in A$ and $v \in V \setminus A$. Then, $(S_G \setminus \{(u, v)\}) \cup \{(u, w)\}$, with $w \in V \setminus A$ and $w \neq v$, is also a solution to MinCBEAP.*

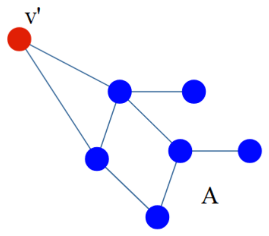
Proof Replacing edge (u, v) by edge (u, w) does not change the distance from any vertex in A to set $V \setminus A$.

Given a non-oriented graph $G = (V, E)$, a source vertex s, and a non-negative integer B, the optimization version of MCBE amounts to finding a minimum cardinality set $S_H \subseteq (V \times V) \setminus E$ such that $\epsilon_{G'=(V,E\cup S_H)}(s) \leq B$.

Then, consider the following transformation from an instance of MinCBEAP defined on graph $G = (V, E)$, as illustrated in Figure 4a, that creates an instance of MCBE on graph $H = (V_H, E_H)$, as illustrated in Figure 4b. In the transformed MCBE instance, $V_H = A \cup \{v'\}$, $s = v'$, and $B = D$, with the dummy vertex v' representing the collapsed set $V \setminus A$. Furthermore, for any vertex $u \in A$ such that there is an edge between u and some vertex $v \in V \setminus A$ in G, then there is an edge between u and v' in H. We also observe that while the number of vertices in $G = (V, E)$ is $|V|$, there are only $|A| + 1$ vertices in the graph $H = (V_H, E_H)$ that defines the MCBE instance.



(a) Instance of MinCBEAP on $G = (V, E)$



(b) Transformed instance of MCBE on $H = (V_H, E_H)$

Fig. 4: Instance transformation.

We make use of this transformation to find a solution for the transformed instance of MCBE, which is then used to obtain a solution for the original instance of MinCBEAP. Let S_H be an optimal solution for the transformed MCBE instance. A solution S_G for the original instance of MinCBEAP can be obtained as follows. Let $e = (u, v) \in S_H$. If both $u, v \in A$, then edge $e = (u, v)$ also belongs to S_G. In case one of the extremities – say, extremity v – of edge e coincides with $v' \notin A$, then we chose at random a vertex $w \in V \setminus A$, and substitute edge $e = (u, v')$ in S_H by edge $e' = (u, w)$ in S_G. Therefore, by construction, the solution S_G obtained for MinCBEAP has $|S_G| = |S_H|$.

3.2 First ILP formulation

Any optimal solution S_H to problem MCBE can be seen as an oriented spanning tree of the graph $H' = (V_H, E_H \cup S_H)$ rooted at vertex v'. The distance from any vertex in the tree to vertex v' should be at most D. The arcs of the oriented spanning tree indicate the paths from each vertex to the root v'.

This formulation makes use of a variant of the Miller-Tucker-Zemlin constraints to avoid cycles [17]. They create an arborescence in which each vertex v is labeled with an integer d_v. The root is labeled with $d_{v'} = 0$ and the vertices in any tree arc (v_1, v_2) are labeled with $d_{v_1} > d_{v_2}$.

The edges in the optimal solution are those associated with arcs that belong to the oriented spanning tree and not to E_H.

For each vertex $u \neq v$, we define the following decision variable:

$$x_{uv} = \begin{cases} 1, & \text{if arc } (u,v) \in A \times (A \cup \{v'\}), \text{ belongs to the oriented spanning tree,} \\ 0, & \text{otherwise.} \end{cases}$$

The integer variable d_v indicates the label of vertex $v \in V_H$. The formulation makes use of weights defined as $w_{uv} = 0$ if the associated edge $(u, v) \in E_H$, $w_{uv} = 1$ otherwise:

$$\min \sum_{u \in A} \sum_{v \in A \cup \{v'\}} w_{uv} x_{uv} \tag{1}$$

subject to:

$$\sum_{v \in A \cup \{v'\}, v \neq u} x_{uv} = 1, \ \forall u \in A \tag{2}$$

$$x_{uv} + x_{vu} \leq 1, \ \forall u, v \in A, \tag{3}$$

$$d_u \geq x_{uv} + d_v - (1 - x_{uv})D, \ \forall u \in A, \forall v \in A \cup \{v'\}, u \neq v \tag{4}$$

$$d_u \leq D, \ \forall u \in A \tag{5}$$

$$d_u \geq 1, \ \forall u \in A \tag{6}$$

$$d_{v'} = 0, \tag{7}$$

$$d_u = 1, \ \forall u \in A, \ v' \in N_H(u) \tag{8}$$

$$x_{uv'} = 1, \ \forall u \in A, \ v' \in N_H(u) \tag{9}$$

$$x_{uv} \in \{0, 1\}, \ \forall (u, v) \in A \times (A \cup \{v'\}) \tag{10}$$

$$d_v \in \{0, ..., D\}, \ \forall u \in A \cup \{v'\}, \tag{11}$$

with $N_H(u) = \{v \in A \cup \{v'\} : (u, v) \in E_H\}$.

The objective function (1) minimizes the number of edges, since the weights of edges in E_H are zero. Constraints (2) indicate that an arc must come out from every vertex of A, tracing the path (i.e., the last vertex before) to vertex v'. Constraints (3) enforce that there is at most one arc between any pair of vertices. Constraints (4) ensure that if $x_{uv} = 1$, i.e., arc (u, v) belongs to the oriented spanning tree, then $d_u > d_v$. On the other hand, if $x_{uv} = 0$, i.e., arc (u, v) does not belong to the oriented spanning tree, then the constraint becomes $d_u \geq d_v - D$ and is satisfied for any $d_u, d_v \in \{0, \ldots, D\}$. Constraints (5) and (6) indicate, respectively, upper and lower bounds to the vertex labels. Constraint (7) sets the label of vertex v' to zero. Constraints (8) set to one the labels of the vertices of A that are adjacent to v', while constraints (9) set to one the variables associated with the vertices of A that are adjacent to v'. Constraints (10) and (11) are the integrality requirements.

We observe that although the model can be solved without constraints (8) and (9), they are added to accelerate the solution process.

3.3 Second ILP formulation

We recall that our formulation addresses the transformed instance of MCBE on graph $H = (V_H, E_H)$, with v' being the dummy vertex. From Lemma 1, we know that there is always a solution S_H with all edges having v' as one of the extremities. Therefore, the problem can be solved by considering only this particular subset of solutions and deciding, for each vertex u, if edge (u, v') should be added to the graph.

Demaine and Zadimoghaddam [6] proposed a model solving the linear feasibility problem associated to the MCBE. The adaptation of this model to an optimization problem is described next. The following decision variabes are defined:

$$y_u = \begin{cases} 1, & \text{if there is an edge between vertex } u \in V_H \text{ and } v', \\ 0, & \text{otherwise;} \end{cases}$$

$$t_{uv} = \begin{cases} 1, & \text{if the shortest path from vertex } v \in V_H \text{ to } v' \text{ makes use of edge } (u, v'), \\ 0, & \text{otherwise.} \end{cases}$$

If the distance between v and v' is greater than D, then a path from v will reach v' using any of the vertices that are at a distance from v' that is smaller than D. Moreover, as noted in [6], vertex v can not use edge (u, v') if the distance between v and u is greater than D:

$$\min \sum_{u \in V_H : (u, v') \notin E_H} y_u \tag{12}$$

subject to:

$$t_{uv} \leq y_u, \;\; \forall u, v \in A \tag{13}$$

$$\sum_{u : dist(u,v) < D} t_{uv} = 1, \;\; v \in A, dist(v, v') > D \tag{14}$$

$$y_u \in \{0, 1\}, \;\; \forall u \in A \cup \{v'\} \tag{15}$$

$$t_{uv} \in \{0, 1\}, \;\; \forall u, v \in A \cup \{v'\} \tag{16}$$

The objective function (12) minimizes the number of edges adjacent to vertex v' to be added. Constraints (13) indicate that if vertex v_j reaches v' using edge (v_i, v'), then vertex v_i must be counted in the objective function.

Moreover, constraint (14) expresses that if the distance between v_j and v' is greater than D, then vertex j reaches v' using exactly one of the vertices that are at a distance to v' that is smaller than D. Constraints (15) and (16) are the integrality requirements.

3.4 Third ILP formulation

In this formulation based only on 0-1 variables, we also make use of Lemma 1 that establishes that there is always a solution to MCBE with all edges incident to the source vertex v'. In addition to the variables

$$y_u = \begin{cases} 1, & \text{if there is an edge between vertex } u \in V_H \text{ and } v', \\ 0, & \text{otherwise;} \end{cases}$$

already used in the previous formulation, we also define

$$d_{vk} = \begin{cases} 1, & \text{if there is an path of size } k \text{ from vertex } v \text{ to vertex } v', \\ 0, & \text{otherwise.} \end{cases}$$

The problem may then be formulated as:

$$\min \sum_{(u,v')\notin E_H} y_u \tag{17}$$

subject to:

$$\sum_{k=1}^{D} d_{uk} = 1, \;\; \forall u \in A \tag{18}$$

$$d_{u0} = 0, \;\; \forall u \in A \tag{19}$$

$$\sum_{k=1}^{D} d_{v'k} = 0, \tag{20}$$

$$d_{v'0} = 1, \tag{21}$$

$$d_{uk} \le \sum_{v \in N_H(u)} d_{vk-1}, \;\; \forall u \in A, k \in \{2, ..., D\} \tag{22}$$

$$d_{u1} = y_u, \;\; \forall u \in A, v' \notin N_H(u) \tag{23}$$

$$d_{u1} = 1, \;\; \forall u \in A, v' \in N_H(u) \tag{24}$$

$$y_u = 0, \;\; \forall u \in A, v' \in N_H(u) \tag{25}$$

$$y_u \in \{0,1\}, \;\; \forall u \in A \cup \{v'\} \tag{26}$$

$$d_{vk} \in \{0,1\}, \;\; \forall u \in A \cup \{v'\}, \forall k \in \{0, ..., D\} \tag{27}$$

The objective function (17) minimizes the number of edges adjacent to vertex v' to be added. Constraints (18) and (19) indicate that $1 \le dist(u, v') \le D$, $\forall u \in A$. Moreover, constraints (20) and (21) express that the distance from vertex v' to itself is zero. Constraints (22) indicate that if the distance from vertex $u \in A$ to vertex v' is $k \geq 2$, then the distance from one of its adjacent vertices to v' must be $k-1$. Constraints (23) ensure that for each vertex u that is not adjacent to v' in H, then its distance to v' will become equal to 1 in $H' = (V_H, E_H \cup S_H)$ if there is an edge between u and v' in the optimal solution. Constraints (24) and (25) fix the variables of the vertices adjacent to v' in H. Constraints (26) and (27) are the integrality requirements.

We now observe that the following property holds:

Proposition 2 *Let S_H be an optimal solution of the MCBE problem defined by a graph $H = (V_H, E_H)$, a source vertex v' and a constant D, and let $u \in V_H$, $u \neq v'$ be a vertex. If $d_H(u, v') = d \leq D$, then $d_{H'=(V_H, E_H \cup S_H)}(u, v') \leq d$.*

Proof Since $H' = (V_H, E_H \cup S_H)$ is a supergraph of $H = (V_H, E_H)$, then it contains all paths from u to v' that already exists in H. Consequently, the distance from vertex u to v' can not increase in H'.

In other words, all vertices $u \neq v'$ with $d(u, v') \leq D$ can not be, in the optimal solution, at a distance greater than their current distance to v'.

Therefore, constraints (18) can be replaced by the constraints below in an improved formulation:

$$\sum_{k=1}^{D} d_{uk} = 1, \ \forall u \in A, \, d(u, v') > D \tag{28}$$

$$\sum_{k=1}^{d(u,v')} d_{uk} = 1, \ \forall u \in A, \, d(u, v') \leq D \tag{29}$$

$$\sum_{k=d(u,v')+1}^{D} d_{uk} = 0, \ \forall u \in A, \, d(u, v') \leq D, \tag{30}$$

where constraints (28), (29) and (30) make use of the additional information about the distances from vertex v' to all other vertices in graph H.

Table 1 compares the three formulations in terms of their number of variables and constraints, where $n = |V_H|$.

Table 1: ILP formulations: number of variables and constraints.

	Variables	Constraints	All variables binary?
First formulation	$O(n^2)$	$O(n^2)$	No
Second formulation	$O(n^2)$	$O(n^2)$	Yes
Third formulation	$O(nD)$	$O(nD)$	Yes

4 Numerical results

The models were implemented and tested using version 12.7.1 of the CPLEX solver on an Intel Core i7 machine with a 3.2 GHz processor and 8 GB of RAM, running under the Windows 10 operating system.

4.1 Ramdomly generated test problems

Several experiments were performed to assess the performance of the integer programming models presented in the previous section. We created two set of instances: small and medium sized instances. The instances were generated as random graphs with two parameters: the number of vertices n and the number of randomly generated edges m inside the set A. The parameter D of the problem was set to 2, which is a reasonable target in practice. The instances are named indicating the values of n and m. For example, the instance named "inst_200v_4x" has $n = 200$ vertices and $m = n \times 4 = 800$ edges in set A.

Tables 2 and 3 contain the experimental results for the small and medium instances, respectively. For each instance and formulation, the tables display the number of added edges in the best solution found by the solver, the running time in seconds, and an indication whether the instance was solved to optimality or not within a time limit of 3600 seconds.

Table 2 shows that the third formulation outperforms the others, solving to optimality all small instances instances with up to 1000 vertices in much smaller running times.

Table 3 reports the same experimental results for the second and third formulations for the medium-sized instances, but exclusively for those where the number of edges is four or eight times the number of vertices, because for them the optimal solution is not quickly reached. The third formulation obtains better results when the number of vertices increases. We also observe that the memory space requirements of the second formulation increase very quickly with the number of vertices, making it impractical on a machine with a limited amount of memory space: not even feasible solutions were found for the instances with 5000 or more vertices due to memory limitations.

Table 4 illustrates the variation of the linear relaxation gap for the instances with 1000 and 2000 vertices with the increase in the number of edges. For the same instances, Figure 5 displays the evolution of the absolute gap when the number of edges increases. We observe that the largest absolute gap values are reached when the number of edges is 5 or 6 times greater than the number of vertices. For the same instances, the third – and best – formulation takes the longest times to reach the optimum. Therefore, instances with these densities seem to be the hardest to be solved by integer programming techniques.

Another observation that can be drawn from Table 4 is that the higher is the density of each of the polarized groups of vertices in a network, the smaller is the number of edges that should be added in the optimal solution, which makes these problems easier to be solved in practice.

Table 2: Results for small instances.

Instance	First formulation			Second formulation			Third formulation		
	Edges	Time (s)	Solved?	Edges	Time (s)	Solved?	Edges	Time (s)	Solved?
inst_100v_1x	29	0.091	yes	29	0.047	yes	29	0.013	yes
inst_100v_2x	19	0.285	yes	19	0.174	yes	19	0.049	yes
inst_100v_4x	9	1.167	yes	9	0.053	yes	9	0.028	yes
inst_100v_8x	3	1.029	yes	3	0.037	yes	3	0.024	yes
inst_100v_16x	1	1.303	yes	1	0.034	yes	1	0.034	yes
inst_200v_1x	75	0.310	yes	75	0.145	yes	75	0.005	yes
inst_200v_2x	42	0.469	yes	42	0.149	yes	42	0.007	yes
inst_200v_4x	17	6.754	yes	17	0.391	yes	17	0.110	yes
inst_200v_8x	7	9.108	yes	7	0.278	yes	7	0.069	yes
inst_200v_16x	1	27.874	yes	1	0.103	yes	1	0.021	yes
inst_500v_1x	177	2.146	yes	177	0.960	yes	177	0.043	yes
inst_500v_2x	100	8.139	yes	100	1.024	yes	100	0.067	yes
inst_500v_4x	45	171.620	yes	45	21.182	yes	45	3.177	yes
inst_500v_8x	20	101.572	yes	20	6.104	yes	20	0.324	yes
inst_500v_16x	7	250.697	yes	7	1.053	yes	7	0.038	yes
inst_1000v_1x	362	1690.160	yes	362	4.335	yes	362	0.057	yes
inst_1000v_2x	253	3610.160	(no)	197	5.064	yes	197	0.083	yes
inst_1000v_4x	187	3607.410	(no)	97	166.762	yes	97	9.414	yes
inst_1000v_8x	53	3647.120	(no)	38	1963.990	yes	38	50.929	yes
inst_1000v_16x	17	3618.540	(no)	12	4.127	yes	12	0.129	yes
inst_1500v_1x	571	3642.440	(no)	544	11.897	yes	544	0.058	yes
inst_1500v_2x	425	3651.650	(no)	294	12.627	yes	294	0.277	yes
inst_1500v_4x	292	3688.460	(no)	141	3607.990	(no)	139	3602.090	(no)
inst_1500v_8x	201	3647.860	(no)	62	3611.450	(no)	62	3602.940	(no)
inst_1500v_16x	100	3654.160	(no)	15	9.635	yes	15	0.132	yes
inst_2000v_1x	875	3736.490	(no)	702	22.718	yes	702	0.110	yes
inst_2000v_2x	1800	3784.000	(no)	390	25.002	yes	390	0.329	yes
inst_2000v_4x	475	3677.580	(no)	186	3617.370	(no)	186	3601.790	(no)
inst_2000v_8x	1800	4301.730	(no)	83	3623.620	(no)	78	3601.030	(no)
inst_2000v_16x	492	3862.770	(no)	19	22.158	yes	19	0.137	yes

Table 3: Results for medium-sized instances.

Instance	Second formulation			Third formulation		
	Edges	Time (s)	Solved?	Edges	Time (s)	Solved?
inst_3000v_4x	286	3776.23	(no)	280	3600.78	(no)
inst_3000v_8x	118	3723.63	(no)	115	3600.71	(no)
inst_4000v_4x	384	3963.45	(no)	373	3600.53	(no)
inst_4000v_8x	157	4176.90	(no)	153	3600.59	(no)
inst_5000v_4x	-	-	(no)	472	3600.57	(no)
inst_5000v_8x	-	-	(no)	200	3600.57	(no)
inst_6000v_4x	-	-	(no)	573	3600.42	(no)
inst_6000v_8x	-	-	(no)	240	3600.32	(no)
inst_7000v_4x	-	-	(no)	674	3600.56	(no)
inst_7000v_8x	-	-	(no)	278	3600.46	(no)
inst_8000v_4x	-	-	(no)	766	3600.62	(no)
inst_8000v_8x	-	-	(no)	306	3600.47	(no)
inst_9000v_4x	-	-	(no)	863	3600.66	(no)
inst_9000v_8x	-	-	(no)	368	3600.49	(no)
inst_10000v_4x	-	-	(no)	976	3600.86	(no)
inst_10000v_8x	-	-	(no)	398	3600.72	(no)

4.2 Real networks and interpretation

We also applied the solution approach proposed in Section 2 to the two instances recovered from Interian and Ribeiro [13] that appear in Figures 1 and 2: books and blogs, respectively. The third formulation of problem MinCBEAP described in Section 3 was solved for both instances.

Table 5 shows the results. We note that the number of edges in the solution that solves optimally each instance is very small in each case. The intervention associated with the addition of these edges to the graph represents, indeed, a small increase of less than 1% in the number of edges. This fact reflects the minimum intervention principle proposed in the problem formulation, showing that polarization can be reduced by small modifications in the structure of the graph.

These results also illustrate that edge additions make it possible to break the isolation of polarized groups by providing them with more plural information coming from other groups, as noticed in [14].

5 Concluding remarks

In this work, we introduced the Minimum-Cardinality Balanced Edge Addition Problem as a strategy for reducing polarization in real-world networks. We proved the NP-completeness of its decision version. We also proposed three new integer linear formulations for the optimization version, discussing computational results on both randomly generated and real-life instances. On the real-life instances, we showed that polarization can be reduced to the desired

Table 4: Linear relaxation gap on the instances with 1000 and 2000 vertices.

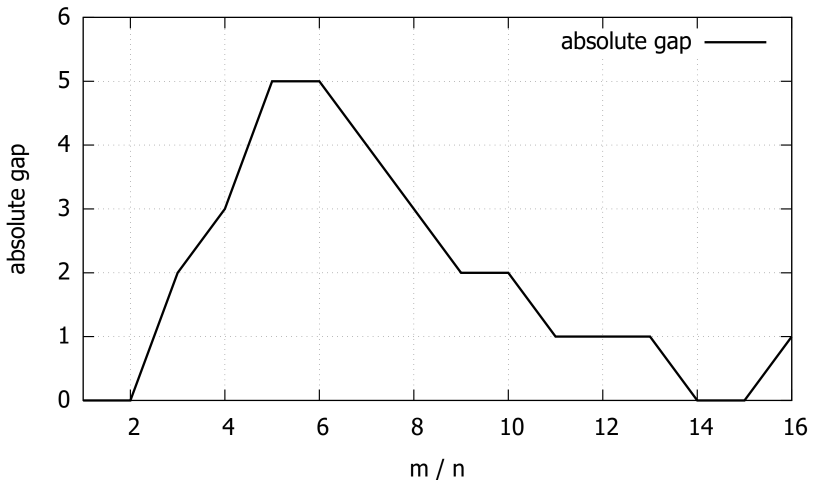
Instance name	Third formulation			Third formulation, LP relaxation			Gap $= \frac{x-\lceil y\rceil}{x}$
	Edges	Time (s)	Solved?	Edges	Time (s)	Solved?	
inst_1000v_1x	362	0.057	yes	362	0.027	yes	0
inst_1000v_2x	197	0.148	yes	196.29	0.034	yes	0
inst_1000v_3x	130	15.463	yes	127.44	0.043	yes	0.015
inst_1000v_4x	97	9.488	yes	93.84	0.041	yes	0.031
inst_1000v_5x	72	779.852	yes	66.79	0.058	yes	0.069
inst_1000v_6x	59	3600.98	(no)	53.41	0.053	yes	0.085
inst_1000v_7x	47	574.383	yes	42.36	0.064	yes	0.085
inst_1000v_8x	38	51.265	yes	34.42	0.041	yes	0.079
inst_1000v_9x	33	144.737	yes	30.14	0.029	yes	0.061
inst_1000v_10x	27	3.942	yes	24.05	0.029	yes	0.074
inst_1000v_11x	23	0.408	yes	21.07	0.044	yes	0.043
inst_1000v_12x	17	0.102	yes	15.89	0.026	yes	0.059
inst_1000v_13x	15	0.120	yes	13.80	0.023	yes	0.067
inst_1000v_14x	13	0.125	yes	12.57	0.024	yes	0
inst_1000v_15x	10	0.078	yes	9.53	0.023	yes	0
inst_1000v_16x	12	0.129	yes	10.62	0.024	yes	0.083
inst_2000v_1x	702	0.110	yes	702	0.056	yes	0
inst_2000v_2x	390	0.329	yes	388.74	0.099	yes	0.003
inst_2000v_3x	253	3603.000	(no)	247.14	0.195	yes	0.020
inst_2000v_4x	186	3601.790	(no)	175.42	0.224	yes	0.054
inst_2000v_5x	140	3601.060	(no)	127.42	0.273	yes	0.086
inst_2000v_6x	117	3600.940	(no)	104.79	0.205	yes	0.103
inst_2000v_7x	94	3601.130	(no)	84.51	0.146	yes	0.096
inst_2000v_8x	78	3601.030	(no)	69.77	0.132	yes	0.103
inst_2000v_9x	58	3601.710	(no)	52.59	0.099	yes	0.086
inst_2000v_10x	50	459.184	yes	45.72	0.078	yes	0.080
inst_2000v_11x	46	3433.460	yes	40.89	0.084	yes	0.109
inst_2000v_12x	38	6.289	yes	35.41	0.069	yes	0.053
inst_2000v_13x	32	4.755	yes	29.50	0.071	yes	0.063
inst_2000v_14x	25	0.507	yes	23.12	0.069	yes	0.040
inst_2000v_15x	22	0.469	yes	20.73	0.072	yes	0.045
inst_2000v_16x	19	0.137	yes	17.84	0.067	yes	0.053

Table 5: Results for real-life instances.

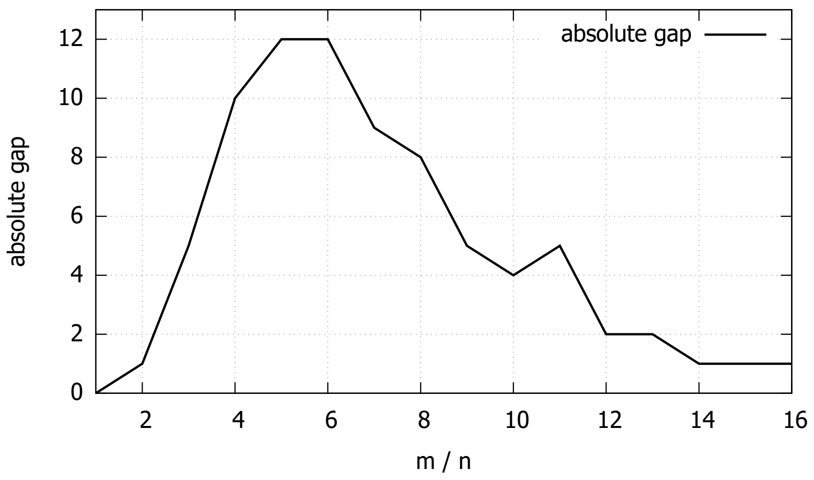
Instance name	Group	Vertices	Edges	ILP Model 3		
				Solution	Time	Solved
books	Conservative	50	420	1	0.006	yes
	Liberal	44	376	2	0.018	yes
	Neutral	14	44	0	0.014	yes
blogs	Republican	637	9352	8	0.048	yes
	Democratic	589	8805	17	0.014	yes

threshold with the addition of a few edges, as established by the minimum intervention principle that guided the problem formulation.

Another interesting conclusion is that in strongly polarized groups, there is often some easy way of spreading polarization-breaking information. This is a consequence of the fact that the higher is the density of a polarized group of vertices in a network, the smaller is the number of edges in the optimal solution, as previously observed in Section 4.1 from the results in Table 4.



(a) Instances with 1000 vertices.



(b) Instances with 2000 vertices.

Fig. 5: Variation of the absolute linear relaxation gaps with the increase in the number of edges: on horizontal axis, the ratio m/n between the number of edges and vertices in set A.

This study also shows that by using edge additions, completely isolated groups mentioned by Landrum [14] can start receiving more plural information, i.e., information coming from more that one group. Therefore, as suggested, disinformation can be broken by providing users a way to encounter diverse views of those practiced by members of the same groups they are trapped in.

Future work involves the study of graph properties that might lead to improvements in the efficiency of exact approaches, as well as the development of heuristic methods for handling hard instances that can not be solved by exact methods.

The minimum intervention principle that guided the approach proposed in this work and the exact methods developed here constitute an effective strategy for tackling polarization problems in real social, interaction, and communication networks. They make it possible to build concrete tools and strategies to address polarization issues in practice, since this is a relevant problem in a world characterized by extreme political and ideological polarization.

References

1. Abello, J., Pardalos, P.M., Resende, M.G.C.: On maximum clique problems in very large graphs. In: J.M. Abello, J.S. Vitter (eds.) External Memory Algorithms, pp. 119–130. American Mathematical Society (1999)
2. Abello, J., Resende, M., Sudarsky, S.: Massive quasi-clique detection. In: J. Abello, J. Vitter (eds.) Proceedings of the 5th Latin American Symposium on the Theory of Informatics, *Lecture Notes in Computer Science*, vol. 2286, pp. 598–612. Springer, Berlin (2002)
3. Adamic, L.A., Glance, N.: The political blogosphere and the 2004 U.S. election: Divided they blog. Proceedings of the 3rd International Workshop on Link Discovery pp. 36–43 (2005)
4. Arendt, H.: Between Past and Future. Viking Press, New York (1968)
5. Bilò, D., Gualà, L., Proietti, G.: Improved approximability and non-approximability results for graph diameter decreasing problems. Theoretical Computer Science **417**, 12–22 (2012)
6. Demaine, E.D., Zadimoghaddam, M.: Minimizing the diameter of a network using shortcut edges. Proceedings of the 12th Scandinavian Workshop on Algorithm Theory pp. 420–431 (2010)
7. Dictionaries, O.: Definition of polarization. Last access on 2017-09-06. https://en.oxforddictionaries.com/definition/polarization
8. Economist, T.: Not persuaded – political polarization. Last access on 2017-11-13. https://www.economist.com/blogs/democracyinamerica/2015/05/political-polarisation
9. Fabrikant, A., Luthra, A., Maneva, E., Papadimitriou, C.H., Shenker, S.: On a network creation game. Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing pp. 347–351 (2003)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
11. Garimella, K., De Francisci Morales, G., Gionis, A., Mathioudakis, M.: Reducing controversy by connecting opposing views. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 81–90. ACM, New York (2017)
12. Harary, F.: Graph Theory. Addison Wesley (1969)
13. Interian, R., Ribeiro, C.C.: An empirical investigation of network polarization. Applied Mathematics and Computation **339**, 651 – 662 (2018)
14. Landrum, A.: YouTube as a primary propagator of flat Earth philosophy. In: American Association for the Advancement of Science Annual Meeting. Washington, DC (2019). https://aaas.confex.com/aaas/2019/meetingapp.cgi/Session/21893
15. Li, C.L., McCormick, S.T., Simchi-Levi, D.: On the minimum-cardinality-bounded-diameter and the bounded-cardinality-minimum-diameter edge addition problems. Operations Research Letters **11**, 303–308 (1992)
16. Mill, J.S.: On liberty. J. W. Parker and Son, London (1859)
17. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. Journal of the ACM (JACM) **7**, 326–329 (1960)
18. Newman, M.E.J.: Network data – Books about US politics. Last access on 2017-12-11. http://www-personal.umich.edu/ mejn/netdata/

19. Papagelis, M.: Refining social graph connectivity via shortcut edge addition. ACM Transactions on Knowledge Discovery Data **10**, 12:1–12:35 (2015)
20. Papagelis, M., Bonchi, F., Gionis, A.: Suggesting ghost edges for a smaller world. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 2305–2308. ACM, New York (2011)
21. Perumal, S., Basu, P., Guan, Z.: Minimizing eccentricity in composite networks via constrained edge additions. In: 2013 IEEE Military Communications Conference, pp. 1894–1899. San Diego (2013)
22. Pinto, B.Q., Ribeiro, C.C., Rosseti, I., Plastino, A.: A biased random-key genetic algorithm for the maximum quasi-clique problem. European Journal of Operational Research **271**, 849–865 (2018)
23. Ribeiro, C.C., Riveaux, J.A.: An exact algorithm for the maximum quasi-clique problem. International Transactions in Operational Research **26**, 2199–2229 (2018)
24. Ribeiro, M.H., Calais, P.H., Almeida, V.A.F., Jr., W.M.: Everything I disagree with is #FakeNews: Correlating Political Polarization and Spread of Misinformation. Workshop on Data Science + Journalism @KDD 2017. https://sites.google.com/view/dsandj2017/accepted-papers
25. Times, N.Y.: How we became bitter political enemies. Last access on 2017-11-13. https://www.nytimes.com/2017/06/15/upshot/how-we-became-bitter-political-enemies.html
26. Vogiatzis, C., Walteros, J.L.: Integer programming models for detecting graph bipartitions with structural requirements. Networks **71**, 432–450 (2018)
27. Walteros, J.L., Veremyev, A., Pardalos, P.M., Pasiliao, E.L.: Detecting critical node structures on graphs: A mathematical programming approach. Networks **73**, 48–88 (2019)

APPENDIX C – An Iterated Greedy Heuristic for the Minimum-Cardinality Balanced Edge Addition Problem

An Iterated Greedy Heuristic for the Minimum-Cardinality Balanced Edge Addition Problem

Ruben Interian, Celso C. Ribeiro

Universidade Federal Fluminense, Institute of Computing, Niterói, RJ 24210-346, Brazil.
rinterian@ic.uff.br, celso@ic.uff.br

Abstract

The Minimum-Cardinality Balanced Edge Addition Problem (MinCBEAP) appears in the context of polarized networks as a strategy to reduce polarization by external interventions using the minimum number of edges. We show that every instance of MinCBEAP can be reduced to an instance of the Minimum-Cardinality-Bounded-Eccentricity Edge Addition Problem (MCBE). A restarted iterated greedy heuristic is developed for solving MinCBEAP via the transformed MCBE. Preliminary computational results are reported.

1 Problem statement

According to the Oxford Dictionaries, polarization is the division into sharply contrasting groups or sets of opinions or beliefs [1]. Academic articles, newspapers, and the media in general constantly report the growth of fake news, misinformation spreading, and polarization. These phenomena are closely interrelated with each other. Fake news spread faster in polarized networks or groups [6]. At the same time, fake and tendentious news can accentuate polarization within already existing echo chambers in the social networks.

Interian and Ribeiro [3] showed that many case studies real-world networks are extremely polarized. A polarized network is divided into two or more strongly connected groups, with few edges between vertices belonging to different groups. Most of the time, only same-group vertices communicate to each other and most of the information that a vertex can receive comes from inside the same group to which it belongs.

In order to reduce polarization, networks can be treated by external interventions consisting of the addition or the removal of vertices and edges. In this work, we address a new optimization problem that treats the issue of polarization reduction by edge additions. Given a graph $G = (V, E)$ and a subset of (polarized) vertices $A \subset V$, we seek a minimum-cardinality set of edges $E' \subseteq V \times V \setminus E$ to be added to G such that all vertices that are not in A can be reached from any vertex in a A by paths with at most D edges, where D is a problem parameter. The decision version of our optimization problem can be cast as:

Minimum-Cardinality Balanced Edge Addition Problem (MinCBEAP):
Instance: Graph $G = (V, E)$, subset $A \subset V$, integer D, integer L.
Goal: Is there a set $E' \subseteq (V \times V) \setminus E$ with cardinality $|E'| \leq L$ such that $d_{G'=(V,E\cup E')}(v, V \setminus A) \leq D$, $\forall v \in A$?

In the above formulation, $d_{G'=(V,E\cup E')}$ denotes the number of edges in a shortest path from $v \in A$ to the closest vertex in $V \setminus A$ in $G' = (V, E \cup E')$.

Given a graph $G = (V, E)$, the eccentricity $\epsilon(v)$ of a vertex $v \in V$ is the longest of the shortest paths in G from v to all other vertices in $V \setminus \{v\}$, see [2]. Any instance of MinCBEAP can be reduced to an instance of the Minimum-Cardinality-Bounded-Eccentricity Edge Addition Problem (MCBE):

Minimum-cardinality-bounded-eccentricity edge addition problem (MCBE):
Instance: Graph $G = (V, E)$, source vertex $s \in V$, integer B, integer p.
Question: Is there a supergraph $G' = (V, E \cup E')$ of G with $E' \subseteq (V \times V) \setminus E$ such that $|E'| \leq p$ and $\epsilon_{G'}(s) \leq B$?

Interian and Ribeiro [4] showed that MCBE is NP-complete. They also showed that MinCBEAP is NP-complete for D greater than or equal to 2, using a polynomial transformation from MCBE.

We propose in the following an efficient iterated greedy heuristic for approximately solving MinCBEAP using a transformation to MCBE. This article is organized as follows. In Section 2, we describe the polynomial transformation from an instance of MinCBEAP to one of MCBE and the iterated greedy heuristic. In Section 3, some preliminary computational results are presented.

2 Iterated greedy heuristic

We propose the following transformation of an instance of MinCBEAP into an instance of MCBE. Consider an instance of the optimization version of MinCBEAP defined by a graph $G = (V, E)$, a subset $A \subset V$, and an integer D. To build the corresponding optimization instance of MCBE, consider a graph $\bar{G} = (\bar{V}, \bar{E})$ and an integer $B = D$. Let $\bar{V} = A \cup \{s\}$, where s is the source vertex representing the set $V \setminus A$, in such a way that if there is an edge $(u, v) \in E$ with $u \in A$ and $v \in V \setminus A$, then there will be an edge $(u, s) \in \bar{E}$.

The MCBE instance is solved after the problem transformation. Let $S_H \subseteq (\bar{V} \times \bar{V}) \setminus \bar{E}$ be the optimal solution of MCBE for the instance defined by $\bar{G} = (\bar{V}, \bar{E})$. This solution is transformed to a solution $S_G \subseteq (V \times V) \setminus E$ of the original MinCBEAP instance as follows. Consider any edge $(u, v) \in S_H$. If both $u, v \in A$, then the edge (u, v) also belongs to S_G. Otherwise, in case the source $s \notin A$ coincides e.g. with extremity u of edge (u, v), then edge (w, v) is placed in S_G, where w is a randomly generated vertex from $V \setminus A$. Following this strategy, we obtain a solution S_G that solves MinCBEAP satisfying $|S_H| = |S_G|$.

The heuristic proposed in this work is based on the following lemma (the proof is omitted here due to space limitations):

Lemma 2.1. *There is a solution $S^* \subseteq (V \times V) \setminus E$ for MCBE such that all edges in S^* are incident to the source vertex s.*

Therefore, the set of candidate edges to be inserted in the solution can be restricted to those incident to vertex s. The problem is then reduced to determining a subset $X^* \subseteq V \setminus \{s\}$ of vertices such that for every vertex $v \in X^*$ there is an edge $\{v, s\} \in S^*$. In the algorithm, the solution S^* will be represented by the corresponding subset X^* of vertices, where $|S^*| = |X^*|$.

The cost of a solution to MCBE is given by the cardinality $|X|$ of the set X. It is well known that local search procedures often do not perform satisfactorily in the context of optimization problems without weights, in which the objective function is related to the cardinality of the solution, due to the lack of gradient information and to the exponential size of neighborhoods based on cardinality improving. In order to overcome this barrier, we propose in the following an iterated greedy heuristic [7] for MCBE, which is solely based on the repeated, successive application of destruction and reconstruction procedures, without appealing to local search.

In summary, iterated greedy starts from a greedy or a semi-greedy candidate solution, and generates a sequence of solutions using two main phases: destruction and reconstruction. During the destruction phase, some edges are removed from the current solution. The reconstruction procedure then applies a greedy constructive heuristic to reconstruct a complete candidate solution. If the cardinality of the newly constructed solution is less than the cardinality of the incumbent solution, then the latter is updated. The heuristic iterates over these steps until some stopping criterion is met [7].

One important building block of the iterated greedy heuristic is the semi-greedy (or greedy randomized) algorithm. There is a number $p(v)$ of non-solved neighbors associated with each vertex $v \in V$, that is, the number of neighbors of v that are still at a distance greater than B from the source vertex s. At each step, this algorithm adds one random vertex from a restricted candidate list (RCL) to the initially empty solution set X. The RCL is formed by all vertices $w \in V \setminus \{s\}$ that satisfy the condition $(1 - \alpha)p_{max} \leq p(w) \leq p_{max}$, where $p_{max} = max\{p(v) : \ v \in V \setminus \{s\}\}$ and α is the greediness parameter of the semi-greedy algorithm. We recall that $\alpha = 0$ corresponds to a purely greedy criterion, while $\alpha = 1$ corresponds to a completely random selection of the next vertex from $V \setminus \{s\}$.

Algorithm 1 shows the pseudo-code of the proposed iterated greedy heuristic. In line 1, the best solution X^* and its cardinality f^* are initialized. The while loop in lines 2–16 performs a new iteration while a stopping criterion is not met. The semi-greedy algorithm is applied in line 3 to build a solution X using the greediness parameter α. This solution is improved by performing a sequence of destruction-reconstruction phases in lines 4–12, until k_{max} iterations without any improvement are executed. The destruction phase is applied in line 7, by randomly removing a fraction β of the vertices in X. Next, the resulting solution is reconstructed by a greedy deterministic algorithm in line 8, which corresponds to the semi-greedy algorithm with $\alpha = 0$. If the cardinality of the newly reconstructed solution X' is less than the cardinality of the incumbent X, then the latter is updated in line 10. If the solution obtained after the sequence of destruction-reconstruction iterations is better than the best known solution X^*, then the best solution X^* and its cardinality f^* are updated in line 14. In line 17, the best solution found X^* is returned.

Algorithm 1 Restarted Iterated Greedy for MCBE

```
    Parameters: $\alpha, \beta, k_{max}$
    Input: $G = (V, E)$
    Output: $X^*$
 1: $X^* \leftarrow V$; $f^* \leftarrow |V|$;
 2: while stopping condition is not met do
 3:     $X \leftarrow SemiGreedy(G, \alpha)$;
 4:     $i \leftarrow 0$;
 5:     while $i \leq k_{max}$ do
 6:         $i \leftarrow i + 1$;
 7:         $X' \leftarrow PartialDestruction(G, X, \beta)$;
 8:         $X' \leftarrow GreedyReconstruction(G, X')$;
 9:         if $|X'| < |X|$ then;
10:             $X \leftarrow X'$; $i \leftarrow 0$;
11:         end if;
12:     end while;
13:     if $|X| < f^*$ then
14:         $X^* \leftarrow X$; $f^* \leftarrow |X|$;
15:     end if;
16: end while;
17: return $X^*$;
```

In the above algorithm, an external loop was added to the iterated greedy heuristic originally described by Ruiz and Stützle [7]. Instead of starting with a completely greedy solution, we start with a semi-greedy solution and the iterated greedy construction can be embedded in a multi-start procedure. This hybrid variant was named Restarted Iterated Greedy (RIG) by Pinto et al. [5].

3 Preliminary numerical results

We generated 13 random graph instances with different number of vertices. The number of edges in each instance was fixed as 8 times the number of vertices. The maximum length of the connecting paths was fixed as $D = 2$. The greediness parameter of the restarted iterated greedy was set at $\alpha = 0.1$, the fraction of the solution to be destructed at $\beta = 50\%$, the number of iterations without improvement at $k_{max} = 100$, and the time limit of 3600 seconds as the stopping criterion for the heuristic.

Table 1 shows preliminary results for the random instances, including the number of vertices $|X^*| = |S^*|$ in the best solution found and the running time in seconds. We are currently developing a mixed integer programming (MIP) approach for exactly solving small- and medium-size instance of MCBE. Preliminary results obtained by the MIP approach make it possible to identify the optimality of the solutions found by the restarted iterated greedy heuristic for the instances with up to 1000 vertices.

Instance	$\lvert A\rvert$	$\lvert X^*\rvert$	Time (s)	Solved?
inst_100v_8x	100	3	0.32	yes
inst_200v_8x	200	7	0.64	yes
inst_500v_8x	500	20	0.19	yes
inst_1000v_8x	1000	38	246.08	yes
inst_2000v_8x	2000	81	3610.24	(no)
inst_3000v_8x	3000	121	3609.47	(no)
inst_4000v_8x	4000	160	3609.60	(no)
inst_5000v_8x	5000	209	3606.40	(no)
inst_6000v_8x	6000	253	3614.21	(no)
inst_7000v_8x	7000	292	3621.25	(no)
inst_8000v_8x	8000	331	3617.79	(no)
inst_9000v_8x	9000	392	3638.78	(no)
inst_10000v_8x	10000	426	3644.16	(no)

Table 1: Results for the restarted iterated greedy heuristic.

References

[1] Oxford Dictionaries. Definition of polarization. Last access on 2017-09-06. https://en.oxforddictionaries.com/definition/polarization.

[2] F. Harary. *Graph Theory*. Addison Wesley, 1969.

[3] R. Interian and C. C. Ribeiro. An empirical investigation of network polarization. *Applied Mathematics and Computation*, 339:651–662, 2018.

[4] R. Interian and C. C. Ribeiro. Minimum-cardinality balanced edge addition in polarized networks: Formulations, complexity, and integer programming approaches. 2019. (Submitted for publication).

[5] B. Q. Pinto, C. C. Ribeiro, I. Rosseti, and A. Plastino. A biased random-key genetic algorithm for the maximum quasi-clique problem. *European Journal of Operational Research*, 271:849–865, 2018.

[6] M. H. Ribeiro, P. H. Calais, V. A. F. Almeida, and W. Meira Jr. Everything I disagree with is #FakeNews: Correlating Political Polarization and Spread of Misinformation. Workshop on Data Science + Journalism @KDD 2017. https://sites.google.com/view/dsandj2017/accepted-papers.

[7] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049, 2007.

APPENDIX D – A GRASP heuristic using path-relinking and restarts for the Steiner traveling salesman problem

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 24 (2017) 1307–1323
DOI: 10.1111/itor.12419

A GRASP heuristic using path-relinking and restarts for the Steiner traveling salesman problem

Ruben Interian and Celso C. Ribeiro

Institute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-346, Brazil
E-mail: rinterian@ic.uff.br [Interian]; celso@ic.uff.br [Ribeiro]

Received 31 July 2016; received in revised form 31 December 2016; accepted 28 February 2017

Abstract

The traveling salesman problem (TSP) is one of the most studied problems in combinatorial optimization. Given a set of nodes and the distances between them, it consists in finding the shortest route that visits each node exactly once and returns to the first. Nevertheless, more flexible and applicable formulations of this problem exist and can be considered. The Steiner TSP (STSP) is a variant of the TSP that assumes that only a given subset of nodes must be visited by the shortest route, eventually visiting some nodes and edges more than once. In this paper, we adapt some classical TSP constructive heuristics and neighborhood structures to the STSP variant. In particular, we propose a reduced 2-opt neighborhood and we show that it leads to better results in smaller computation times. Computational results with an implementation of a GRASP heuristic using path-relinking and restarts are reported. In addition, ten large test instances are generated. All instances and their best-known solutions are made available for download and benchmarking purposes.

Keywords: Steiner traveling salesman problem; traveling salesman problem; GRASP; path-relinking; restarts

1. Introduction

The traveling salesman problem (TSP) is one of the fundamental combinatorial optimization problems (Garey and Johnson, 1979; Zhang et al., 2015) and has numerous real-life applications in transportation, logistics, vehicle routing, genome sequencing, and other areas. In its undirected version it consists in, given a set of nodes and the distances between them, find the shortest route that visits each node exactly once and returns to the first city. Mathematically, the TSP can be defined as follows (Cornuéjols et al., 1985). Given a graph $G = (V, E)$ and a function $w : E \to \mathbb{R}$ that associates a weight $w(e)$ to each edge $e \in E$, the goal is to find a Hamiltonian cycle of minimum total weight (or cost). The TSP is NP-hard, since its decision version is proven to be NP-complete by a simple reduction from the Hamiltonian cycle problem (Garey and Johnson, 1979).

However, in many practical applications it is more frequent to find the following variant of the TSP. A set $V_R \subseteq V$ of required nodes is given. Instead of searching for a Hamiltonian cycle visiting

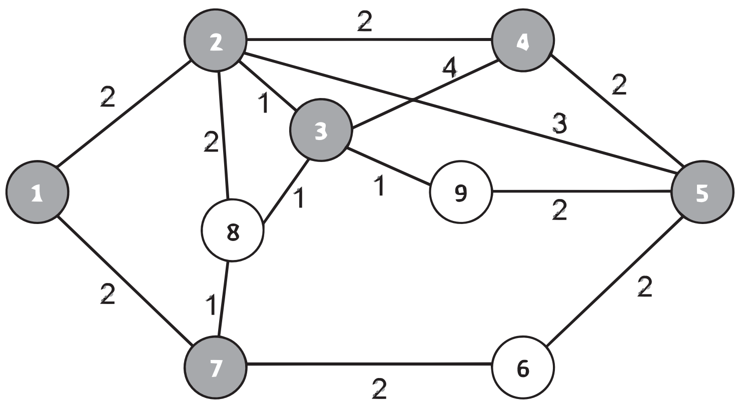


Fig. 1. Instance of Steiner TSP with nine nodes: the six required nodes are darkened.

all nodes, a minimum-weight closed walk is requested that visits only the required nodes. Since only a walk is sought, nodes can be visited more than once and edges may be traversed more than once. The so-called Steiner TSP (STSP, for short) was first proposed in Cornuéjols et al. (1985) and Fleischmann (1985), where its NP-hardness is also proved. The Steiner TSP is specially suitable to model network design (Borne et al., 2013), package delivery (Zhang et al., 2015, 2016), and routing (Letchford et al., 2013) problems. All of them are typically modeled using sparse graphs. Figure 1 illustrates an instance of the Steiner TSP with nine nodes, six of which are required.

Most studies on the Steiner TSP focus on integer programming formulations and valid inequalities. The STSP is solved efficiently (in linear time) for series-parallel graphs in Cornuéjols et al. (1985). Compact, polynomial size integer programming formulations of the TSP are extended to the STSP in Letchford et al. (2013). An extension of the Steiner TSP that adds penalties to the nodes not visited by the cycle is proposed in Salazar-González (2003). A network design problem consisting of multiple Steiner TSPs with order constraints is studied in Borne et al. (2013), using an integer linear programming formulation and a branch-and-cut algorithm. An extension of the STSP in which the edge traversal costs are stochastic and correlated is studied in Letchford and Nasiri (2015). An online algorithm is proposed in Zhang et al. (2015, 2016) to solve another extension of the STSP considering real-time edge blockages.

This paper is organized as follows. In the next section, adaptive greedy constructive heuristics for the Steiner TSP are presented. Section 3 reports local search strategies that are explored by the GRASP with path-relinking heuristic presented in Section 4. Computational experiments are reported in Section 5 and extended in Section 6, where an improved strategy exploring periodical restarts is developed. In addition, ten large test instances are generated and their best-known solutions are made available for download and benchmarking purposes in Section 7. Concluding remarks are drawn in the last section.

2. Greedy algorithms for the Steiner TSP

The following strategy can be applied as a heuristic for the Steiner TSP (Letchford et al., 2013). First, the instance of the STSP is reduced to a TSP instance in a complete graph defined by the set of required nodes, in which the new distances correspond to the shortest paths between every pair of required nodes in the original graph. Next, any exact or heuristic algorithm is used to solve the TSP in this new, complete graph. Finally, the solution of the TSP is converted into an STSP solution by expanding every edge by the corresponding shortest path between the two consecutive required nodes. However, if the original STSP instance is a sparse graph, the conversion to a standard TSP instance significantly increases the number of edges, some of which may never be used.

Therefore, instead of using a complete graph formed by the required nodes, we will use the original graph for searching a minimum-weight closed walk. We use a straightforward adaptation of the nearest neighbor TSP adaptive greedy heuristic (see, e.g., Resende and Ribeiro, 2016, Chapter 3) to the STSP described in Algorithm 1, which builds the solution greedily by choosing at each iteration the closest required node to the last node added to the walk.

The algorithm starts in line 1 by arbitrarily selecting any initial node $i \in V_R$ to start the walk. The set of required nodes $\mathcal{N}$ already visited by the walk is initialized in line 2. The partially built walk $\mathcal{P}$ is initialized in line 3. The currently visited required node *current* is set in line 4. The loop in lines 5–11 is performed until all required nodes have been visited. At each iteration, the node *next* to be visited is set to the closest among all yet unvisited required nodes. The shortest path $\mathcal{P}'$ from *current* to *next* is computed in line 7. The partially built walk $\mathcal{P}$ is updated in line 8 by appending the shortest path $\mathcal{P}'$ to it. The set of already visited required nodes $\mathcal{N}$ and the current node are updated in lines 9 and 10, respectively. Finally, after completing the loop, the shortest path from *current* to the initial node i is appended to the walk in lines 12 and 13. The result is returned in line 14.

Algorithm 1. Nearest neighbor adaptive greedy heuristic for STSP

```
1: Select initial required node $i \in V_R$;
2: $\mathcal{N} \leftarrow \{i\}$;
3: $\mathcal{P} \leftarrow \{i\}$;
4: $current \leftarrow i$;
5: while $\mathcal{N} \neq V_R$ do
6:     $next \leftarrow$ closest node to $current$ among all those in $V_R \setminus \mathcal{N}$;
7:     $\mathcal{P}' \leftarrow$ shortest path from $current$ to $next$;
8:     $\mathcal{P} \leftarrow \mathcal{P} \oplus \mathcal{P}'$;
9:     $\mathcal{N} \leftarrow \mathcal{N} \cup \{next\}$;
10:    $current \leftarrow next$;
11: end while;
12: $\mathcal{P}' \leftarrow$ shortest path from $current$ to initial node $i$;
13: $\mathcal{P} \leftarrow \mathcal{P} \oplus \mathcal{P}'$;
14: return $\mathcal{P}$.
```

In the case of the Steiner TSP, the greedy criterion is the choice of the nearest required node to be visited. Algorithms that add randomization to a greedy or adaptive greedy algorithm are called *semigreedy* or *randomized greedy* algorithms. Randomization is an important feature in the implementation of effective heuristics. Semigreedy algorithms act by replacing the deterministic greedy choice of the next element to be incorporated into the solution under construction by the random selection of an element from a restricted set of best candidate elements, called the restricted candidate list (RCL).

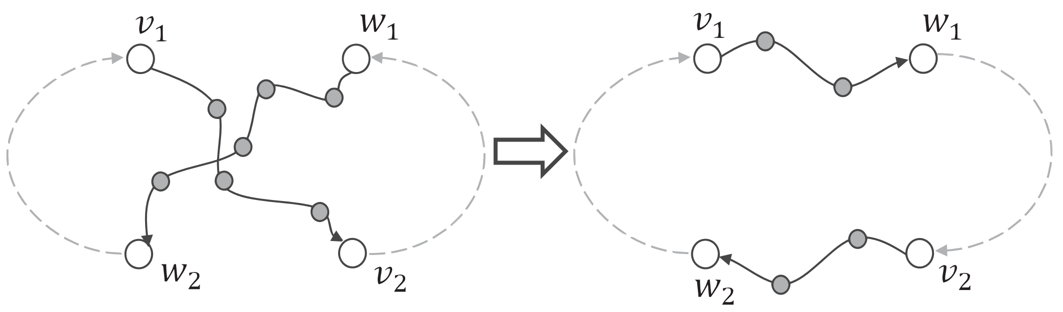


Fig. 2. 2-Opt move for STSP.

A simple quality-based scheme is used to define an RCL. Let $g_{\min} = \min\{g_i : i \in V_R \setminus \mathcal{N}$ and g_i is the shortest path from *current* to node $i\}$ and $g_{\max} = \max\{g_i : i \in V_R \setminus \mathcal{N}$ and g_i is the shortest path from *current* to node $i\}$. Furthermore, let α be such that $0 \le \alpha \le 1$. The RCL is formed by all yet unselected required nodes $i \in V_R \setminus \mathcal{N}$ satisfying $g_{\min} \le g_i \le g_{\min} + \alpha(g_{\max} - g_{\min})$.

3. Local search

Local search procedures are used to iteratively improve the quality of an initial solution, usually obtained by a constructive heuristic. First-improving (FI) and best-improving (BI) strategies are proposed and compared in terms of their performance. Efficient objective function updates are used, without the need of recalculating the objective function values from scratch: the previous walk weight is used in order to find the weight of the walk obtained after the changes performed during each local search iteration.

3.1. Neighborhood structure

The 2-opt neighborhood is the most commonly used neighborhood structure for the TSP problem and consists in replacing any pair of nonadjacent edges of the current solution by the unique pair of new edges that recreates a cycle.

The following property holds: Let $W = (v_1, \dots, v_i, \dots, v_j, \dots, v_m)$ be any optimal solution of the Steiner TSP. Then, the subpath $(v_i, \dots, v_j)$ is also a shortest path between the required nodes v_i and v_j. This is true because if this subpath was not the shortest, then W would not be optimal. Therefore, it is not necessary to investigate moves that involve changes in the order in which the nonrequired nodes are visited. Then, the problem amounts to determining the order in which the required nodes should be visited and then finding the shortest path between any pair of consecutive required nodes in the walk.

In consequence, we explore a 2-opt neighborhood for the STSP that is formed by all moves that replace the paths between two pairs of consecutive required nodes in the walk by the two unique pairs of shortest paths that reconnect a closed walk, as illustrated in Figure 2.

3.2. Reduced 2-opt neighborhood

A reduced 2-opt neighborhood can be defined in order to take advantage of the problem structure. In fact, convergence can be faster if only a few, promising moves in the neighborhood are considered.

We implement this idea in the following way. For each required node v, let $I(v)$ be the set formed by all required nodes that are reachable from v by a shortest path that does not visit any other required node. In other words, $I(v)$ represents the set of required nodes that are closer to v, in the sense that they necessarily belong to paths to farther nodes.

Using this auxiliary data structure, we restrict the 2-opt moves to pairs of consecutive required nodes (v_1, v_2) and (w_1, w_2) satisfying the condition that $w_1 \in I(v_1)$; see Figure 2.

3.3. First-improving versus best-improving local search

In a first-improving local search strategy, the algorithm moves from the current solution to any neighbor with a better value for the objective function. In contrast, in a best-improving local search strategy, the algorithm always moves from the current solution to the best of its neighbors. This implies that the time consumed by each iteration will be longer, but also that a better solution will result at the end of the iteration. Both strategies will be considered and compared using the 2-opt and restricted 2-opt neighborhoods.

4. GRASP with path-relinking heuristic

GRASP (which stands for *greedy randomized adaptive search procedures*) is a multistart metaheuristic in which each iteration consists of two main phases: construction and local search. The first phase is the construction of a feasible solution, usually by a greedy randomized algorithm. Once a feasible solution is obtained, its neighborhood is investigated until a local minimum is found during the second phase of local search. The best overall solution is kept as the result. The reader is referred to Resende and Ribeiro (2016) for a complete account of GRASP. Annotated bibliographies of GRASP appeared in Festa and Resende (2009a, 2009b). A recent application of GRASP appeared in Ferone et al. (2016).

We used the adaptive greedy randomized heuristic presented in Section 2 and the local search strategies described in Section 3 to customize a GRASP with path-relinking heuristic for the Steiner TSP.

Path-relinking is an intensification strategy that explores trajectories connecting elite solutions produced by metaheuristics. Path-relinking is usually carried out between two solutions: one is the initial solution S_i, while the other is the guiding solution S_g. A path that connects these solutions is constructed in the search for better solutions. Local search may be applied to the best solution in the path, since there is no guarantee that this solution is locally optimal. In the context of GRASP, path-relinking may be used to connect solutions obtained after the local search step with elite solutions produced during previous iterations, providing a sort of memory mechanism.

More specifically, in the context of the STSP, path-relinking attempts to preserve common characteristics of good walks, that is, common subpaths. As explained below, path-relinking matches the positions of the largest common subpath to the initial and guiding solutions and then swaps the positions of nodes that do not belong to this common subpath.

We first observe that any solution of the STSP has no unique representation as a sequence of the visited required nodes, since any closed walk can start from different nodes and can be traversed in two directions (forward and backward). Therefore, the representation of the initial and guiding solutions must be adjusted to facilitate the operation of relinking them. With this purpose, before applying path-relinking, we adjust the representations of the initial and guiding solutions by detecting the largest common subpath $w_l = (v_i \dots v_j)$ between them.

In our implementation, we choose to detect the largest (or longest) common subpath, instead of the longest common subsequence, in order to prioritize consecutive sequences of nodes in both solutions. This problem is known as the largest (or longest) common substring problem and can be solved in $O(n)$ time and space (Hui, 1992).

The guiding solution S_g and the initial solution S_i are oriented in the same direction according with w_l. Next, the initial nodes of the walks associated with S_g and S_i are made to coincide with the initial node v_i of w_l.

To move from the initial to the guiding solution, path-relinking considers a restricted neighborhood. Each move in this restricted neighborhood involves the swap of two required nodes in the walk corresponding to the current solution that are not in the same positions as they are visited in the guiding solution. In addition, each move should place at least one of the two involved nodes in the appropriate position corresponding to the order in which it will be visited in the guiding solution. After two required nodes are swapped, the shortest paths from their predecessors and to their successors are updated. Since at least one node is placed in the appropriate position of the guiding solution at each iteration, path-relinking will take at most as many iterations as the number of required nodes that were misplaced in the initial solution with respect to the guiding solution.

Algorithm 2 presents the pseudocode of path-relinking from the initial solution S_i to the guiding solution S_g. The current solution S and the best solution S^* are initialized in line 1. The cost f^* of the best solution found by path-relinking is initialized in line 2. The loop in lines 3–10 is performed until the current solution reaches the guiding solution. S' is set to the best solution in the restricted neighborhood of the current solution S in line 4. The best solution S^* found by path-relinking and its cost f^* are updated in lines 6 and 7, respectively, if the new solution S' improved the previous best. The current solution is updated in line 9 and a new path-relinking iteration resumes. The best solution found by path-relinking is returned in line 11.

Algorithm 2. Path-relinking algorithm for STSP

```
 1: $S, S^* \leftarrow S_i$;
 2: $f^* \leftarrow cost(S_i)$;
 3: while $S \neq S_g$ do
 4:     $S' \leftarrow$ best solution in the restricted neighborhood of $S$;
 5:     if $cost(S') < f^*$ then
 6:         $S^* \leftarrow S'$;
 7:         $f^* \leftarrow cost(S')$;
 8:     end if;
 9:     $S \leftarrow S'$;
10: end while;
11: return $S^*$.
```

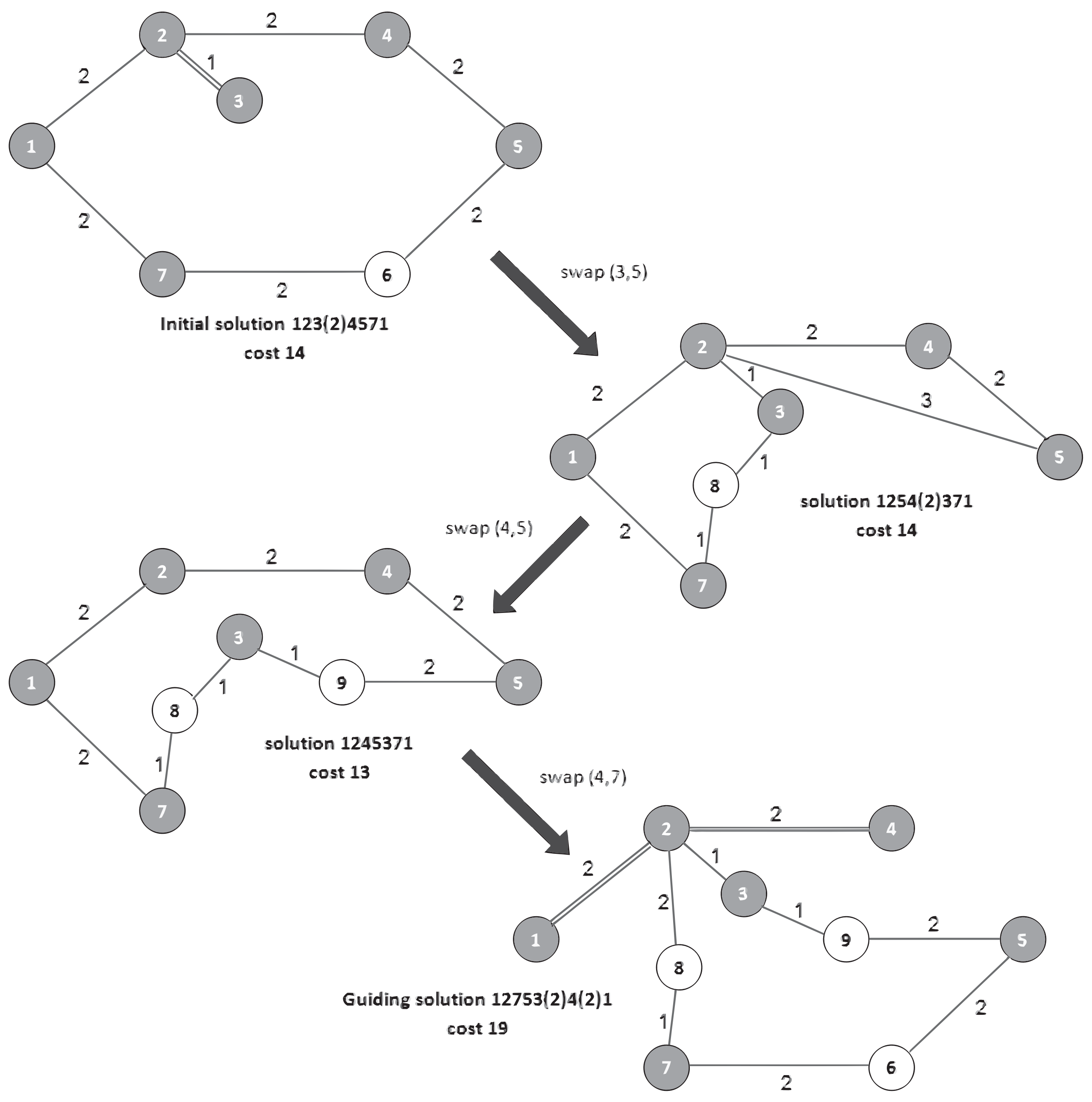


Fig. 3. Three iterations of path-relinking applied to initial and guiding solutions of the Steiner TSP instance in Figure 1: the six required nodes are darkened.

Figure 3 illustrates the application of path-relinking to the Steiner TSP instance described in Figure 1. The graph has six required nodes and three nonrequired nodes. The cost of the initial solution is 14, while that of the guiding solution is 19. The guiding solution is reached after three path-relinking iterations and the cost of the best solution found along them is 13.

The pseudocode in Algorithm 3 summarizes the main steps of the proposed GRASP with path-relinking (GRASP + PR) heuristic, following the same structure proposed in (Resende and Ribeiro, 2016, Section 9.3). The set of elite solutions is initialized in line 1. The loop in lines 2–12 is performed until some stopping criterion is satisfied. An initial solution is built in line 3 by the greedy randomized constructive heuristic described in Section 2. A local search procedure is used in line 4 to improve the solution obtained at the end of the construction phase. Except for the first iteration, when the elite set is still empty, lines 6–9 amount to the application of path-relinking. Line 6 randomly selects an elite solution S' from the elite set $\mathcal{E}$. The representation of solution S is adjusted considering the selected elite solution S'. Backward path-relinking is applied from the initial solution $S_i = S'$ to the guiding solution $S_g = S$. Local search is applied in line 9 to the solution obtained by path-relinking. The elite set $\mathcal{E}$ is updated with the new solution S in line 11. The best elite solution is returned in line 13.

Algorithm 3. GRASP+PR algorithm for STSP

```
1: $\mathcal{E} \leftarrow \emptyset$;
2: while stopping criterion not satisfied do
3:     $S \leftarrow RandomizedGreedy$;
4:     $S \leftarrow LocalSearch(S)$;
5:     if $|\mathcal{E}| > 0$ then
6:         Select solution $S'$ at random from $\mathcal{E}$;
7:         $S \leftarrow AdjustRepresentation(S, S')$
8:         $S \leftarrow PathRelinking(S, S')$;
9:         $S \leftarrow LocalSearch(S)$;
10:    end if;
11:    $UpdateEliteSet(S, \mathcal{E})$;
12: end while;
13: Return the best solution $S$ in $\mathcal{E}$.
```

5. Computational experiments

Several experiments were performed to assess the performance of the algorithms presented above and their variants. The algorithms were implemented in C# programming language and compiled by Roslyn, a reference C# compiler, in an Intel Core i5 machine with a 2.9 GHz processor and 8 GB of random-access memory, running under the Windows 10 operating system.

We considered the same test problems used by Letchford et al. (2013), as well as Letchford and Nasiri (2015) and Zhang et al. (2015, 2016), created by a random generator described in Letchford et al. (2013). This generator was designed to create graphs that resemble real-life road networks. It creates connected sparse graphs and a fraction of required nodes is specified for each instance. In addition to graphs from Letchford et al. (2013), we considered some larger instances with up to 300 nodes. Altogether, ten sparse weighted graphs with 50 to 300 nodes were used to assess the performance of the heuristics. Each graph generated two instances: one with $\lfloor \frac{N}{3} \rfloor$ required nodes and another with $\lfloor \frac{2 \cdot N}{3} \rfloor$ required nodes, where N is the total number of nodes, corresponding to 20 different instances. We observe that individual optimal values for each of these instances have not been previously reported in Letchford et al. (2013).

Table 1
Greedy randomized heuristic: average and best value over 100 runs

	1/3 of required nodes				2/3 of required nodes			
	$\alpha = 0.10$		$\alpha = 0.05$		$\alpha = 0.10$		$\alpha = 0.05$	
Nodes	Average	Best	Average	Best	Average	Best	Average	Best
50	944.63	**813**	947.48	814	1207.51	**979**	1213.34	1031
75	1068.80	**848**	1055.51	**848**	1308.21	1125	1309.49	**1094**
100	1070.02	959	1070.26	**947**	1607.99	1375	1599.44	**1299**
125	1436.82	1275	1425.84	**1228**	1929.31	1627	1911.41	**1623**
150	1410.76	**1151**	1403.36	1230	2110.56	**1875**	2102.40	1899
175	1642.85	**1379**	1615.89	1411	2248.23	1970	2265.40	**1913**
200	1759.45	1512	1765.95	**1445**	2615.95	2369	2614.78	**2354**
225	1826.48	1610	1845.74	**1604**	2817.25	2524	2844.12	**2522**
250	2045.90	**1769**	2041.03	1804	3022.70	2748	2988.79	**2723**
300	2186.87	**1907**	2226.41	1991	3242.32	**2952**	3264.87	2977

5.1. Selecting the quality measure for the RCL

In order to compare the effect of the value of α in the quality-based scheme used to define an RCL, we ran the randomized nearest neighbor constructive heuristic with $\alpha = 0.1$ and $\alpha = 0.05$. The greedy randomized algorithm was applied to all instances. Average and best values over 100 runs are presented in Table 1. As for all tables that follow, the best solution values found for each instance are depicted in boldface.

Although for the instances with one-third of required nodes quality-based constructive heuristic with $\alpha = 0.10$ found slightly more better solutions, the randomized heuristic with $\alpha = 0.05$ found significantly more better solutions for the instances with two-thirds of required nodes. We observe that the use of a better constructive method for building the initial solutions is likely to improve the quality of the solutions produced by the GRASP heuristic.

5.2. Comparing the local search variants

The solutions obtained by the greedy randomized construction algorithm can be refined either by first-improving (FI) or by best-improving (BI) local search strategies. In this experiment, we considered 100 iterations of the pure GRASP (without path-relinking) heuristic using the 2-opt neighborhood structure. The results can be seen in Table 2.

GRASP with best-improving LS clearly outperformed GRASP with the first-improving LS strategy, since substantially better solution values were reached.

5.3. Reduced 2-opt neighborhood

We now address the benefits of using the reduced 2-opt neighborhood, designed specifically for this problem. Preliminary computational experiments have shown that the use of this reduced

Table 2
First-improving versus best-improving local search, 100 GRASP iterations

	1/3 of required nodes		2/3 of required nodes	
Nodes	FI	BI	FI	BI
50	**789**	**789**	979	**978**
75	**830**	**830**	1049	**1035**
100	**931**	934	1280	**1239**
125	1188	**1171**	**1489**	1496
150	1152	**1110**	1699	**1643**
175	1325	**1286**	1759	**1743**
200	1375	**1338**	2051	**1976**
225	1470	**1442**	2202	**2094**
250	1589	**1515**	2313	**2224**
300	1798	**1717**	2603	**2409**

FI, first-improving; BI, best-improving.

Table 3
2-opt versus reduced 2-opt neighborhoods, 100 GRASP iterations

	1/3 of required nodes				2/3 of required nodes			
	2-Opt neighborhood		Reduced 2-opt		2-Opt neighborhood		Reduced 2-opt	
Nodes	Value	Seconds	Value	Seconds	Value	Seconds	Value	Seconds
50	789	0.204	789	0.171	978	0.484	978	0.36
75	830	0.469	830	0.375	**1035**	1.078	1045	0.985
100	934	0.766	**919**	0.672	1239	2.359	**1208**	1.610
125	1171	1.313	**1166**	1.093	1496	4.454	**1469**	2.921
150	**1110**	2.015	1121	1.703	1643	7.468	**1615**	4.703
175	1286	3.016	**1272**	2.281	1743	11.313	**1719**	6.687
200	1338	4.343	**1304**	3.235	1976	17.469	**1925**	9.187
225	1442	5.610	**1415**	4.140	2094	24.891	**2047**	12.828
250	**1515**	7.656	1539	5.578	2224	32.546	**2170**	15.297
300	1717	11.531	**1687**	7.328	2409	55.718	**2285**	26.578

neighborhood led to some target objective function values much faster than the use of the entire 2-opt neighborhood. These empirical observations were explored by the implementation of an alternative VND (variable neighborhood descent) local search procedure. First, only moves in the reduced 2-opt neighborhood are applied. The full neighborhood is explored only after a local minimum is obtained in the reduced 2-opt neighborhood.

Table 3 illustrates the efficiency of the VND approach when compared with the classic use of the full 2-opt neighborhood. It presents the solution values and computation times in seconds for 100 iterations of the pure GRASP (without path-relinking) heuristic using both pure 2-opt neighborhood and VND approach for local search. In both cases, local search was implemented following a best improvement strategy.

The VND local search strategy starting by the reduced 2-opt neighborhood led to the best solutions for 17 out of the 20 test problems. In addition, its computation times have been significantly

Table 4
GRASP versus GRASP+PR, 200 pure GRASP iterations, instances with one-third of required nodes

	Greedy ($\alpha = 0$)	GRASP (200 iterations)		GRASP + PR (same running time)		
Nodes	Value	Value	Seconds	Value	Seconds	Iterations
50	906	**789**	0.359	**789**	0.359	207
75	1181	**830**	0.782	**830**	0.797	191
100	1030	**919**	1.406	**919**	1.406	191
125	1306	**1145**	2.343	1148	2.344	188
150	1429	1121	3.469	**1108**	3.469	191
175	1606	**1272**	4.828	**1272**	4.828	193
200	1595	**1295**	6.500	**1295**	6.500	187
225	1625	1416	8.594	**1401**	8.609	195
250	1956	1531	10.859	**1491**	10.860	186
300	2030	1693	15.750	**1657**	15.797	191

smaller for all instances. As an example, in the case of the largest instance with two-thirds of required nodes, the time taken by 100 GRASP iterations using the VND local search strategy amounted to only 47.8% of the time taken when exclusively the complete 2-opt neighborhood was used. The GRASP heuristic using the VND local search strategy performed better both in terms of solution quality and computation times.

5.4. *Probabilistic choice of α*

We have already shown in Section 5.1 that although choosing $\alpha = 0.05$ most often leads to better results than $\alpha = 0.10$ for the greedy randomized heuristic proposed for the STSP, for some instances the latter was a better choice than the former. Different probabilistic strategies were considered in Prais and Ribeiro (2000) for the choice of the RCL parameter α, in contrast with the commonly used choice of fixing its value (see also Resende and Ribeiro, 2016). It was shown that a randomly chosen α from a decreasing nonuniform discrete probability distribution offers a good compromise between the running time of the algorithm and the quality of the solutions produced by the randomized heuristic. We relied on this work to sustain that it could be a good choice, in addition to the previously used appropriate value $\alpha = 0.05$, to consider also other higher values for this parameter with smaller probabilities of being chosen at each iteration. Therefore, in the following experiments, we used $\alpha = 0.05$ with a probability 70% and the values $\alpha = 0.10$ and $\alpha = 0.20$ with probabilities 20% and 10%, respectively.

5.5. *Path-relinking*

In this section, we address the impact of path-relinking in the search process. Tables 4 and 5 show the lengths of the solutions produced by the nearest neighbor adaptive greedy heuristic for the pure GRASP heuristic running for 200 iterations (together with its running time in seconds), and by the GRASP with backward path-relinking running by the same time taken by 200 pure GRASP

Table 5
GRASP versus GRASP + PR, 200 pure GRASP iterations, instances with two-thirds of required nodes

	Greedy ($\alpha = 0$)	GRASP (200 iterations)		GRASP + PR (same running time)		
Nodes	Value	Value	Seconds	Value	Seconds	Iterations
50	1356	**978**	0.750	**978**	0.750	179
75	1204	1031	1.781	**1029**	1.782	183
100	1290	1211	3.484	**1193**	3.485	182
125	1915	1450	6.297	**1427**	6.313	196
150	1847	1615	9.812	**1567**	9.844	184
175	2085	1704	14.640	**1667**	14.641	191
200	2484	1952	19.828	**1895**	19.828	177
225	2549	2024	27.266	**1973**	27.281	191
250	2523	2165	35.078	**2080**	35.172	182
300	2759	2308	62.328	**2219**	62.609	193

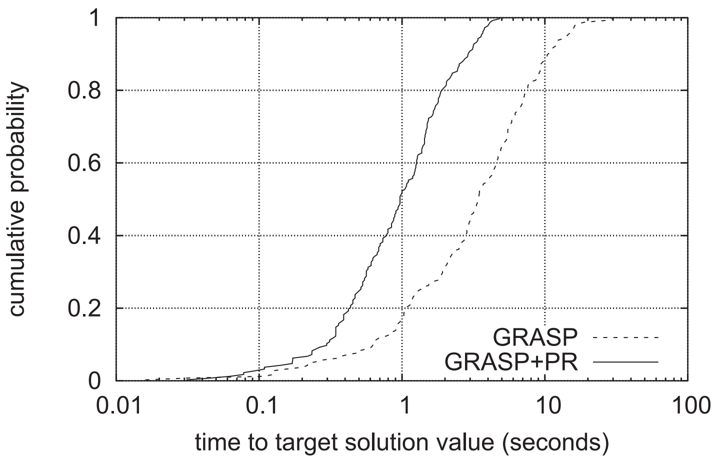


Fig. 4. Time-to-target plot for 200-node instance with one-third of required nodes and target value set to 1330.

iterations. The randomized heuristic used in the construction phase of both pure GRASP and GRASP with path-relinking algorithms uses the probabilistic criterion for the choice of α discussed in Section 5.4. The local search phase of both pure GRASP and GRASP with path-relinking algorithms was implemented using the best improvement VND strategy starting by the reduced 2-opt neighborhood. Path-relinking used elite sets formed by at most ten elements.

Path-relinking considerably improved GRASP performance, leading to better solutions for all but one instance in the same running time and fewer iterations than the pure GRASP heuristic. Time-to-target plots for GRASP and GRASP with path-relinking (GRASP+PR) algorithms for 200-node instances are shown in Figures 4 and 5. The target values are 1330 and 2000 for the instances with one-third and two-thirds of required nodes, respectively. Each algorithm was run 200 times. The plots in these figures provide empirical evidence that algorithm GRASP + PR outperforms GRASP for these instances and target values.

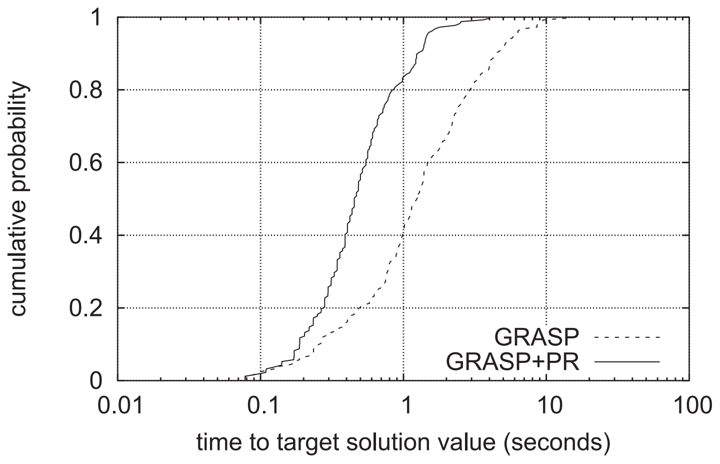


Fig. 5. Time-to-target plot for 200-node instance with two-thirds of required nodes, and target value set to 2000.

Table 6
Restart strategies for 1000 iterations, instances with one-third of required nodes

	No restarts		Restart(100)		Restart(200)	
Nodes	Value	Seconds	Value	Seconds	Value	Seconds
50	**789**	1.75	**789**	1.71	**789**	1.70
75	**830**	4.03	**830**	4.06	**830**	4.04
100	**919**	7.23	**919**	7.20	**919**	7.18
125	**1143**	12.37	**1143**	12.26	**1143**	12.20
150	**1105**	19.07	**1105**	18.51	**1105**	18.32
175	**1272**	26.73	**1272**	27.01	**1272**	27.78
200	**1295**	36.28	**1295**	35.62	**1295**	35.59
225	**1377**	47.03	1384	48.95	1389	47.71
250	1489	62.23	**1487**	59.00	1498	60.50
300	1648	88.51	**1629**	85.95	1645	86.46

6. Restart strategies for GRASP with path-relinking

Resende and Ribeiro (2011) have shown that restart strategies are able to reduce the running time to reach a target solution value for many problems. We apply the same type of restart(κ) strategy in which the elite set is emptied and the heuristic restarted from scratch after κ consecutive iterations have been performed without improvement in the best solution found. We evaluate the performance of the restart strategies for the Steiner TSP for $\kappa = 100$ and $\kappa = 200$. Computational results for the restart strategies for STSP are displayed in Tables 6 and 7, showing that they contribute to find better solutions in the same number of iterations, mainly when the problem size increases.

Figure 6 depicts time-to-target plots for the restart(200) strategy, compared to the original strategy without restarts, for the 200-node instance with two-thirds of required nodes and the target value set to 1900.

As previously observed in Resende and Ribeiro (2011, 2016), the effect of restart strategies can be mainly noted in the longest runs. Considering the 200 runs for the 200-node instance with the

Table 7
Restart strategies for 1000 iterations, instances with two-thirds of required nodes

	No restarts		Restart(100)		Restart(200)	
Nodes	Value	Seconds	Value	Seconds	Value	Seconds
50	**978**	4.03	**978**	3.96	**978**	3.95
75	**1029**	9.46	**1029**	9.45	**1029**	9.26
100	**1193**	18.95	**1193**	18.73	**1193**	18.43
125	1421	33.04	**1417**	33.93	1420	33.09
150	1565	53.39	1564	53.04	**1562**	52.23
175	1657	77.25	1665	76.84	**1652**	76.23
200	1883	105.53	1885	105.68	**1867**	105.20
225	1941	148.96	1953	144.68	**1928**	142.26
250	2054	173.68	2073	175.01	**2035**	176.04
300	2203	304.40	**2192**	310.03	2205	296.76

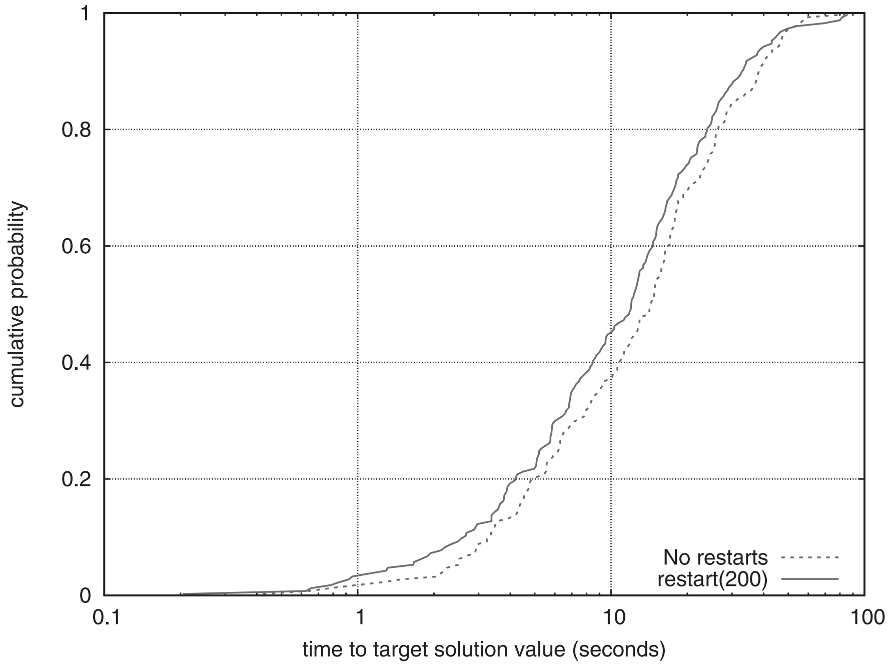


Fig. 6. Time-to-target plot for 200-node instance with two-thirds of required nodes and target value set to 1900.

target value set to 1900, they are associated with the column corresponding to the fourth quartile of Table 8. Entries in this quartile correspond to those in the heavy tails of the runtime distributions. The restart strategies in general do not affect too much the other quartiles of the distributions, which is a desirable characteristic. Compared to the no restart strategy, the restart(200) strategy was able to reduce not only the average running time in the fourth quartile, but also in the third and second quartiles. Consequently, strategy restart(200) performed the best among those tested, with the smallest average running times over the 200 runs.

Table 8
Summary of computational results for each restart strategy for the 200-node instance: 200 independent runs were executed for each strategy. Each run was made to stop when a solution as good as the target solution value 1900 was found. For each strategy, the table shows the distribution of the running times by quartile. For each quartile, the table gives the average running times in seconds over all runs in that quartile. The average running times over the 200 runs are also given for each strategy.

	Average running times in quartile (seconds)				
Strategy	1st	2nd	3rd	4th	Average
Without restarts	3.648	9.915	17.952	37.355	17.218
Restart(100)	2.933	8.466	17.067	37.509	16.494
Restart(200)	2.955	**8.093**	**15.410**	**34.878**	**15.334**

Table 9
Larger instances: description and numerical results for 1000 iterations of the GRASP heuristic using restarts and path-relinking

Instance name	Nodes	Edges	Required nodes	Solution value	Time (seconds)
euroroad05	1174	1417	58	31,571	147.5
euroroad10	1174	1417	117	49,975	354.9
euroroad15	1174	1417	176	58,016	684.6
euroroad20	1174	1417	234	71,967	1162.0
isprouters05	2113	6632	105	20,258	1030.5
isprouters10	2113	6632	211	37,486	3360.7
isprouters15	2113	6632	316	51,571	6850.6
rome05	3353	8870	167	481,048	1893.5
rome10	3353	8870	335	622,491	5196.2
rome15	3353	8870	502	795,535	8494.4

7. Results for larger instances

We have also created a set of ten substantially larger test instances for future benchmarking purposes. These instances were created from tree sparse graphs:

- street network of the city of Rome (Storchi et al., 1999);
- main roads between European cities (Rossi and Ahmed, 2015); and
- network of main Internet service providers at global level (Spring et al., 2002).

For each of the above graphs, we created several different instances with different number of required nodes, randomly chosen from the set of vertices of the graph. The number of nodes, edges, and required nodes, together with the walk lengths and the running times in seconds obtained by the GRASP heuristic using restarts and path-relinking, are presented in Table 9. All data are available from http://www2.ic.uff.br/~rinterian/instances/allinstances.html.

8. Concluding remarks

In the STSP, one seeks a minimum-weight closed walk that visits a subset of required nodes. Since only a walk is sought, nodes can be visited more than once and edges may be traversed more than once.

In this paper, we developed a GRASP with path-relinking and restarts for solving the STSP. The algorithm used in the construction phase is a randomized extension of the nearest neighbor heuristic for the TSP. A VND strategy exploring a reduced 2-opt neighborhood is used to optimize a best-improving local search scheme. Path-relinking and restart strategies are used to improve the efficiency of the GRASP algorithm.

Extensive computational results for a set of instances previously used in the literature are reported. In addition, we also considered a set of larger test instances derived from real-life graphs. Since neither optimal values nor even upper bounds have been previously reported for these instances, the solutions obtained by the GRASP with path-relinking and restarts heuristic proposed in this work cannot be directly compared to other solutions.

As a step toward avoiding this difficulty and facilitating the research on this problem, we made all test instances considered in this paper available at the URL http://www2.ic.uff.br/~rinterian/instances/allinstances.html together with their best-known solution values. This website will be updated with information provided by other researchers working on this problem with optimal values, lower and upper bounds for these and other benchmarking instances.

References

Borne, S., Mahjoub, A.R., Taktak, R., 2013. A branch-and-cut algorithm for the multiple Steiner TSP with order constraints. *Electronic Notes in Discrete Mathematics* 41, 487–494.

Cornuéjols, G., Fonlupt, J., Naddef, D., 1985. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming* 33, 1–27.

Ferone, D., Festa, P., Resende, M.G.C., 2016. Hybridizations of GRASP with path relinking for the far from most string problem. *International Transactions in Operational Research* 23, 481–506.

Festa, P., Resende, M.G.C., 2009a. An annotated bibliography of GRASP, Part I: Algorithms. *International Transactions in Operational Research* 16, 1–24.

Festa, P., Resende, M.G.C., 2009b. An annotated bibliography of GRASP, Part II: Applications. *International Transactions in Operational Research* 16, 131–172.

Fleischmann, B., 1985. A cutting plane procedure for the travelling salesman problem on road networks. *European Journal of Operational Research* 21, 3, 307–317.

Garey, M.R., Johnson, D.S., 1979. *Computersand Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York.

Hui, L.C.K., 1992. *Color set size problem with application to string matching. Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching*, Springer-Verlag, London, pp. 230–243.

Letchford, A.N., Nasiri, S.D., 2015. The Steiner travelling salesman problem with correlated costs. *European Journal of Operational Research* 245, 62–69.

Letchford, A.N., Nasiri, S.D., Theis, D.O., 2013. Compact formulations of the Steiner traveling salesman problem and related problems. *European Journal of Operational Research* 228, 83–92.

Prais, M., Ribeiro, C.C., 2000. Parameter variation in GRASP procedures. *Investigación Operativa* 9, 1–20.

Resende, M.G.C., Ribeiro, C.C., 2011. Restart strategies for GRASP with path-relinking heuristics. *Optimization Letters* 5, 467–478.

Resende, M.G.C., Ribeiro, C.C., 2016. *Optimization by GRASP*. Springer, Boston.

Rossi, R.A., Ahmed, N.K., 2015. The network data repository with interactive graph analytics and visualization. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, pp. 4292–4293.

Salazar-González, J.J., 2003. The Steiner cycle polytope. *European Journal of Operational Research* 147, 3, 671–679.

Spring, N., Mahajan, R., Wetherall, D., 2002. Measuring ISP topologies with Rocketfuel. SIGCOMM'02, Vol. 32, Pittsburgh, pp. 133–145.

Storchi, G., Dell'Olmo, P., Gentili, M., 1999. 9th DIMACS Implementation Challenge—Shortest Paths, Piscataway. Available at http://www.dis.uniroma1.it/challenge9/download.shtml (accessed 31 December 2016).

Zhang, H., Tong, W., Xu, Y., Lin, G., 2015. The Steiner traveling salesman problem with online edge blockages. *European Journal of Operational Research* 243, 30–40.

Zhang, H., Tong, W., Xu, Y., Lin, G., 2016. The Steiner traveling salesman problem with online advanced edge blockages. *Computers & Operations Research* 70, 26–38.

APPENDIX E – A GRASP with restarts heuristic for the Steiner traveling salesman problem

A GRASP with restarts heuristic for the Steiner traveling salesman problem

Ruben Interian, Celso C. Ribeiro

Institute of Computing, Universidade Federal Fluminense,
Niterói, RJ 24210-346, Brazil.
rinterian@ic.uff.br, celso@ic.uff.br

Abstract

Given a set of nodes and the distances between them, the traveling salesman problem (TSP) consists in finding the shortest route that visits each node exactly once and returns to the first. The Steiner traveling salesman problem (STSP) is a variant of the TSP that assumes that only a given subset of nodes must be visited by a shortest route, eventually visiting some nodes and edges more than once. In this paper, we extend some classical TSP constructive heuristics and neighborhood structures to the STSP variant. In particular, we propose a reduced 2-opt neighborhood and we show that it leads to better results in smaller computation times. Computational results with an implementation of a GRASP heuristic using path-relinking and restarts are reported. In addition, a set of test instances and best known solutions is made available for benchmarking purposes.

1 Introduction

The traveling salesman problem (TSP) is one of the fundamental combinatorial optimization problems [4, 12] and has numerous real-life applications in transportation, logistics, vehicle routing, genome sequencing, and other areas. Given a set of nodes and the distances between them, it consists in finding the shortest route that visits each node exactly once and returns to the first. Mathematically, the TSP can be defined as follows [2]. Given a graph $G = (V, E)$ and a function $w : E \to \mathbb{R}$ that associates a weight $w(e)$ to each edge $e \in E$, the goal is to find a Hamiltonian cycle of minimum total weight (or cost). The traveling salesman problem is NP-hard, since its decision version is proven to be NP-complete by a simple reduction from the Hamiltonian cycle problem [4].

However, in many practical applications it is more frequent to find the following variant of the TSP. A set $V_R \subseteq V$ of required nodes is given. Instead of searching for a Hamiltonian cycle visiting all nodes, a minimum-weight closed walk is requested that visits only the required nodes. Since only a walk is sought, nodes can be visited more than once and edges may be traversed more than once. The so-called Steiner traveling salesman problem (Steiner TSP, or STSP) was first proposed in [2, 3], where its NP-hardness is also proved. The Steiner TSP is specially suitable to model network design [1], package delivery [12, 13], and routing [7] problems. All of them are typically modeled using sparse graphs.

Most studies on the Steiner TSP focus on integer programming formulations and valid inequalities. The STSP is solved efficiently (in linear time) for series-parallel graphs in [2]. Compact, polynomial size integer programming formulations of the TSP are extended to the STSP in [7]. An extension of the Steiner TSP that adds penalties to the nodes not visited by the cycle is proposed in [11]. A network design problem consisting of multiple Steiner TSPs with order constraints is studied in [1], using an integer linear programming formulation and a branch-and-cut algorithm. An extension of the STSP in which the edge traversal costs are stochastic and correlated is studied in [6]. An online algorithm is proposed in [12, 13] to solve another extension of the STSP considering real-time edge blockages.

This paper is organized as follows. In the next section, adaptive greedy constructive heuristics for the Steiner TSP are presented. Section 3 reports on local search strategies that are explored by the GRASP with path-relinking heuristic presented in Section 4. Computational experiments are reported in Section 5 and extended in Section 6, where an improved strategy exploring periodical restarts is developed. Concluding remarks are drawn in the last section.

2 Greedy algorithms for the Steiner TSP

The following strategy can be applied as a heuristic for the Steiner TSP [7]. First, the instance of the STSP is reduced to a TSP instance in a complete graph defined by the set of required nodes, in which the new distances correspond to the shortest paths between every pair of required nodes in the original graph. Next, any exact or heuristic algorithm is used to solve the TSP in this new, complete graph. Finally, the solution of the TSP is converted into an STSP solution by expanding every edge by the corresponding shortest path between the two consecutive required nodes.

However, if the original STSP instance is a sparse graph, the conversion to a standard TSP instance significantly increases the number of edges, some of which may never be used. Therefore, instead of using a complete graph formed by the required nodes, we shall use the original graph for searching a minimum-weight closed walk. We use a straightforward adaptation of the nearest neighbor TSP adaptive greedy heuristic (see e.g. Chapter 3 of [10]) to the STSP described in Algorithm 1, which builds the solution greedily by choosing at each iteration the closest required node to that previously added to the walk.

The algorithm starts in line 1 by arbitrarily selecting any initial node $i \in V_R$ to start the walk. The set of required nodes $\mathcal{N}$ already visited by the walk is initialized in line 2. The partially built walk $\mathcal{P}$ is initialized in line 3. The currently visited required node $current$ is set in line 4. The loop in lines 5 to 11 is performed until all required nodes have been visited. At each iteration, the node $next$ to be visited is set to the closest among all yet unvisited required nodes. The shortest path $\mathcal{P}'$ from $current$ to $next$ is computed in line 7. The partially built walk $\mathcal{P}$ is updated in line 8 by appending the shortest path $\mathcal{P}'$ to it. The set of already visited required nodes $\mathcal{N}$ and the current node are updated in lines 9 and 10, respectively. Finally, after completing the loop, the shortest path from $current$ to the initial node i is appended to the walk in lines 12 and 13. The result is returned in line 14.

Algorithm 1 Nearest neighbor adaptive greedy heuristic for STSP.

```
1:  Select initial required node $i \in V_R$;
2:  $\mathcal{N} \leftarrow \{i\}$;
3:  $\mathcal{P} \leftarrow \{i\}$;
4:  $current \leftarrow i$;
5:  while $\mathcal{N} \neq V_R$ do
6:      $next \leftarrow$ closest node to $current$ among all those in $V_R \setminus \mathcal{N}$;
7:      $\mathcal{P}' \leftarrow$ shortest path from $current$ to $next$;
8:      $\mathcal{P} \leftarrow \mathcal{P} \oplus \mathcal{P}'$;
9:      $\mathcal{N} \leftarrow \mathcal{N} \cup \{next\}$;
10:     $current \leftarrow next$;
11: end while;
12: $\mathcal{P}' \leftarrow$ shortest path from $current$ to initial node $i$;
13: $\mathcal{P} \leftarrow \mathcal{P} \oplus \mathcal{P}'$;
14: return $\mathcal{P}$.
```

In the case of the Steiner TSP, the greedy criterion is the choice of the nearest required node to be visited.

Algorithms that add randomization to a greedy or adaptive greedy algorithm are called *semi-greedy* or *randomized greedy* algorithms. Randomization is an important feature in the implementation of effective heuristics. Semi-greedy algorithms act by replacing the deterministic greedy choice of the next element to be incorporated into the solution under construction by the random selection of an element from a restricted set of best candidate elements, called the restricted candidate list (RCL).

A simple quality-based scheme is used to define a restricted candidate list. Let $g_{\min} = \min\{g_i : i \in V_R \setminus \mathcal{N}$ and g_i is the shortest path from $current$ to node $i\}$ and $g_{\max} = \max\{g_i : i \in V_R \setminus \mathcal{N}$ and g_i is the shortest path from $current$ to node $i\}$. Furthermore, let α be such that $0 \leq \alpha \leq 1$. The RCL is formed by all yet unselected required nodes $i \in V_R \setminus \mathcal{N}$ satisfying $g_{\min} \leq g_i \leq g_{\min} + \alpha(g_{\max} - g_{\min})$.

3 Local search

Local search procedures are used to iteratively improve the quality of an initial solution, usually obtained by a constructive heuristic. First-improving and best-improving strategies are proposed and compared in terms of their performance. Efficient objective function updates are used, without the need of recalculating the objective function values from scratch: the weight of the previous walk is used in order to find that of the walk obtained after the changes performed during each iteration.

3.1 Neighborhood structure

The 2-opt neighborhood is the most commonly used neighborhood structure for the TSP problem and consists in replacing any pair of nonadjacent edges of the current solution by the unique pair of new edges that recreates a cycle.

The following property holds: Let $W = (v_1, \ldots, v_i, \ldots, v_j, \ldots, v_m)$ be any optimal solution of the Steiner TSP. Then, the subpath $(v_i, \ldots, v_j)$ is also a shortest path between the required nodes v_i and v_j. This is true because, if this subpath was not the shortest, then W would not be optimal. Therefore, it is not necessary to investigate moves that involve changes in the order in which the non-required nodes are visited. Then, the problem amounts to determining the order in which the required nodes should be visited and then finding the shortest path between any pair of consecutive required nodes in the walk.

In consequence, we explore a 2-opt neighborhood for the STSP that is formed by all moves that replace the paths between two pairs of consecutive required nodes in the walk by the two unique pairs of shortest paths that reconnect a closed walk.

3.2 Reduced 2-opt neighborhood

A reduced 2-opt neighborhood can be defined in order to take advantage of the problem structure. In fact, convergence can be faster if only a few, promising moves in the neighborhood are considered.

We implement this idea in the following way. For each required node v, let $I(v)$ be the set formed by all required nodes that are reachable from v by a shortest path that does not visit any other required node. In other words, $I(v)$ represent the set of required nodes that are closer to v, in the sense that they necessarily belong to paths to farther nodes.

Using this auxiliary data structure, we restrict the 2-opt moves to pairs of consecutive required nodes (v_1, v_2) and (w_1, w_2) satisfying the condition that $w_1 \in I(v_1)$.

4 GRASP with path-relinking heuristic

GRASP (which stands for *greedy randomized adaptive search procedures*) is a multi-start metaheuristic, in which each iteration consists of two main phases: construction and local search. The first phase is the construction of a feasible solution, usually by a greedy randomized algorithm. Once a feasible solution is obtained, its neighborhood is investigated until a local minimum is found during the second phase of local search. The best overall solution is kept as the result. The reader is referred to Resende and Ribeiro [10] for a complete account of GRASP.

We used the adaptive greedy randomized heuristic presented in Section 2 and the local search strategies described in Section 3 to customize a GRASP with path-relinking heuristic for the Steiner TSP.

Path-relinking is an intensification strategy that explores trajectories connecting elite solutions produced by metaheuristics. Path-relinking is usually carried out between two solutions: one is the initial solution S_i, while the other is the guiding solution S_g. A path that connects these solutions is constructed in the search for better solutions. Local search may be applied to the best solution in the path, since there is no guarantee that this solution is locally optimal. In the context of GRASP, path-relinking may be used to connect solutions obtained after the local search step with elite solutions produced during previous iterations, providing a sort of memory mechanism.

More specifically, in the context of the STSP, path-relinking attempts to preserve common characteristics of good walks, i.e. common subpaths. As explained below, path-relinking matches the positions of the largest common subpath to the initial and guiding solutions and then swaps the positions of nodes that do not belong to this common subpath.

We first observe that any solution of the STSP has no unique representation as a sequence of the visited required nodes, since any closed walk can start from different nodes and can be traversed in two directions (forward and backward). Therefore, the representation of the initial and guiding solutions must be adjusted to facilitate the operation of relinking them. With this purpose, before applying path-relinking, we adjust the representations of the initial and guiding solutions by detecting the largest common subpath $w_l = (v_i \dots v_j)$ between them.

In our implementation, we choose to detect the largest (or longest) common subpath, instead of the longest common subsequence, in order to prioritize consecutive sequences of nodes in both solutions. This problem is known as the largest (or longest) common substring (LCS) problem and can be solved in $O(n)$ time and space [5].

The guiding solution S_g and the initial solution S_i are oriented in the same direction according with w_l. Next, the initial nodes of the walks associated with S_g and S_i are made to coincide with the initial node v_i of w_l.

To move from the initial to the guiding solution, path-relinking considers a restricted neighborhood. Each move in this restricted neighborhood involves the swap of two required nodes in the walk corresponding to the current solution that are not in the same positions as they are visited in the guiding solution. In addition, each move should place at least one of the two involved nodes in the appropriate position corresponding to the order in which it will be visited in the guiding solution. After two required nodes are swapped, the shortest paths from their predecessors and to their successors are updated. Since at least one node is placed in the appropriate position of the guiding solution at each iteration, path-relinking will take at most as many iterations as the number of required nodes that were misplaced in the initial solution with respect to the guiding solution.

Algorithm 2 presents the pseudo-code of path-relinking from the initial solution S_i to the guiding solution S_g. The current solution S and the best solutions S^* are initialized in line 1. The cost f^* of the best solution found by path-relinking is initialized in line 2. The loop in lines 3 to 10 is performed until the current solution reaches the guiding solution. S' is set to the best solution in the restricted neighborhood of the current solution S in line 4. The best solution S^* found by path-relinking and its cost f^* are updated in lines 6 and 7, respectively, if the new solution S' improved the previous best. The current solution is updated in line 9 and a new path-relinking iteration resumes. The best solution found by path-relinking is returned in line 11.

Algorithm 2 Path-relinking algorithm for STSP.

```
1:  $S, S^* \leftarrow S_i$;
2:  $f^* \leftarrow cost(S_i)$;
3:  while $S \neq S_g$ do
4:      $S' \leftarrow$ best solution in the restricted neighborhood of $S$;
5:      if $cost(S') < f^*$ then
6:          $S^* \leftarrow S'$;
7:          $f^* \leftarrow cost(S')$;
8:      end if;
9:      $S \leftarrow S'$;
10: end while;
11: return $S^*$.
```

The pseudo-code in Algorithm 3 summarizes the main steps of the proposed GRASP with path-relinking (GRASP+PR) heuristic, following the same structure proposed in Section 9.3 of [10]. The set of elite solutions is initialized in line 1. The loop in lines 2 to 12 is performed until some stopping criterion is satisfied. An initial solution is built in line 3 by the greedy randomized constructive heuristic described in Section 2. A local search procedure is used in line 4 to improve the solution obtained at the end of the construction phase. Except for the first iteration, when the elite set is still empty, lines 6 to 9 amount to the application of path-relinking. Line 6 randomly selects an elite solution S' from the elite set $\mathcal{E}$. The representation of solution S is adjusted considering the selected elite solution S'. Backward path-relinking is applied from the initial solution $S_i = S'$ to the guiding solution $S_g = S$. Local search is applied in line 9 to the solution obtained by path-relinking. The elite set $\mathcal{E}$ is updated with the new solution S in line 11. The best elite solution is returned in line 13.

Algorithm 3 GRASP+PR algorithm for STSP

```
1:  $\mathcal{E} \leftarrow \emptyset$;
2:  while stopping criterion not satisfied do
3:      $S \leftarrow RandomizedGreedy$;
4:      $S \leftarrow LocalSearch(S)$;
5:      if $|\mathcal{E}| > 0$ then
6:          Select solution $S'$ at random from $\mathcal{E}$;
7:          $S \leftarrow AdjustRepresentation(S, S')$
8:          $S \leftarrow PathRelinking(S, S')$;
9:          $S \leftarrow LocalSearch(S)$;
10:     end if;
11:     $UpdateEliteSet(S, \mathcal{E})$;
12: end while;
13: Return the best solution $S$ in $\mathcal{E}$.
```

5 Computational experiments

Several experiments were performed to assess the performance of the algorithms presented above and their variants. The algorithms were implemented in C# programming language and compiled by Roslyn, a reference C# compiler, in a Intel Core i5 machine with a 2.9 GHz processor and 8 GB of random-access memory, running under the Windows 10 operating system.

We considered the same test problems used by Letchford et al. [6, 7] and Zhang et al. [12, 13], created by a random generator described in [7]. This generator was designed to create graphs that resemble real-life road networks. It creates connected sparse graphs and a fraction of required nodes is specified for each instance. In addition to the graphs from [7], we considered some larger instances, with up to 300 nodes. Altogether, ten sparse weighted graphs with 50 to 300 nodes were used to assess the performance of the heuristics. Each graph generated two instances: one with $\left\lfloor \frac{N}{3} \right\rfloor$ required nodes and another with $\left\lfloor \frac{2 \cdot N}{3} \right\rfloor$ required nodes, where N is the total number of nodes, corresponding to 20 different instances. We observe that individual optimal values for each of these instances have not been previously reported in [7]. Due to space limitations, we report here only numerical results for the instances with $\left\lfloor \frac{2 \cdot N}{3} \right\rfloor$ required nodes. Additional computational results will be reported in the final, extended version of this work.

5.1 Selecting the quality measure for the RCL

In order to compare the effect of the value of α in the quality-based scheme used to define a restricted candidate list, we ran the randomized nearest neighbor constructive heuristic with $\alpha = 0.1$ and $\alpha = 0.05$. The greedy randomized algorithm was applied to all instances. Average and best values over 100 runs are presented in Table 1. As for all tables that follow, the best solution values found for each instance are depicted in boldface. The randomized heuristic with $\alpha = 0.05$ found significantly more better solutions. We observe that the use of a better constructive method for building the initial solutions is likely to improve the quality of the solutions produced by the GRASP heuristic.

Table 1: Greedy randomized heuristic: average and best value over 100 runs.

	$\alpha = 0.10$		$\alpha = 0.05$	
Nodes	average	best	average	best
50	1207.51	**979**	1213.34	1031
75	1308.21	1125	1309.49	**1094**
100	1607.99	1375	1599.44	**1299**
125	1929.31	1627	1911.41	**1623**
150	2110.56	**1875**	2102.40	1899
175	2248.23	1970	2265.40	**1913**
200	2615.95	2369	2614.78	**2354**
225	2817.25	2524	2844.12	**2522**
250	3022.70	2748	2988.79	**2723**
300	3242.32	**2952**	3264.87	2977

5.2 Reduced 2-opt neighborhood

We now address the benefits of using the reduced 2-opt neighborhood, designed specifically for this problem. Preliminary computational experiments have shown that the use of this reduced neighborhood led to some target objective function values much faster than the use of the entire 2-opt neighborhood. These empirical observations were explored by the implementation of an alternative VND (variable neighborhood descent) local search procedure. First, only moves in the reduced 2-opt neighborhood are applied. The full neighborhood is explored only after a local minimum is obtained in the reduced 2-opt neighborhood.

Table 2 illustrates the efficiency of the VND approach when compared with the classic use of the full 2-opt neighborhood. It presents the solution values and the computation times in seconds for 100 iterations of the pure GRASP (without path-relinking) heuristic using both the pure 2-opt neighborhood and the VND approach for local search. In both cases, local search has been implemented following a best-improvement strategy.

Table 2: 2-opt vs reduced 2-opt neighborhoods, 100 GRASP iterations.

	2-opt neighborhood		reduced 2-opt	
Nodes	value	seconds	value	seconds
50	**978**	0.484	**978**	0.36
75	**1035**	1.078	1045	0.985
100	1239	2.359	**1208**	1.610
125	1496	4.454	**1469**	2.921
150	1643	7.468	**1615**	4.703
175	1743	11.313	**1719**	6.687
200	1976	17.469	**1925**	9.187
225	2094	24.891	**2047**	12.828
250	2224	32.546	**2170**	15.297
300	2409	55.718	**2285**	26.578

The VND local search strategy starting by the reduced 2-opt neighborhood led to the best solutions for nine out of the ten test problems. In addition, its computation times have been significantly smaller for all instances. As an example, in the case of the largest instance with 300 nodes, the time taken by 100 GRASP iterations using the VND local search strategy amounted to only 47.8% of the time taken when exclusively the complete 2-opt neighborhood is used. The GRASP heuristic using the VND local search strategy performs better both in terms of solution quality and computation times.

5.3 Probabilistic choice of α

We have already shown in Section 5.1 that, although choosing $\alpha = 0.05$ most often leads to better results than $\alpha = 0.10$ for the greedy randomized heuristic proposed for the STSP, for some instances the latter was a better choice than the former. Different probabilistic strategies were considered in [8] for the choice of the RCL parameter α, in contrast with the commonly used choice of fixing its value (see also [10]). It was shown that a randomly chosen α from a decreasing nonuniform discrete probability distribution offers a good compromise between the running time of the algorithm and the quality of the solutions produced by the randomized heuristic. We relied on this work to sustain that it can be a good choice, in addition to the previously used "good" value $\alpha = 0.05$, to consider also other higher values for this parameter with smaller probabilities of being chosen at each iteration. Therefore, in the following experiments, we used $\alpha = 0.05$ with a probability 70% and the values $\alpha = 0.10$ and $\alpha = 0.20$ with probabilities 20% and 10%, respectively.

5.4 Path-relinking

In this section, we address the impact of path-relinking in the search process. Table 3 shows the lengths of the solutions produced by the nearest neighbor adaptive greedy heuristic, by the pure GRASP heuristic running for 200 iterations (together with its running time in seconds), and by the GRASP with backward path-relinking running by the same time taken by 200 pure GRASP iterations. The randomized heuristic used in the construction phase of both the pure GRASP and GRASP with path-relinking algorithms makes use of the probabilistic criterion for the choice of α discussed in Section 5.3. The local search phase of both the pure GRASP and the GRASP with path-relinking algorithms was implemented using the best improvement VND strategy starting by the reduced 2-opt neighborhood. Path-relinking made use of elite sets formed by at most ten elements.

Table 3: GRASP and GRASP with path-relinking, 200 pure GRASP iterations.

	Greedy ($\alpha = 0$)	GRASP (200 iterations)		GRASP+PR (same running time)		
Nodes	value	value	seconds	value	seconds	iterations
50	1356	**978**	0.750	**978**	0.750	179
75	1204	1031	1.781	**1029**	1.782	183
100	1290	1211	3.484	**1193**	3.485	182
125	1915	1450	6.297	**1427**	6.313	196
150	1847	1615	9.812	**1567**	9.844	184
175	2085	1704	14.64	**1667**	14.641	191
200	2484	1952	19.828	**1895**	19.828	177
225	2549	2024	27.266	**1973**	27.281	191
250	2523	2165	35.078	**2080**	35.172	182
300	2759	2308	62.328	**2219**	62.609	193

Path-relinking considerably improved GRASP performance, leading to better solutions for all instances in the same running time and fewer iterations than the pure GRASP. Time-to-target plots for pure GRASP and GRASP with path-relinking algorithms for a 200-node instance are shown in Figure 1. The target value is 2000. Each algorithm was run 200 times. The plots in this figure provide empirical evidence that algorithm GRASP+PR outperforms pure GRASP for this instance and target value.

6 Restart strategies for GRASP with path-relinking

Resende and Ribeiro [9] have shown that restart strategies are able to reduce the running time to reach a target solution value for many problems. We apply the same type of restart(κ) strategy, in which the elite set is emptied and the heuristic restarted from scratch after κ consecutive iterations have been performed

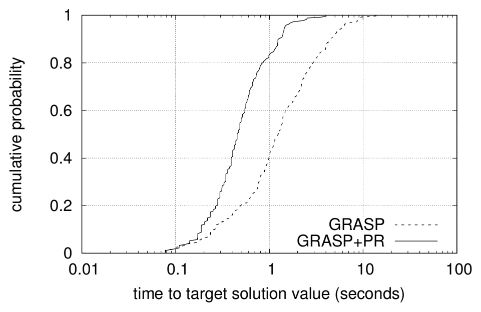


Figure 1: Time-to-target plot for 200-node instance and target value set to 2000.

without improvement in the best solution found. Computational results for restart strategies for STSP are displayed in Table 4, showing that they contribute to find better solutions in the same number of iterations, mainly when the problem size increases.

Table 4: Restart strategies for 1000 iterations.

	no restarts		restart(100)		restart(200)	
Nodes	value	seconds	value	seconds	value	seconds
50	**978**	4.03	**978**	3.96	**978**	3.95
75	**1029**	9.46	**1029**	9.45	**1029**	9.26
100	**1193**	18.95	**1193**	18.73	**1193**	18.43
125	1421	33.04	**1417**	33.93	1420	33.09
150	1565	53.39	1564	53.04	**1562**	52.23
175	1657	77.25	1665	76.84	**1652**	76.23
200	1883	105.53	1885	105.68	**1867**	105.20
225	1941	148.96	1953	144.68	**1928**	142.26
250	2054	173.68	2073	175.01	**2035**	176.04
300	2203	304.40	**2192**	310.03	2205	296.76

As previously observed in [9, 10], the effect of restart strategies can be mainly noticed in the longest runs. Considering the 200 runs for the 200-node instance with the target value set to 1900, they are associated with the column corresponding to the fourth quartile of Table 5. Entries in this quartile correspond to those in the heavy tails of the runtime distributions. The restart strategies in general do not affect too much the other quartiles of the distributions, which is a desirable characteristic. Compared to the norestart strategy, the restart(200) strategy was able to reduce not only the average running time in the fourth quartile, but also in the third and second quartiles. Consequently, strategy restart(200) performed the best among those tested, with the smallest average running times over the 200 runs.

7 Concluding remarks

In the Steiner TSP, one seeks a minimum-weight closed walk that visits a subset of required nodes. Since only a walk is sought, nodes can be visited more than once and edges may be traversed more than once.

We developed a GRASP with path-relinking and restarts for solving the Steiner Traveling Salesman Problem. The algorithm used in the construction phase is a randomized extension of the nearest

Table 5: Summary of computational results for each restart strategy for the 200-node instance: 200 independent runs were executed for each strategy. Each run was made to stop when a solution as good as the target solution value 1900 was found. For each strategy, the table shows the distribution of the running times by quartile. For each quartile, the table gives the average running times in seconds over all runs in that quartile. The average running times over the 200 runs are also given for each strategy.

	Average running times in quartile (seconds)				
Strategy	1st	2nd	3rd	4th	average
Without restarts	3.648	9.915	17.952	37.355	17.218
restart(100)	2.933	8.466	17.067	37.509	16.494
restart(200)	2.955	**8.093**	**15.410**	**34.878**	**15.334**

neighbor heuristic for the Traveling Salesman Problem. A variable neighborhood descent (VND) strategy exploring a reduced 2-opt neighborhood is used to optimize a best improving local search scheme. Path-relinking and restart strategies are used to improve the efficiency of the GRASP algorithm.

Extensive computational results for a set of instances previously used in the literature are reported. Since neither optimal values nor even upper bounds have been previously reported for these instances, the solutions obtained by the GRASP with path-relinking and restarts heuristic proposed in this work cannot be directly be compared with other solutions.

As a step towards avoiding this difficulty and facilitating the research on this problem, we made all test instances considered in this paper (together with their best known solutions and their costs) available at http://www2.ic.uff.br/˜rinterian/instances/allinstances.html. This website will be continuously updated with information provided by other researchers working on this problem with optimal values, upper bounds and best known feasible solutions for these and other benchmarking instances.

References

[1] S. Borne, A. R. Mahjoub, and R. Taktak. A branch-and-cut algorithm for the multiple Steiner TSP with order constraints. *Electronic Notes in Discrete Mathematics*, 41:487–494, 2013.

[2] G. Cornuéjols, J. Fonlupt, and D. Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33:1–27, 1985.

[3] B. Fleischmann. A cutting plane procedure for the travelling salesman problem on road networks. *European Journal of Operational Research*, 21(3):307–317, 1985.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[5] L. C. K. Hui. Color set size problem with application to string matching. In *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching*, CPM '92, pages 230–243. Springer-Verlag, 1992.

[6] A. N. Letchford and S. D. Nasiri. The Steiner travelling salesman problem with correlated costs. *European Journal of Operational Research*, 245:62–69, 2015.

[7] A. N. Letchford, S. D. Nasiri, and D. Oliver Theis. Compact formulations of the Steiner traveling salesman problem and related problems. *European Journal of Operational Research*, 228:83–92, 2013.

[8] M. Prais and C. C. Ribeiro. Parameter variation in GRASP procedures. *Investigación Operativa*, 9:1–20, 2000.

[9] M. G. C. Resende and C. C. Ribeiro. Restart strategies for GRASP with path-relinking heuristics. *Optimization Letters*, 5:467–478, 2011.

[10] M. G. C. Resende and C. C. Ribeiro. *Optimization by GRASP*. Springer, New York, 2016.

[11] J.-J. Salazar-González. The Steiner cycle polytope. *European Journal of Operational Research*, 147:671–679, 2003.

[12] H. Zhang, W. Tong, Y. Xu, and G. Lin. The Steiner traveling salesman problem with online edge blockages. *European Journal of Operational Research*, 243:30–40, 2015.

[13] H. Zhang, W. Tong, Y. Xu, and G. Lin. The Steiner traveling salesman problem with online advanced edge blockages. *Computers & Operations Research*, 70:26–38, 2016.

APPENDIX F – An Efficient Algorithm for Combining Verification and Validation Methods

An Efficient Algorithm for Combining Verification and Validation Methods

Isela Mendoza[1](✉), Uéverton Souza[1], Marcos Kalinowski[2], Ruben Interian[1], and Leonado Gresta Paulino Murta[1]

[1] Computer Institute, Fluminense Federal University, Niterói, Rio de Janeiro, Brazil
{imendoza, ueverton, rinterian, leomurta}@ic.uff.br

[2] Informatics Department, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, Brazil
kalinowski@inf.puc-rio.br

Abstract. An adequate combination of verification and validation (V&V) methods is important to improve software quality control throughout the development process and to reduce costs. However, to find an appropriate set of V&V methods that properly addresses the desired quality characteristics of a given project is a NP-hard problem. In this paper, we present a novel approach that combines V&V methods efficiently in order to properly cover a set of quality characteristics. We modelled the problem using a bipartite graph to represent the relationships between V&V methods and quality characteristics. Then we interpreted our problem as the Set Cover problem. Although Set Cover is considered hard to be solved, through the theoretical framework of Parameterized Complexity we propose an FPT-Algorithm (fixed-parameter tractable algorithm) that effectively solves the problem, considering the number of quality characteristics to be covered as a fixed parameter. We conclude that the proposed algorithm enables combining V&V methods in a scalable and efficient way, representing a valuable contribution to the community.

Keywords: Combination · Verification · Validation · Software quality FPT · Set cover · Parameterized Complexity

1 Introduction

Studies suggest high costs related to quality assurance activities in software development projects [1]. The appropriate combination of verification and validation (V&V) methods is seen in the literature as a way to reduce these costs and increase product quality [2]. Over the years, some knowledge has been generated regarding V&V methods when observed in isolation [3]. However, the selection of different V&V methods as well as the interdependencies among them are still not well-understood [4].

A significant part of the software industry is made up of small and medium-sized companies that, given the lack of guidelines for performing the right combination of V&V methods, have difficulties in optimizing this combination for their context, increasing the costs of resources and time and mainly harming the quality of the produced software.

B. Catania et al. (Eds.): SOFSEM 2019, LNCS 11376, pp. 324–340, 2019.
https://doi.org/10.1007/978-3-030-10801-4_26

According to the Guide to the Software Engineering Body of Knowledge (SWEBOK) [5], verification is used to ensure that the software product is built in the correct way, that is, it complies with the previously defined specifications. On the other hand, validation guarantees that the product is adherent to the user needs. It is known that an adequate combination of V&V methods outperforms any method alone [6]. Most of the studies presented in a systematic mapping [9] do not clearly specify which V&V methods cover which quality characteristics (e.g., considering the quality characteristics described in the ISO 25010 quality model standard).

Finding a set of methods that together properly addresses all quality characteristics of interest can be seen as a Set Cover Problem (SCP) [11]. The SCP is a classic NP-hard problem in the computational complexity area, whose decision version belongs to the list of the 21 Karp's NP-complete problems [11]. This means that when the number of methods or quality characteristics increase, the performance of an algorithm to aiming at combining them in an optimal way would drastically decrease.

The existence of efficient algorithms to solve NP-complete, or otherwise NP-hard, problems is unlikely, if the input parameters are not fixed; all known algorithms that solve these problems require exponential time (or at least super-polynomial time) in terms of the input size. However, some problems can be solved by algorithms for which we can split the running time into two parts: one exponential, but only with respect to the size of a fixed parameter, and another polynomial in the size of the input. Such algorithms are denoted FPT (fixed-parameter tractable) in the *Parameterized Complexity* field, because the problem can be solved efficiently for small values of the fixed parameter [13–15]. This field emerged as a promising alternative for working with NP-hard problems [12].

In this paper, we propose an algorithm to obtain an optimal combination of methods covering software quality characteristics of interest in reasonable computational time. In order to find an optimal solution for the problem (based on the desired software quality characteristics and the relation between those and V&V methods, provided as input), we adopted a parameterized approach, considering the set of quality characteristics as fixed parameter, and obtaining an algorithm classified as FPT. The implemented FPT algorithm is the first of its kind that solves the SCP.

Our proposed algorithm reached its goals: it runs in $O(f(k) \times n)$, where the constant k is the number of quality characteristics, n is the number of methods, and $f(k)$ is some function of k. Considering that the number of quality characteristics of a given quality standard is always constant, the algorithm runs in polynomial time in terms of the number of V&V methods to be combined. As a result, it provides the minimum set of V&V methods addressing all quality characteristics of interest. While this information is surely useful, we are aware that companies may choose to complement these methods with others to further assure the quality of the product (or even chose others) and that other factors, such as cost, should be considered when taking the final decision.

The remainder of this paper is organized in the following sections: Sect. 2 presents the background and related work concerning quality characteristics, V&V methods, and the combination of V&V methods. In Sect. 3 the problem is modeled as a SCP. Section 4 briefly introduces parameterized complexity theory. Section 5 presents the FPT–Algorithm that obtains the optimal combination. Section 6 contains a computational experiment analysis. Section 7 discusses the contributions and limitations of our approach. Section 8 presents the concluding remarks.

2 Background and Related Work

2.1 Quality Characteristics

Concerning software product quality characteristics, the product quality model defined in the ISO 25010 standard includes eight characteristics, for which quality requirements may be defined and measured during software development [10]. The characteristics and short descriptions for them, based on the ISO 25010 standard, can be found in Table 1.

Table 1. ISO 25010 quality characteristics.

Characteristic	Short description
Function suitability	Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions
Performance efficiency	Represents the performance relative to the amount of resources used under stated conditions
Compatibility	Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment
Usability	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use
Reliability	Degree to which a system, product or component performs specified functions under specified conditions for a specified period
Security	Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization
Maintainability	Degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements
Portability	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another

2.2 V&V Methods

Several V&V methods have been proposed over the years. In this paper we concentrate on a subset of V&V methods extracted mainly from the SWEBOK [5] and some other sources [19, 20] to compose our corpus. The list of methods can be found in Table 2. Due to space constraints, a short description of the methods is not provided, but it can be easily obtained in the cited sources. It is noteworthy that there are variations for each of these methods (e.g., different control flow-based criteria, different inspection techniques). Nevertheless, we use this more generic classification as a starting point, given that characterizing all possible variations to obtain a representative input for our algorithm would be hard to accomplish. Thus, a method covering a quality characteristic, in the context of this paper, means that there are ways of appropriately addressing it using the method.

Table 2. V&V methods.

Classification	Method
Based on intuition & experience	*Ad Hoc* Testing
	Exploratory Testing
Input domain-based	Equivalence Partitioning
	Pair wise Testing
	Boundary-Value Analysis
	Random Testing
	Cause-Effect Graphing
Code-based	Control Flow-Based Criteria
	Data Flow-Based Criteria
Fault-based	Error Guessing
	Mutation Testing
Usage-based	Operational Profile
	Usability Inspection Methods
Model-based	Finite-State Machines
	Workflow Models
Reviews	Walkthrough
	Peer Review or desk checking
	Technical Review
	Inspection

2.3 Combination of V&V Methods

It is known that the quality of software products is strongly dependent on the appropriate combination of V&V methods employed during development [2]. Experimental studies have long demonstrated that the use of combinations of different V&V methods to ensure the quality of a software is more effective than using isolated methods [7, 8].

Elbertzhager *et al.* [9] conducted a mapping study concerning the combination of V&V methods. They describe two fundamental approaches: *Compilation* and *Integration*. We focus on the *Compilation* approach since our purpose is purely to combine existing V&V methods (*Compilation* process). We are not focusing on *creating* new techniques by combining different methods into one, nor in using the results of the application of some technique as an instance to apply another one (*Integration* of V&V methods).

In order to establish how other works perform the combination of V&V methods in the *Compilation* approach, Elberzhager *et al.* [9] created a categorization to classify and organize these studies into three subgroups.

In the first subgroup, static and dynamic techniques are combined, focusing on thread escape analysis, atomicity analysis, protocol analysis, vulnerability analysis, concurrent program analysis or on defects in general. All these combinations are supported by open-source or proprietary tools.

The second subgroup compares different testing and inspection techniques discussing advantages and disadvantages among them. In most cases, two or three techniques are compared to each other. Several studies initially perform inspections, followed by some tests, corroborating then the effectiveness of the combination of both techniques.

The last subgroup describes other combinations, such as testing techniques and inspections combined with formal specifications, bug-finding tools, comprehensive quality control processes in industrial environments, comprising several inspections and technical tests, requirements and static analysis, tutorials, simulations, and vision-based approaches.

The most cited papers in the systematic mapping [9] regarding the *Compilation* approach are: Basili [22], Kamsties and Lott [23], and Wagner *et al.* [24]. Basili [22] makes a comparison of three software testing techniques: reading of code by gradual abstraction, functional testing using equivalence partition and border value analysis, and structural testing using total coverage of criticism, according to efficiency, cost, and fault detection classes. Kamsties and Lott [23], evaluate three techniques through a controlled experiment: reading of code by gradual abstraction, functional (black-box) testing and structural (white-box) testing. Wagner [24] describes a case study where several projects are analyzed in an industrial environment. In this project, automatic static analysis, testing, and reviews are used to detect defects. Their results show that these techniques complement each other and that they should be combined.

In the systematic mapping [9], papers were analyzed until 2010. This led us to carry out an update regarding the compilation approach, with the aim of finding more relevant and recent papers from 2010 to present. Due to space constraints the details of our mapping update will not be provided in this paper and we focus directly on the recent related work.

Dwyer and Elbaum [25] suggest an approach based on dividing V&V methods into two main classes: those that make dynamic analyses (or focused on behavior of the system, e.g., testing) and those that use static analysis (typically focused on a single property of the system at a time). Runeson *et al.* [27] compare code inspections and structural unit tests by analyzing three replications of an experiment in order to know which method finds more faults. Olorisade *et al.* [28] investigate the effectiveness of two test techniques (partition of equivalence class and decision coverage) and one review technique (code by abstraction) in terms of their ability to detect faults. Cotroneo *et al.* [29] combine testing techniques adaptively, based on machine learning, during the testing process, by learning from past experience and adapting the technique selection to the current testing session. Bishop *et al.* [30] combine a monotonicity analysis with a defined set of tests, showing that, unlike "independent" dynamic methods, this combination provides a full error coverage. Solari and Matalonga [31] study the behavior of two techniques, equivalence partition and decision coverage, to determine the types of defects that are undetectable for either of them. Finally, Gleirscher *et al.* [32] analyze three different techniques of automated static analysis: code clone detection, bug pattern detection, and architecture conformance analysis. They claim that this combination tends to be affordable in terms of application effort and cost to correct defects.

It is noteworthy that none of the related work has implemented something similar to our proposal, since we focus on covering a set of quality characteristics with few methods, thus obtaining an optimal combination of V&V methods. While applying all available methods represents a solution, this option might not be applicable due to cost constraints.

3 Modeling the Problem

In this section we describe how the problem of finding the smallest combination of methods that cover a specific set of quality characteristics can be modelled as a *Set Cover Problem*.

Consider C as the set of characteristics, and $N(m)$ as the subset of C that is covered by a specific method m. We need to *find the smallest set of subsets that cover* C. The problem is NP-hard in general. The relation between the characteristics and the methods can be modeled as an undirected bipartite graph as shown in Fig. 1.

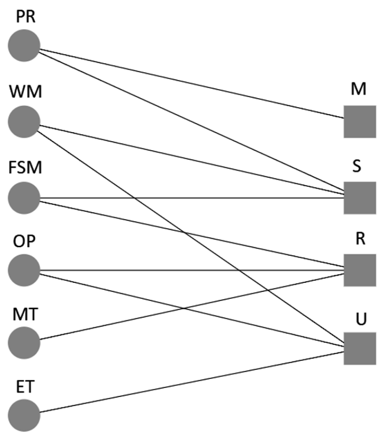


Fig. 1. Undirected bipartite graph. In the left-hand side the methods are positioned, and in the right-hand side the characteristics. Edges reflect the relationship between methods and characteristics. In the depicted instance, the set of methods contains the following elements: Peer Review (*PR*), Workflow Models (*WM*), Finite-State Machines (*FSM*), Operational Profile (***OP***), Mutation Testing (*MT*), and Exploratory Testing (*ET*). The set of characteristics is composed by four characteristics: Usability (**U**); Reliability (**R**); Security (**S**), and Maintainability (**M**). The graph shows a scenario in which method *PR* covers characteristics **M** and **S**, *WM* covers **S** and **U**, *FSM* covers **S** and **R,** *OP* covers **R** and **U,** *MT* covers **R**, and *ET* covers **U**.

The example instance was obtained from the results of a survey [21] that collected the opinion of experts about these V&V methods. The experts answered about their agreement on the suitability of the methods to address the different quality attributes of the ISO 25010 standard. The relationship between some method *m* and some characteristic *c* is obtained from the median of the survey answers (1 – disagree, 2 – partially disagree, 3 – partially agree, 4 – agree). In this example we considered that *m* properly covers *c* if the median is bigger or equal to 3. I.e., only methods that cover a quality characteristic to a certain degree will have edges in the graph. The outcome of the survey relating the quality characteristics to the V&V method can be seen in more details in [21].

For illustrative purposes we built this graph instance taking a subset of our real data, considering only four quality characteristics and some of the methods that can be used to properly address them according to the answers of the respondents (19 experts from 7 different countries, all with PhDs in software engineering, active in major software engineering and V&V venue committees, and with relevant publications in the area of V&V). The example perfectly serves our illustrative purposes to present the V&V method combination algorithm. Actually, this smaller example allows providing a better understanding of the algorithm's execution and correctness.

4 Parameterized Complexity

The *Parameterized Complexity* field emerged as a promising way to deal with NP-hard problems [12, 26]. It is a branch of the Computational Complexity Theory that focuses on classifying computational problems according to their hardness with respect to different parameters of the input. The complexity of a problem is mainly expressed through a function of these parameters.

The theory of NP-completeness was developed to identify problems that cannot be solved in polynomial time if $P \neq NP$. However, several NP-complete and NP-hard problems still need to be solved in practice.

For many problems, only super-polynomial time algorithms are known when the complexity is measured according to the size of the input, and in general, they are considered "intractable" from the theoretical point of view assuming that *P* is different from *NP*. Nevertheless, for several problems we can develop algorithms in which we can split the running time into a part computed in polynomial time with respect to the size of the input and another part computed in at least exponential time, but only with respect to a parameter *k*. Consequently, if we set the parameter *k* to a small value and its growth is relatively small we could consider these problems as "manageable" and not "intractable" [13–15].

Thus, an important question arises: *"Do these hard problems admit non-polynomial time algorithms whose exponential complexity part is a function of merely some aspects of the problem?"* [12]. The existence of such algorithms was analyzed by Downey and Fellows in [13], and is briefly discussed in the next section.

4.1 Fixed-Parameter Tractable (FPT) Approach

The fixed-parameter tractable (FPT) approach [13] considers the following format for the problems: *"Given an object x and a non-negative integer k, the goal is to determine whether x has some property that depends on k?"* The parameter *k* is considered small compared to the size of x. The relevance of these parameters lies precisely in the small range of values they can take, being a very important factor in practice [12].

The FPT-algorithms sacrifice the execution time, which can be exponential, but guarantee that the exponential dependency is restricted to the parameter k, which means that the problem can be solved efficiently for small values of that fixed parameter. The use of these algorithms provides a more rigorous analysis of problem's time complexity since this complexity is generally obtained from the size of the input [12].

Formally, a problem Π belongs to the class *FPT* (it is fixed-parameter tractable) with respect to a parameter k if it admits an algorithm to solve it whose running time is of the form: $f(k) \times n^a$, where a is a constant, and f is an arbitrary computable function. Note that whenever k is bounded by a constant we have $f(k) = O(1)$, hence the running time of the algorithm will be polynomial.

Finally, for the problem of this paper, we present a fixed-parameter tractable algorithm where the size k of the set of characteristics to be covered is the parameter. I.e., we are limiting the complexity by the number of relevant product characteristics to be considered when developing the software.

4.2 Scalability of the FPT-Algorithms

Scalability is the ability of a system or process to handle an increasing amount of data [16]. Computer algorithms can be called scalable if they are efficient when applied to large instances, i.e., instances with a large size of the input [17].

We can say that FPT-algorithms are scalable because they are efficient when executed in large instances. These algorithms take advantage of the specific structure of the instances, which is a differential when compared to exact or exhaustive search algorithms that require high computational time.

It is important to note that the studied problem can handle a large number of V&V methods, given that the number of quality characteristics tends to be relatively small. Therefore, an FPT-algorithm with respect to the number of characteristics to be covered will produce a tool for combination of V&V methods with high scalability.

Indeed, in our problem, the number of quality characteristics is already a known small integer (in the ISO standard this number is 8). Therefore, scalability relies on the ability to find the optimal solution even if the number of considered methods is growing. Our initial set comprises 19 methods, but additional methods have been reported by the survey respondents and our algorithm allows to efficiently work, for example, with 30, 50, or 100 methods.

5 FPT–Algorithm to Combine V&V Methods

The goal of the algorithm, shown in the Fig. 2, is to obtain the optimal combination (smallest number) of V&V methods that properly cover all the relevant quality characteristics for the product to be developed. Certainly, a software organization could complement the resulting set with other V&V methods that cover similar quality characteristics to find more defects and to further enhance quality, but at least they would know about the minimum set of methods to consider in order to address all the quality characteristics that are relevant for the product to be developed. I.e., a combination such that there is a method properly addressing (i.e., with an edge in the graph for) each relevant quality characteristic and none of them remains uncovered.

The objective function is the number of selected methods that properly cover all the characteristics. The parameter to be set is the number of the selected quality characteristics. In this way, we are parameterizing the *Set Cover Problem* by the number of characteristics to be covered by the V&V methods.

Coming up next, we present some definitions that are used in the algorithm presented in Fig. 2:

C – set of characteristics.
M – set of methods.
$N(m)$ – set of characteristics covered by the method m.
$P(c) = \{x \in M : c \in N(x)\}$ – set of methods that cover the characteristic c.
$R(m) = \{x \in M : N(x) \subseteq N(m)\}$ – set of methods that cover a subset of $N(m)$.

The input parameters are the set of characteristics C and the set of methods M. The redundant methods are removed in line 6 by using a simple preprocessing step. It removes methods that cover a subset of characteristics covered by any other method. A characteristic c is selected from the set of characteristics in line 7. The algorithm then focuses on selecting the method that will cover c in the optimal solution. In line 8, the

```
**Algorithm 1** Set Cover Algorithm, **Parameters:** sets C, M
1: $M^* \leftarrow M$
2: $f^* = |M|$
3: **if** $C = \emptyset$ **then**
4:     **return** $\emptyset$
5: **else**
6:     $M \leftarrow RemoveSubsets(M)$
7:     $c \leftarrow SelectCharacteristic(C)$
8:     $C_t \leftarrow C \setminus \{c\}$
9:     **for all** $m \in P(c)$ **do**
10:        $M' \leftarrow \{m\}$
11:        $C_t \leftarrow C_t \setminus N(m)$
12:        $M_t \leftarrow M \setminus R(m)$
13:        **while** $\exists c' \in C_t : |P(c') \cap M_t| = 1$ **do**
14:            Let $m' \in P(c')$
15:            $C_t \leftarrow C_t \setminus N(m')$
16:            $M_t \leftarrow M_t \setminus \{m'\}$
17:            $M' \leftarrow M' \cup \{m'\}$
18:        **end while**
19:        $M'^* \leftarrow SetCover(C_t, M_t)$
20:        **if** $|M'^*| + |M'| < f^*$ **then**
21:            $f^* \leftarrow |M'^*| + |M'|$
22:            $M^* \leftarrow M'^* \cup M'$
23:        **end if**
24:        $C_t \leftarrow C \setminus \{c\}$
25:    **end for**
26: **end if**
27: **return** $M^*$
```

Fig. 2. Pseudocode of the set cover FPT-algorithm.

variable C_t that contains the characteristics to be covered is initialized. A loop runs through all the methods that cover c in lines 9–25. The set M' that stores the methods that will be part of a feasible solution is initialized with method m in line 10. The set of characteristics to cover C_t is updated in line 11 by removing the characteristics already covered by m. The set M_t, containing the methods available to cover C_t, is initialized in line 12 with all methods of M except those covering a subset of $N(m)$. In lines 13–18 a loop is executed while there are characteristics c' that are covered by a single method m'. The variables C_t, M_t and M' are updated in lines 15–17. The available M_t methods and the characteristics that have not been covered until now are used to obtain an optimal sub-problem solution by recursively calling the *SetCover* algorithm. In line 19, the obtained optimal solution is stored in $M^{'*}$. If the methods selected in M' together with the optimal solution $M^{'*}$ of the sub-problem improve the optimum value found so far (f^*), then f^* and M^* are updated in lines 20–23. The value of C_t is reinitialized in line 24. The best solution found (M^*), is returned as the optimal solution to the problem in line 27.

5.1 Execution of the Set Cover Algorithm

Taking the graph represented in Fig. 1 as the entry of the algorithm, we now illustrate the execution of the pseudocode. After initialization steps 1–5, line 6 removes redundant methods. In this case, methods *MT* and *ET* are removed, because they cover only one characteristic, already covered by other methods. The result is shown in Fig. 3.

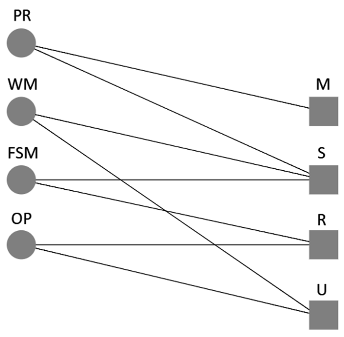


Fig. 3. Algorithm execution. State of the graph after the preprocessing step.

Afterwards, the first characteristic **M** is chosen as c, and all the methods that cover **M** must be considered in the loop that begins on line 9. Therefore, method PR is selected. In line 11, we remove all the characteristics already covered by PR, that is, **M** and **S**. The variable M_t gets the set of methods $\{WM, FSM, OP\}$ in line 12. Since there are no characteristics covered by only one method, the loop on lines 13–18 does not perform any action, and the algorithm is called recursively in line 19 with set of characteristics $\{\mathbf{R}, \mathbf{U}\}$, and set of methods $\{WM, FSM, OP\}$ as parameters. Figure 4 illustrates the graph at this stage.

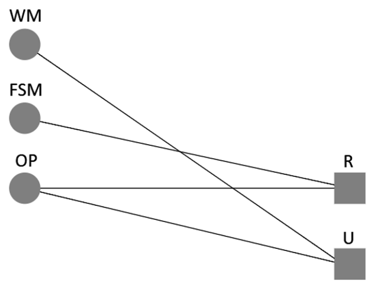


Fig. 4. Algorithm execution. State of the graph after the recursive call.

Finally, the algorithm is executed again from the beginning. Methods *WM* and *FSM* are immediately removed as redundant, and the remaining method *OP* is selected to cover the last two characteristics. The variables C_t and M_t became empty, and in the next recursive call, the stopping criterion is reached. The *OP* method is returned as a solution of the instance represented in Fig. 4, forming the final solution of the whole instance together with already selected method *PR*. The smallest set of methods M^* is set as $\{PR, OP\}$, and the optimal value f^* is set to 2.

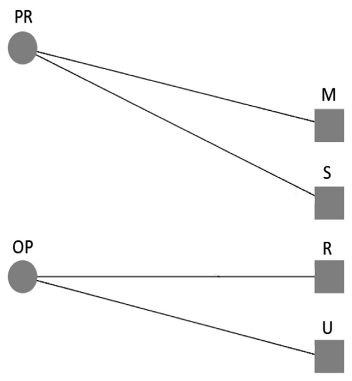


Fig. 5. Methods that form the optimal solution returned by the algorithm when executed in the graph. If **M** is selected as the first characteristic at the beginning of the execution, then the optimal set of methods returned by the algorithm is $\{PR, OP\}$.

As can be observed from the execution, the algorithm considers all the possible ways of covering the quality characteristics, keeping the most efficient ones. In this sense, the obtained solution can be considered optimal for the problem and model we pose. In fact, the algorithm is able to determine the optimal combination (smallest number) of V&V methods that properly cover quality characteristics of interest for a product to be developed based on any initial graph configuration connecting V&V methods to the quality characteristics they properly address (Fig. 5).

5.2 Running Time Analysis

Suppose that there are n methods in the set M, and there are k characteristics in the set C, being k some small integer. We note that a naive (brute force) algorithm would test all solutions (subsets of the set M) and chose which of them cover C having smaller size. Because there are 2^n subsets of the set M, this naive algorithm has a time complexity of $O(2^n)$. This exponential order is intractable even for some relatively small values of n, like 30 or 40 (which could easily be achieved when including specific variations of the V&V methods as input).

Instead, the proposed algorithm tries to determinate which method is the best option to cover each characteristic. After choosing some characteristic c, the algorithm tries to select each method that properly covers c, covering the rest of characteristics recursively. Because the number of methods that cover each characteristic is at most n, the order of this algorithm can be initially bounded by $O(n^k)$. In general, this order is already better than the 'naive' solution.

Nevertheless, we improve the upper bound of our algorithm's running time by refining the actual number of methods it will analyze. In fact, there are only 2^k different ways of covering a set C of k elements. If there is more than 2^k methods, then necessarily there will be two of them that cover exactly the same set of characteristics. That means that these two methods would be indistinguishable to our algorithm; that is, if they cover the same characteristics, any one of them can be used. Using this fact, we can successively preprocess the input, improving the algorithm performance from $O(n^k)$ to $O(f(k) \times n)$, where $f(k) \leq (2^k)^k = 2^{k^2}$. In our case, $k = 8$ and this means that $f(k)$ is bounded above by a constant, i.e., $f(k) = O(1)$. Then, we have a linear algorithm for the problem instead of an exponential or even a $O(n^8)$-time algorithm.

Once again, the upper bound for $f(k)$ is improved (decreased) using the fact that if some method m_1 covers a subset of characteristics covered by some other method m_2, then m_1 can be removed from the set of methods. This is because if m_1 is actually chosen, you can instead choose m_2, since m_2 is 'better' method in the sense that it covers all that m_1 covers, and possibly more. Lubell [18] showed that there are no more than $\binom{k}{\lfloor k/2 \rfloor}$ combinations with the property that no one is a subset of the other. This implies that for $k = 8$, there can be much less than 2^k different methods with the property that there is no method that covers a subset of characteristic of some other method. In particular, for $k = 8$ there can be at most $\binom{8}{4} = 70$ methods satisfying this property, and there can be at most $\binom{7}{3} = 35$ of these methods covering one common characteristic. Therefore, the redundant methods are removed by using a simple preprocessing step that searches for methods that cover a subset of characteristics covered by any other method. At each iteration of the algorithm, the number of characteristics to be covered decreases and the previous steps of the algorithm are repeated considering a decremented k value.

Summarizing, it holds that:

$$f(k) < \binom{k-1}{\lfloor (k-1)/2 \rfloor} \times \binom{k-2}{\lfloor (k-2)/2 \rfloor} \times \cdots \times \binom{2}{1} \qquad (1)$$

For $k = 8$, it follows that $f(k) < 35 \times 20 \times 10 \times 6 \times 3 \times 2$, then our $O(f(k) \times n)$-time algorithm is an efficient (linear) algorithm where $f(k)$ is bounded above by a constant. In practice, this constant is even lower, and does not depend on the number of existing methods, which produces scalability with respect to the number of methods to be worked.

6 Computational Experiments

Several experiments were performed to assess the algorithm presented above. The algorithm was implemented in C# programming language and compiled by Roslyn, a reference C# compiler, in an Intel Core i3 machine with a 2.0 GHz processor and 4 GB of random-access memory, running under the Windows 10 operating system.

A number of test problems created by a random generator is considered. Each test problem has two parameters: the number of vertices n and the probability p of a method to cover a characteristic.

The FPT–algorithm is also executed on the instance obtained from the survey described in [21], according to the criteria explained in the *Sect.* 3 when we build the example with a subset of this same data.

Table 3 shows the optimal solution sizes and execution times (in seconds) for the FPT-Algorithm solver with and without instance preprocessing (Fig. 2, line 6) and a naive algorithm (brute force), for each instance. The name of the instance indicates the number of methods, followed by the probability of a characteristic to be covered by a method, in percent. The optimal solution sizes (number of methods returned) are equal for all instances, indicating the correctness of the Algorithms.

The FTP–Algorithm with the preprocessing is more efficient than without preprocessing and the Naive Algorithm, obtaining the result in less than 0:01 s in all cases even for the biggest instances, unlike the other algorithms in which for some instance sizes the solution is not found in a reasonable waiting time (—). The FTP–Algorithm without processing proves to be, in turn, more efficient than the Naive Algorithm, executing more instances with better runtime.

For the instance obtained from the survey [21], the FPT–Algorithm with the processing of the instance returned the methods: 12, 19 and the FPT–Algorithm without the processing and the Naive Algorithm returned the methods: 12, 18. The solution returned by the FPT–Algorithm with the processing contains the method 19 that covers a superset of the characteristics covered by the method 18 in the others algorithm solutions, showing that FPT–Algorithm with preprocessing performs better when concerning coverage, by using this additional comparison criterion.

Table 3. Computational experiments

Instance	Runtime FPT-Alg (with pre-processing)	Optimal solution FPT-Alg (with pre-processing)	Runtime FPT-Alg (no pre-processing)	Optimal solution FPT-Alg (no pre-processing)	Runtime Alg-Naive (brute force)	Optimal Alg-Naive (brute force)
Instance_20_10	0.00031	5	0.00375	5	0.14907	5
Instance_20_20	0.00047	3	0.01094	3	0.17703	3
Instance_20_50	0.00062	2	0.08859	2	0.22297	2
Instance_50_10	0.00031	5	0.14359	5	—	—
Instance_50_20	0.00110	2	4.97453	2	—	—
Instance_50_50	0.00094	2	11.4025	2	—	—
Instance_100_10	0.00094	2	64.7025	2	—	—
Instance_100_20	0.00125	3	—	—	—	—
Instance_100_50	0.00110	2	—	—	—	—
Instance_200_10	0.00344	3	—	—	—	—
Instance_200_20	0.00188	2	—	—	—	—
Instance_200_50	0.00094	1	—	—	—	—
Instance_500_10	0.00359	3	—	—	—	—
Instance_500_20	0.00469	2	—	—	—	—
Instance_500_50	0.00234	1	—	—	—	—
Instance_1000_10	0.00766	3	—	—	—	—
Instance_1000_20	0.00578	2	—	—	—	—
Instance_1000_50	0.00500	1	—	—	—	—

7 Discussion

Our proposed algorithm is effective, being able to provide the optimal combination (smallest number) of V&V methods properly covering a set of chosen quality characteristics to be considered when developing a software product. Additionally, it is more efficient than brute-force or exhaustive search algorithms and its execution time properties match the particularities of the problem well. Indeed, the algorithm can be applied to instances of different sizes, making our approach scalable, i.e., suitable for larger case studies (for instance, considering more V&V methods, including specific variations of the more generic methods used for our sample).

There is, however, a basic assumption for applying the algorithm, which is having a defined input with information on which V&V methods properly address the different quality characteristics. For illustrative purposes, our example was based on initial outcomes of an expert survey. It is also noteworthy that our set of 19 V&V methods represents generic methods for which several variations are available (e.g., applying specific testing criteria or variations of inspection methods). While they perfectly fit our illustrative example and allowed us getting feedback from experts on whether they can be employed to properly address quality characteristics, information on more specific methods could be provided as input to combination algorithm. We highlight that this initial configuration is out of the scope of the intended contribution of this paper and that companies could use an initial configuration based on their own sets of evidence on the V&V methods they typically use or on their own elicited expert beliefs.

Moreover, from a practical point of view, companies might decide to complement the optimal solution provided by the algorithm by applying additional V&V methods that cover similar quality characteristics (e.g., aiming at finding additional defects and further enhancing product quality), in particular for critical projects. However, using our approach at least they would know about a minimum set of methods that would allow them avoiding neglecting quality characteristics that are relevant for the product to be developed.

Also, specialists on software engineering economics might argue that our solution providing the smallest number of V&V methods is not considering the cost of applying each method. However, to address this issue we would need to know the relative cost among the V&V methods and this information is extremely context specific and hard to generalize. We are aware of this limitation and further addressing it is part of our future work. A solution option to handle this issue when using the approach described in this paper would be removing the methods that are cost restrictive from the initial configuration.

8 Concluding Remarks

In this paper, we modeled the problem of finding a combination of V&V methods to cover software quality characteristics as the Set Cover problem, a NP-hard combinatorial optimization problem. We defined a parameterized FPT algorithm that is specially designed for our instances, since typically the number of considered quality characteristics is small. Provided by a valid input, the proposed algorithm is able to efficiently provide an optimal combination (smallest number) of V&V methods properly covering a set of chosen quality characteristics to be considered when developing a software product. Additionally, we showed that it is more efficient than Naive (brute-force) algorithms. Furthermore, the algorithm can be applied to instances of different sizes, making our approach scalable, i.e., suitable for larger studies (for instance, considering more V&V methods).

Our future works consist of development of a support tool that, given a set of selected quality characteristics and an initial configuration (e.g., from the survey results, or any other source such as within-company expert belief elicitation), provide the optimal combination of V&V methods. Finally, for now we focused on product quality, and a next step would be to integrate cost-related issues into the approach. Moreover, we believe that the Fixed-Parameter Tractable algorithm approach can be applied to solve other problems in the software engineering domain and that sharing our V&V method combination experience with the community could foster discussions towards other graph theory-based solutions for relevant software engineering problems.

Acknowledgment. The authors would like to thank CNPq and FAPERJ for the financial support (Project N^{o}. E-26/010.001578/2016, Title: "*Resolution of Critical Problems of the Software Industry through Graph Theory and its algorithms"*). Thanks also to the survey respondents, which provided us the initial configurations to test our approach.

References

1. Meyers, G.J., Badgett, T., Thomas, T., Csandler, C.: The Art of Software Testing, 3rd edn. Wiley, Hoboken (2011). ISBN 978-1118031964
2. Feldt, R., Torkar, R., Ahmad, E., Raza, B.: Challenges with software verification and validation activities in the space industry. In: Third International Conference on Software Testing, Verification and Validation (ICST) (2010)
3. Boehm, B., Basili, V.: Software defect reduction top 10 list. IEEE Softw. **34**(1), 135–137 (2001)
4. Feldt, R., Marculescu, B., Schulte, J., Torkar, R., Preissing, P., Hult, E.: Optimizing verification and validation activities for software in the space industry. In: Data Systems in Aerospace (DASIA), Budapest (2010)
5. Bourque, P., Fairley, R.E.: SWEBOK guide V3.0, guide to the software engineering body of knowledge. IEEE Computer Society (2004)
6. Endres, A., Rombach, D.: A Handbook of Software and Systems Engineering. Addison Wesley, Reading (2003)
7. Myers, G.J.: A controlled experiment in program testing and code walkthroughs/inspections. Commun. ACM **21**(9), 760–768 (1978)
8. Wood, M., Roper, M., Brooks, A., Miller, J.: Comparing and combining software defect detection techniques: a replicated empirical study. In: Jazayeri, M., Schauer, H. (eds.) ESEC/SIGSOFT FSE-1997. LNCS, vol. 1301, pp. 262–277. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63531-9_19
9. Elberzhager, F., Münch, J., Nha, V.T.N.: A systematic mapping study on the combination of static and dynamic quality assurance techniques. Inf. Softw. Technol. **54**(1), 1–15 (2012)
10. ISO25000 Software Product Quality, ISO/IEC 25010, Official site (2011). http://iso25000.com/index.php/en/iso-25000-standards/iso-25010
11. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum, New York (1972)
12. dos Santos, V.F., dos Santos Souza, U.: Uma Introdução à Complexidade Parametrizada. In: Anais da 34^{o} Jornada de Atualização em Informática, CSBC, pp. 232–273 (2015)
13. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999). https://doi.org/10.1007/978-1-4612-0515-9
14. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-29953-X
15. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, Oxford (2006)
16. Bondi, A.B.: Characteristics of scalability and their impact on performance. In: Proceedings Second International Workshop on Software and Performance WOSP, pp. 195–203 (2000)
17. Laudon, K.C., Traver, C.G.: E-commerce: Business, Technology, Society. Stanford University, Stanford (2008)
18. Lubell, D.: A short proof of Sperner's lemma. J. Comb. Theory **1**(2), 299 (1996)
19. Wagner, S.: Software Product Quality Control. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38571-1

20. Wiegers, K.E.: Peer Reviews in Software: A Practical Guide, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2002)
21. Mendoza, I., Kalinowski, M., Souza, U., Felderer, M.: Relating verification and validation methods to software product quality characteristics: results of an expert survey. In: 11th Software Quality Days (SWQD). Lecture Notes on Business Information Processing, Vienna, Austria. Springer (2019, to appear)
22. Basili, V.R.: Comparing the effectiveness of software testing strategies. IEEE Trans. Softw. Eng. **13**(12), 1278–1296 (1987)
23. Kamsties, E., Lott, C.M.: An empirical evaluation of three defect-detection techniques. In: Schäfer, W., Botella, P. (eds.) ESEC 1995. LNCS, vol. 989, pp. 362–383. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60406-5_25
24. Wagner, S., Jürjens, J., Koller, C., Trischberger, P.: Comparing bug finding tools with reviews and tests. In: Khendek, F., Dssouli, R. (eds.) TestCom 2005. LNCS, vol. 3502, pp. 40–55. Springer, Heidelberg (2005). https://doi.org/10.1007/11430230_4
25. Dwyer, M.B., Elbaum, S.: Unifying verification and validation techniques: relating behavior and properties through partial evidence. In: FSE/SDP Workshop on Future of Software Engineering Research (FOSE), Santa Fe, New Mexico, USA, pp. 93–98 (2010)
26. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
27. Runeson, P., Stefik, A., Andrews, A., Grönblom, S., Porres, I., Siebert, S.: A comparative analysis of three replicated experiments comparing inspection and unit testing. In: Proceedings 2nd International Workshop on Replication in Empirical Software Engineering Research (RESER), Banff, AB, Canada, Article No. 6148335, pp. 35–42 (2012)
28. Olorisade, B.K., Vegas, S., Juristo, N.: Determining the effectiveness of three software evaluation techniques through informal aggregation. Inf. Softw. Technol. **55**(9), 1590–1601 (2013)
29. Cotroneo, D., Pietrantuono, R., Russo, S.: A learning-based method for combining testing techniques. In: Proceedings 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA, Article No. 6606560, pp. 142–151 (2013)
30. Bishop, P., Bloomfield, R., Cyra, L.: Combining testing and proof to gain high assurance in software: a case study. In: IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), Pasadena, CA, USA, Article No. 6698924, pp. 248–257 (2013)
31. Solari, M., Matalonga, S.: A controlled experiment to explore potentially undetectable defects for testing techniques. In: Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE), Canada, pp. 106–109 (2014)
32. Gleirscher, M., Golubitskiy, D., Irlbeck, M., Wagner, S.: Introduction of static quality analysis in small- and medium-sized software enterprises: experiences from technology transfer. Softw. Qual. J. **22**(3), 499–542 (2014)

APPENDIX G – DyeVC: an approach for monitoring and visualizing distributed repositories

Cesario *et al. Journal of Software Engineering Research and Development* (2017) 5:5
DOI 10.1186/s40411-017-0039-8

Journal of Software Engineering Research and Development

RESEARCH **Open Access**

DyeVC: an approach for monitoring and visualizing distributed repositories

Cristiano Cesario, Ruben Interian and Leonardo Murta*

* Correspondence:
leomurta@ic.uff.br
Instituto de Computação, Universidade Federal Fluminense (UFF), Niteroi, RJ, Brazil

Abstract

Software development using distributed version control systems has become more frequent recently. Such systems bring more flexibility, but also greater complexity to manage and monitor multiple existing repositories as well as their myriad of branches. In this paper, we propose DyeVC, an approach to assist developers and repository administrators in identifying dependencies among clones of distributed repositories. It allows understanding what is going on around one's clone and depicting the relationship between existing clones. DyeVC was evaluated over open source projects, showing how they could benefit from having such kind of tool in place. We also ran an observational and a performance evaluation over DyeVC, and the results were promising: it was considered easy to use and fast for most repository history exploration operations while providing the expected answers.

Keywords: Distributed version control, Monitoring, Visualization, Awareness

1 Background

Version Control Systems (VCS) date back to the 70s when SCCS emerged (Rochkind 1975). Their primary purpose is to keep software development under control (Estublier 2000). Along these almost 40 years, VCSs have evolved from a centralized repository with local access (e.g., SCCS and RCS (Tichy 1985)) to a client-server architecture (e.g., CVS (Cederqvist 2005) and Subversion (Collins-Sussman et al. 2011)). More recently, distributed VCSs (DVCS) arose (e.g., Git (Chacon 2009) and Mercurial (O'Sullivan 2009a)) allowing clones of the entire repository in different locations. According to a survey conducted by the Eclipse community (2014), Git and GitHub combined usage increased from 6.8 to 42.9% between 2010 and 2014 (a growth greater than 500%). During this same period, Subversion and CVS combined usage decreased from 71 to 34.4%. This clearly shows momentum and a strong tendency in the adoption of DVCSs in the open source community.

Besides these changes from local to client-server and then to a distributed architecture, the concurrency control policy adopted by VCSs also changed from lock-based (pessimistic) to branch-based (optimistic). According to Walrad and Strom (Walrad and Strom 2002), creating branches in VCSs is essential to software development because it enables parallel development, allowing the maintenance of different versions of a system, the customization to different platforms/customers, among other features. DVCSs include better support for working with branches (O'Sullivan 2009b), turning the branch creation into a recurring pattern, no matter if this creation is explicitly done by executing a "branch" command or implicitly when a repository is cloned.

However, distributed software development, especially from the geographical perspective (Gumm 2006), brings a set of risk factors, and Configuration Management (CM) is affected by them. The increasing growth of development teams and their distribution along distant locations, together with the proliferation of branches, introduce additional complexity for perceiving actions performed in parallel by different developers. According to Perry et al. (1998), concurrent development increases the number of defects in software. Besides, da Silva et al. (2006) say that branches are frequently used for promoting isolation among developers, postponing the perception of conflicts that result from changes made by co-workers. These conflicts are noticed only after pulling changes in the context of DVCSs. Moreover, Brun et al. (2011) show that even using modern DVCSs, conflicts during merges are frequent, persistent, and appear not only as overlapping textual edits (i.e., physical conflicts) but also as subsequent build (i.e., syntactic conflicts) and test failures (i.e., semantic conflicts).

By enabling repository clones, DVCSs expand the branching possibilities discussed by Appleton et al. (1998), allowing several repositories to coexist with fragments of the project history. This may lead to complex topologies where changes can be sent to or received from any clone. This scenario generates traffic similar to that of peer-to-peer applications. In practice, projects impose some restrictions on this topology freedom. However, it can be still much more complex than the traditional client-server topology found in centralized VCS.

With this diversity of topologies, managing the evolution of a complex system becomes a tough task, making it difficult to find answers to the following questions:

- Q1: Which clones were created from a repository?
- Q2: What are the communication paths among different clones?
- Q3: Which changes are under work in parallel (in different clones or different branches) and which of them are available to be incorporated into others' clones?

Most of the existing works, such as Palantir (Sarma and van der Hoek 2002), FASTDash (Biehl et al. 2007), Lighthouse (da Silva et al. 2006), CollabVS (Dewan and Hegde 2007), Safe-Commit (Wloka et al. 2009), Crystal (Brun et al. 2011), and WeCode (Guimarães and Silva 2012), deal with question Q3, giving to the developers awareness of concurrent changes. However, they do not provide an overview of the topology of repositories, indicating which commits belong to which clones. This overview is essential to understand the distributed evolution of the project.

To answer the questions above, we propose DyeVC,[1] a novel monitoring and visualization approach for DVCS that gathers information about different repositories and presents them visually to the user. DyeVC allows developers to perceive how their repository evolved over time and how this evolution compares to the evolution of other repositories in the project. DyeVC's main goal is two-fold: increasing the developers' knowledge of what is going on around their repository and the repositories of their teammates, and enabling repository administrators to visualize the relationship between existing clones. DyeVC was evaluated over open source projects, showing how they could benefit from having such kind of tool in place. We also ran an observational and a performance evaluation over DyeVC, and the results were promising: it was considered easy to use and fast for most repository history exploration operations while providing the expected answers.

This paper extends a previous conference paper (Cesario and Murta 2016) by including a more thorough discussion of our approach, including how DyeVC discovers the topology and a formal definition of the process underneath DyeVC. Moreover, as the previous version of DyeVC struggled when dealing with large repositories (over 6500 commits), we also added an automatic collapsing feature. This new feature provides a dual contribution: it allowed DyeVC to deal with larger repositories and reduced cluttering when presenting information to users. The performance evaluation was expanded to present an assessment of the automatic collapsing feature. Finally, we included a deeper comparison of DyeVC with its related work. This paper is organized as follows: Section 2 shows a motivational example. Section 3 presents the DyeVC approach. Section 4 presents the technologies used in our prototype implementation. Section 5 describes the evaluation of DyeVC. Section 6 discusses related work, and Section 7 concludes the paper and presents some suggestions for future work.

2 Motivational example

Figure 1 shows a scenario with some developers, each one owning a clone of the repository created at Xavier Institute. Xavier Institute acts like a central repository, where code developed by all teams is integrated, tested, and released to production. There is a team working at Xavier Institute, led by Professor Xavier, and a remote developer (Storm) that periodically receives updates from the Institute. Outside the Institute, Wolverine leads a remote team located in a different site, which is constantly synchronized with the Institute. Solid lines in Fig. 1 indicate data being pushed, whereas dotted lines indicate data being pulled. Thus, for example, Rogue can both pull updates from Gambit and push updates to him, and Beast can pull updates from Rogue, but cannot push updates to her.

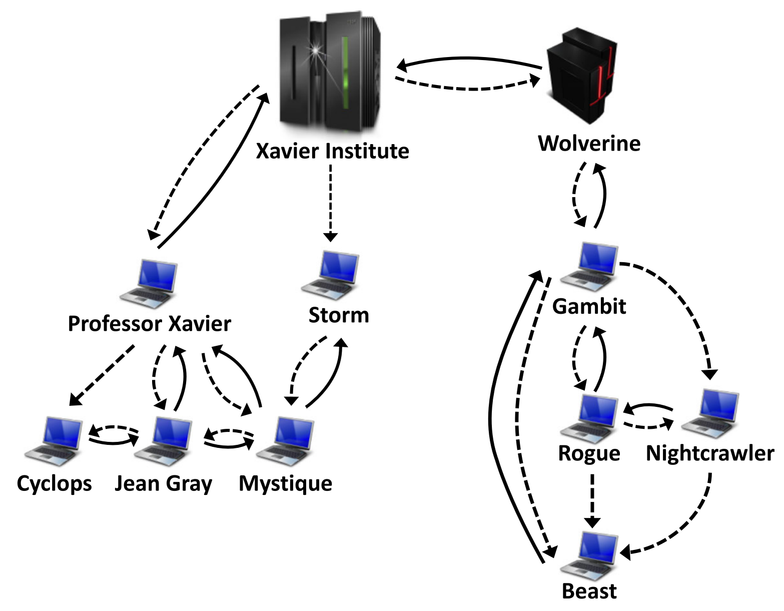


Fig. 1 Development scenario involving some developers

Each one of the developers has a complete copy of the repository. Luckily, this scenario has a CM Plan in action. Otherwise, each one would be able to send and receive updates to and from any other, leading to a total of $n \times (n - 1)$ different possibilities of communication (where n is the number of developers in the topology). In practice, however, this limit is not reached: while interaction amongst some developers is frequent, it may happen that others have no idea about the existence of some coworkers. It occurs with Mystique and Nightcrawler, for example, where there is no direct communication.

As an example, from a developer's point of view, like Beast, questions such as the following can arise:

- How can he know at a given moment if there are commits in Rogue, in Gambit, or in Nightcrawler clones that were not pulled yet? Suppose that Beast is working on a feature that depends on a utility class developed by Gambit. If such class has a bug, and Gambit is working to solve it, Beast would want to know when Gambit's commit is ready to be pulled. Moreover, if Gambit is evolving such class, and Beast has this information, he could decide to anticipate a pull to incorporate Gambit's changes in his workspace.
- Would it be the case that local commits are pending to be pushed to Gambit? Beast could certainly periodically pull changes from his peers, checking if there were updates available, but this would be a manual procedure, prone to be forgotten. It would be more practical if Beast could have an up to date knowledge of his peers, warning him about any local or remote updates that had not been synchronized yet.

On the other hand, from an administrator's point of view, questions such as the following are pertinent:

- How can she knows which are the existing clones of a project and how they relate to each other? This is a common need to repository administrators. It helps not only in identifying who must be notified regarding any news related to the repository but also helps in visually verifying if pull/push policies are being followed by the team. Having a map of all existing clones can help repository administrators in identifying who is pushing to / pulling from each other. For instance, unauthorized access to push to a production repository can be visualized, and the administrator can take actions to revoke such access.
- How can she know if there are pending commits to be sent from a staging repository to a production one? Having the ability to know how many commits are pending and which commits are these can help administrators decide if this is the right time to release a new version of the system to production.

3 DyeVC approach

Aiming at supporting both developers and repository administrators in understanding the interaction among repository clones, the main features of DyeVC include: (1) a mechanism to gather information from a set of clones (such as their relationships and known commits) and (2) a set of extensible views with different levels of detail, which let DyeVC users visualize this information. We detail in the following sub-section how

DyeVC gathers information from DVCSs. Next, we discuss how this information is presented using different levels of detail. Finally, we show what happens behind the scenes, discussing the algorithm involved in the data synchronization process.

3.1 Information gathering

DyeVC continuously gathers information from interrelated clones, starting from clones registered by the user. Figure 2 shows a deployment view of DyeVC's architecture. For each clone *rep* that the user registers to monitor, DyeVC transparently creates a local clone *rep'* in the user's home folder to fetch data from all of the peers with which *rep* communicates. Data is gathered by DyeVC instances running at each user machine and is stored in a central document database. In this way, information from one DyeVC instance is made available to every other instance in the topology.

DyeVC gathers information from registered clones in the user's machine and also from their peers, which are clones that communicate with them. Since there is a communication path between a registered clone and its peers (either to push data or to pull data), we can analyze the commits that exist in these peers. This allows us to present a broader topology visualization that contains not only registered clones, but also those that have a push or pull relationship with them. DyeVC finds out related clones by looking at the remote repositories registered in the DVCS configuration. More details on how data is gathered are explained in section 3.3.

Figure 3 shows how DyeVC discovers the topology from the nodes where it is running and the registered clones. Blue nodes represent registered clones where DyeVC is running, yellow nodes represent known clones located at nodes where DyeVC is not running, dashed nodes and dashed lines represent clones and communication paths, respectively, that are not known yet. Suppose a scenario where the existing clones and interdependencies are shown in Fig. 3a, which depicts the same scenario shown in Section 2 but here represented by the first letter of each clone. After installing DyeVC and

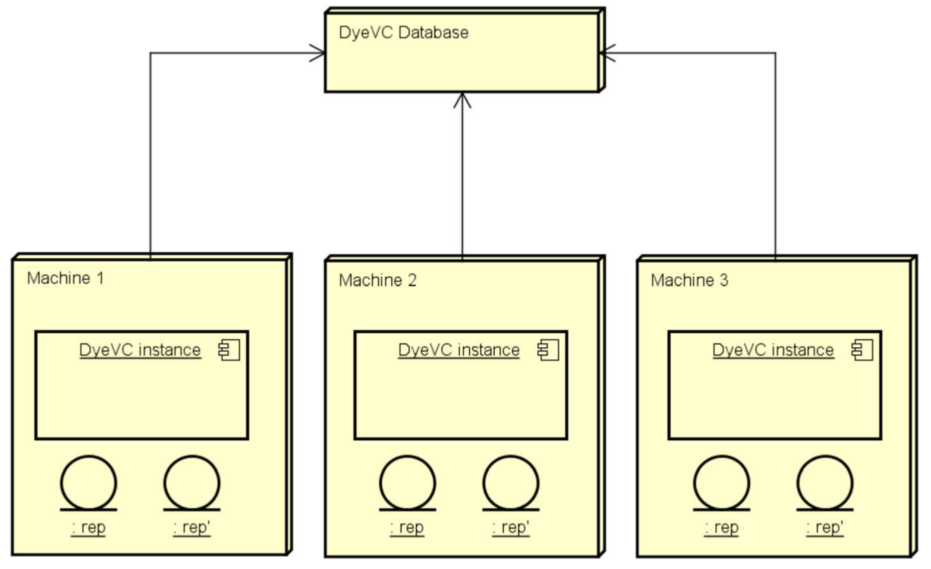


Fig. 2 How DyeVC gathers information

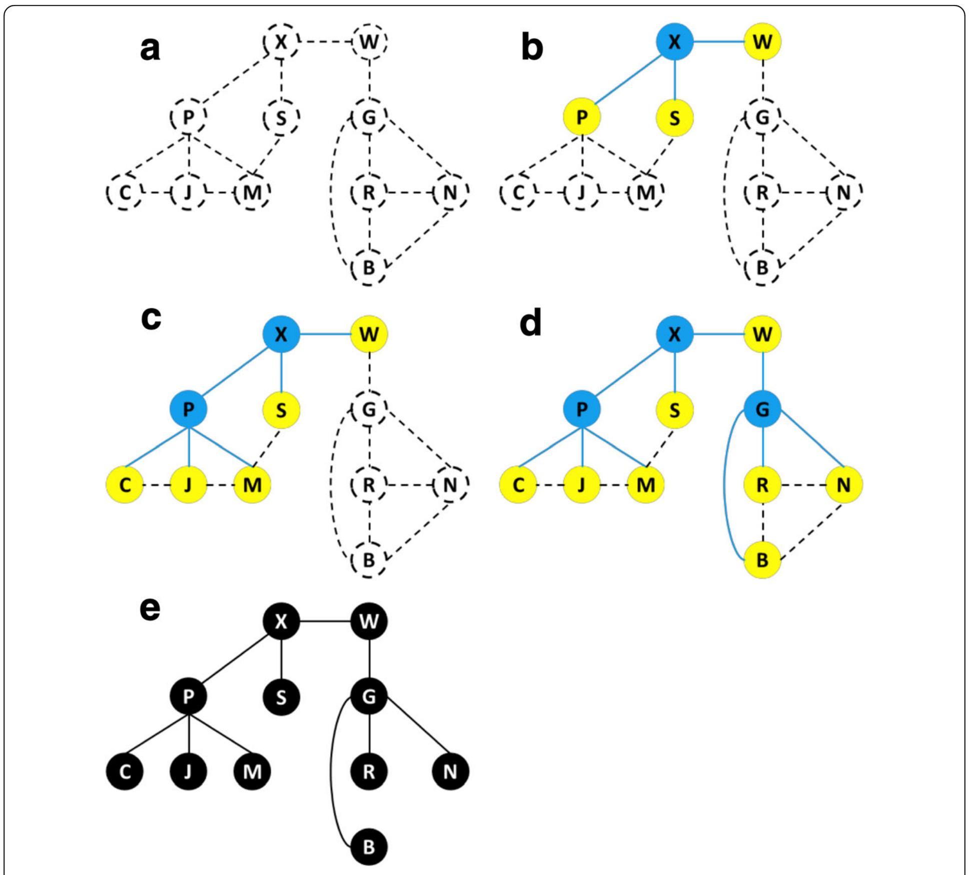
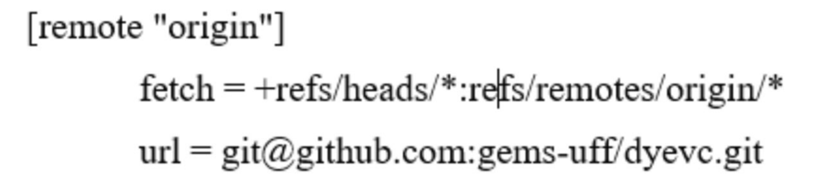


Fig. 3 DyeVC discovering the topology: actual topology (**a**), discovered topology with X (**b**), discovered topology with X and P (**c**), discovered topology with X P and G (**d**), final discovered topology (**e**)

registering clone *X*, DyeVC finds out that this clone communicates with clones *W*, *P*, and *S* (either by pushing to or pulling from them), as shown in Fig. 3b. Later on, clone *P* is registered and clones *C*, *J* and *M* are included as known clones in the topology (Figure 3c). Clone *G* is the next to be registered, allowing DyeVC to discover that clones *R*, *N*, and *B* also exist, as well as the communication between clone *G* and clone *W*, which was already a known clone (Fig. 3d). Assuming that no other clones are registered, the known topology is shown in Fig. 3e. Notice that, although only clones *G*, *P*, and *X* were registered, DyeVC is also aware of the existence of clones *B*, *C*, *J*, *M*, *N*, *S* and *W*. Only some communication paths between clones will not be known (*C-J, J-M*, *S-M, R-B*, *R-N* and *N-B*).

DyeVC finds out related clones by looking at the remote repositories, which are registered in Git's *config* file of each clone. Figure 4 shows an example of this configuration, taken from a local clone of the *DyeVC* project, where there is a remote named *origin*, which is located at *github.com/gems-uff/dyevc*. This information is in the *url* parameter, which indicates to Git that pushes and pulls use the same location. If there were a *pushurl* parameter in the configuration, besides the *url* parameter, pulls would use the location in the *url* parameter and pushes would use the location in the *pushurl* parameter.

Data stored in the central database follows the metamodel presented in Fig. 5. A *Project* groups repository clones of the same system. Clones are stored as *RepositoryInfo*



```
[remote "origin"]
        fetch = +refs/heads/*:refs/remotes/origin/*
        url = git@github.com:gems-uff/dyevc.git
```

Fig. 4 Remote repository configuration in Git's *config* file

and are identified by an *id* and a meaningful clone name provided by the user (*cloneName* attribute). A *RepositoryInfo* has a list of clones to which it pushes data and a list of clones from which it pulls data. These lists are represented respectively by the self-associations *pushesTo* and *pullsFrom*. Finally, a *RepositoryInfo* stores the *hostName* where it resides (e.g., a server name or localhost), its *clonePath* (be it an operating system path or an URL) and the set of DyeVC instances that have registered it to be monitored (*monitoredBy* attribute).

Branches are part of a *RepositoryInfo*. A *Branch* has a name and a boolean attribute *isTracked*, which is true if the branch tracks a remote branch. A *RepositoryInfo* may have one or many branches (it must have at least one branch, which is the main one). A *Branch* has two associations with *CommitInfo*: through the first association, a *Branch* knows which commit is its *head* and, conversely, a commit knows which branches point to it as a head (*headOf* association end). The second association represents which commits are reachable from a given branch (*reachableCommits* association end) and, conversely, the branches from which the commit is reachable (*reachableFrom* association end).

The finer grain of information is the *CommitInfo*, which represents each commit in the topology. A commit is identified by a hash code (*hash* attribute) and refers to its

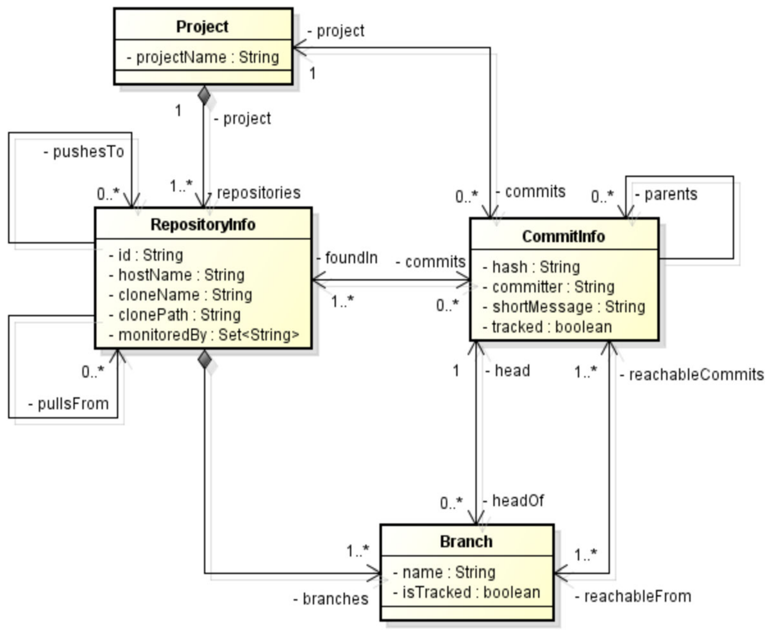


Fig. 5 Metamodel used to store DyeVC data

parents (except for the first commit in the repository, which does not have any parent). As each commit may not exist in all clones of the topology, we store the list of clones where each commit can be found (*foundIn* association end). We also store the *committer*, the commit message (*shortMessage* attribute), and whether the commits belong to tracked or non-tracked branches (*tracked* attribute).

3.2 Information visualization

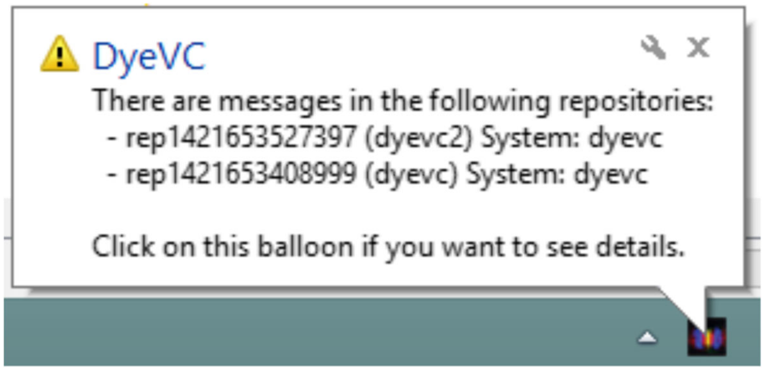
DyeVC presents information at four different levels of detail: Level 1 shows high-level notifications about registered repositories; Level 2 shows the whole topology of a given project. Level 3 zooms into the branches of the repository, showing the status of each tracked branch. Lastly, Level 4 zooms into the commits of the repository, showing a visual log with information about each commit. The following sections discuss these levels.

3.2.1 Level 1: Notifications

In Level 1, our approach periodically monitors registered repositories and presents notifications whenever a change is detected in any known peer. The period between subsequent runs is configurable, and notifications are presented in the system notification area, in a non-obtrusive way. Figure 6 shows an example of this kind of notification, where DyeVC detected changes in two different repositories. The notification shows the repository id, the clone name, and the project (system) name. Clicking on the balloon opens DyeVC main screen.

3.2.2 Level 2: Topology

Aiming at helping to answer questions Q1 and Q2, we present a topology view showing all repositories for a given project (Fig. 7), where each node represents a known clone. A blue computer represents the current user clone, and black computers represent other clones where DyeVC is running. Servers represent central repositories that do not pull from nor push to any other clone, or clones where DyeVC is not running. Both kinds of nodes use the same representation because, once DyeVC is not running at a given clone, we cannot infer the *pushesTo* and *pullsFrom* lists, which will thus be empty as in a server. At first sight, this could be understood as a risk within topology view. However, DyeVC considers servers as clones. The denomination “server” is just to visually differentiate it from other clones. We believe that plotting servers and clones where


DyeVC
There are messages in the following repositories:
- rep1421653527397 (dyevc2) System: dyevc
- rep1421653408999 (dyevc) System: dyevc

Click on this balloon if you want to see details.

Fig. 6 DyeVC showing notifications in notification area

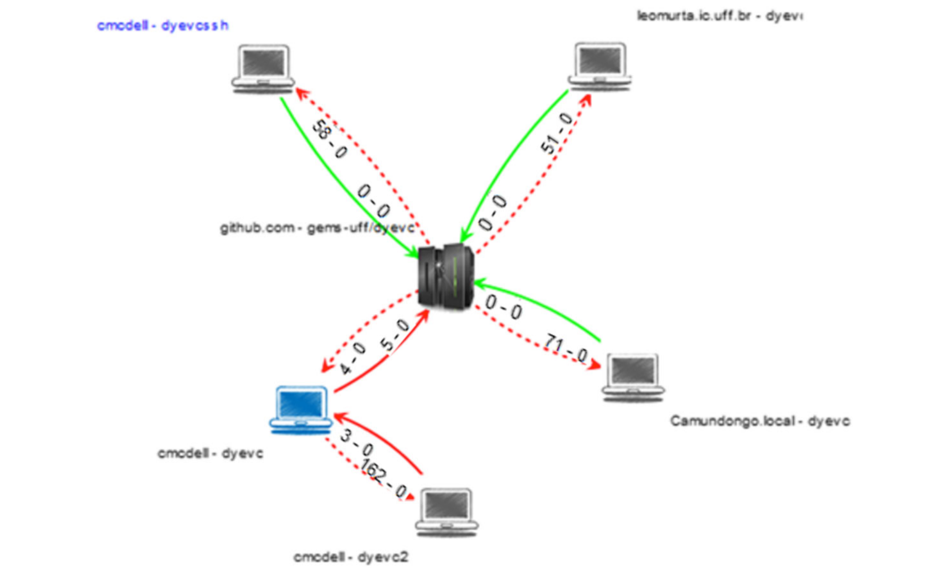


Fig. 7 Topology view for a given project

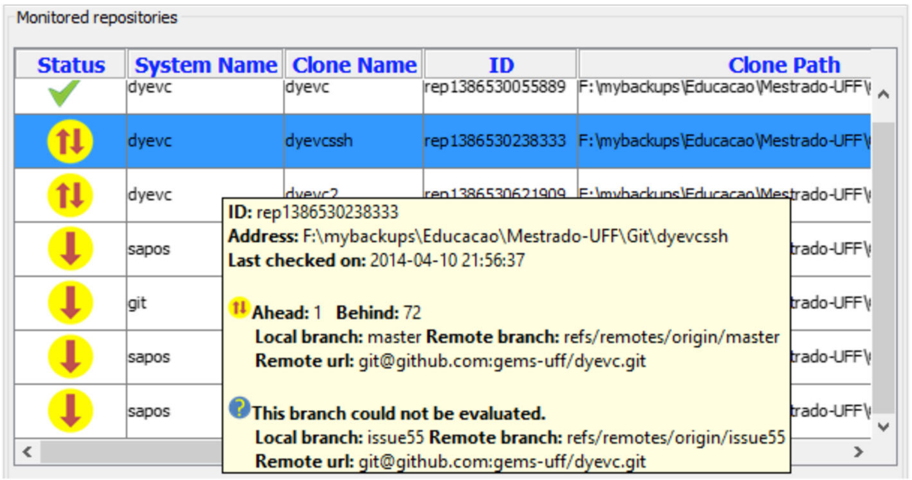
DyeVC is not running with the same icons is not a risk because the topology view brings more information about the clones (e.g., clone address and name). Thus, a repository administrator can distinguish the servers among the plotted clones.

Each edge in the graph represents a relationship between two repositories. Continuous edges mean that the source clone pushes to the destination clone, whereas dashed edges mean that the destination clone pulls from the source clone. The edge labels show two numbers separated by a dash. The first and second numbers represent how many commits in tracked and non-tracked branches of the source clone are missing in the destination clone, respectively. The edge colors are used to represent the synchronization status: green edges mean that both clones are synchronized (i.e., the destination clone has all the commits present in the source clone), whereas red edges mean that the pair is not synchronized and indicates the direction that is missing commits. For example, it is possible to observe in Fig. 7 that the current user clone (blue computer) is hosted at *cmcdell* and is named *dyevc*. This clone pulls from *gems-uff/dyevc*, which is located at *github.com*, and there are four tracked commits ready to be pulled (i.e., commits that exist in the remote repository and do not exist locally). It also pushes to the same peer, having five tracked commits ready to be pushed. In this case, both edges are red, which raises attention to investigate further what is happening, because such situation may lead to integration conflicts.

3.2.3 Level 3: Tracked branches

For helping answering question Q3, DyeVC's main screen (see Fig. 8) shows Level 3 information, allowing one to view the status of each tracked branch in registered repositories regarding their peers. This information is complemented with that of Level 4, shown in the next section.

The status evaluation considers the existing commits in each repository individually. Due to the nature of DVCS, old data is almost never deleted, and commits are cumulative. Thus, if commit *N* is created over commit *N* – *1*, the existence of commit *N* in a



Monitored repositories

Status	System Name	Clone Name	ID	Clone Path
✓	dyevc	dyevc	rep1386530055889	F:\mybackups\Educacao\Mestrado-UFF\
⇅	dyevc	dyevcssh	rep1386530238333	F:\mybackups\Educacao\Mestrado-UFF\
⇅	dyevc	dyevc2	rep1386530621909	F:\mybackups\Educacao\Mestrado-UFF\
↓	sapos			trado-UFF\
↓	git			trado-UFF\
↓	sapos			trado-UFF\
↓	sapos			trado-UFF\

ID: rep1386530238333
Address: F:\mybackups\Educacao\Mestrado-UFF\Git\dyevcssh
Last checked on: 2014-04-10 21:56:37

⇅ **Ahead:** 1 **Behind:** 72
Local branch: master **Remote branch:** refs/remotes/origin/master
Remote url: git@github.com:gems-uff/dyevc.git

? **This branch could not be evaluated.**
Local branch: issue55 **Remote branch:** refs/remotes/origin/issue55
Remote url: git@github.com:gems-uff/dyevc.git

Fig. 8 DyeVC main screen

given repository implies that commit *N – 1* also exists in the repository. In this way, by using set theory, it is possible to subtract the set of commits in the local repository from the set of commits in its peers, resulting in the set of commits not pulled yet. In this case, local repository will be *behind* its peers (arrow down in Fig. 8). Conversely, subtracting the sets in the inverse order will result in the set of commits not pushed yet, meaning that local repository is ahead of its peers (arrow up). When both sets are empty, local repository is synchronized (green checkmark in Fig. 8) and when both sets have elements, it is both ahead and behind its peer (arrow up and down in Fig. 8).

Let us assume that each commit is represented by an integer number to illustrate how our approach works. At a giving moment, the local repositories of each developer have the commits shown in Table 1. Consider the synchronization paths presented in the right-hand side of Fig. 1, where the perception of each developer regarding their known peers is shown in Table 2. Notice that the perceptions are not symmetric. For instance, as Gambit does not pull updates from Nightcrawler, there is no sense in giving him information regarding Nightcrawler. Furthermore, it is uncommon to have a scenario where pushes are performed from a developer to another (such as the one between Beast and Gambit). What happens is that a developer pulls from another (for example, between Gambit and Nightcrawler), avoiding inadvertent inclusion of commits inside others' clones. Although infrequent, this scenario helps in understanding the need to have awareness about who are the peers in a project and what are their interdependencies.

3.2.4 Level 4: Commits

Level 4 complements information of Level 3 to provide an answer to Question Q3. Differently from the usual repository version graph, it presents a combined version graph of the entire topology (Fig. 9). Each vertex in the graph represents either a

Table 1 Existing commits in each repository

Repository	Wolverine	Gambit	Rogue	Nightcrawler	Beast
Commits	10; 11	10; 11	10; 12	10; 11; 13	10

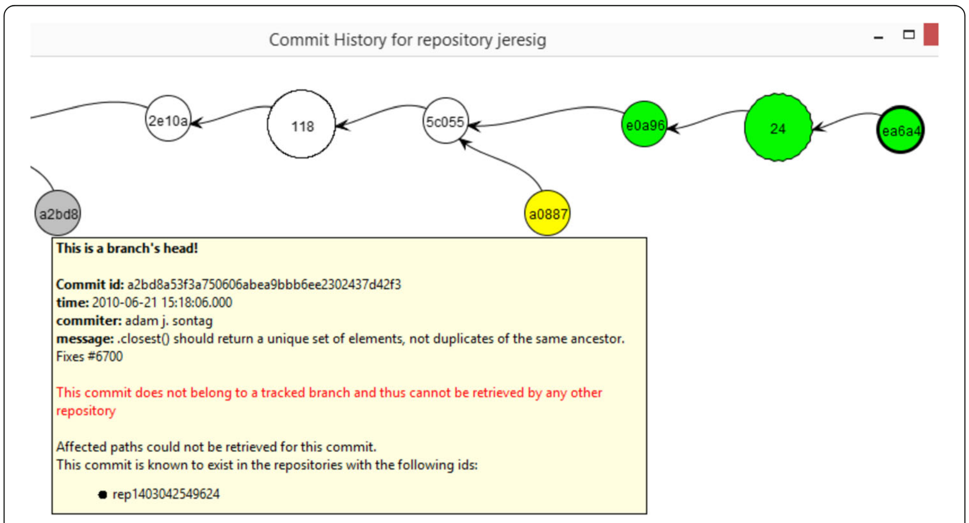
Table 2 Status of each repository based on known remote repositories

Repository	Wolverine	Gambit	Rogue	Nightcrawler	Beast
Wolverine	-	-	-	-	-
Gambit	✓	-	-	-	-
Rogue	-	⇅	-	-	-
Nightcrawler	-	✓	⇅	-	-
Beast	-	⬇	⬇	⬇	-

known commit in the topology, which is named after its hash's five first characters (e.g., the node labeled *2e10a* in Fig. 9), or a collapsed node, representing several commits blended. We implement two ways of collapsing nodes to provide a better understanding over huge amounts of data: manual and automatic. Manually collapsed nodes are named after the number of contained nodes, such as the white node containing 118 commits and the green node containing 24 commits in Fig. 9). Automatically collapsed nodes have ellipses before and after the number of contained nodes in their names (if the first collapse of Fig. 9 were automatic, its name would be "…118…"). Automatic collapsing is detailed in Section 3.2.5.

Thicker borders denote that the commit is a branch's head (e.g., commit *ea6a4*). Commits are drawn according to their precedence order. Thus, if a commit *N* is created over a commit *N – 1*, then commit *N* will be located to the right of commit *N – 1*. For each commit, DyeVC presents the information described in Fig. 5 (gathered from the central database), along with information that is read in real time from the repository metadata, such as branches that point to that commit and affected files (added, edited, and deleted).

This visualization contains all commits of all clones in an integrated graph. Each commit is painted according to its existence in the local repository and the peers' repositories. Ordinary commits that exist locally and in all peers are painted in white. Green commits are ready to be pushed, as they exist locally but do not exist in peers



This is a branch's head!

Commit id: a2bd8a53f3a750606abea9bbb6ee2302437d42f3
time: 2010-06-21 15:18:06.000
commiter: adam j. sontag
message: .closest() should return a unique set of elements, not duplicates of the same ancestor. Fixes #6700

This commit does not belong to a tracked branch and thus cannot be retrieved by any other repository

Affected paths could not be retrieved for this commit.
This commit is known to exist in the repositories with the following ids:

- rep1403042549624

Fig. 9 Collapsed commit history

on the push list. Yellow commits need attention because they exist in at least one peer in the pull list, but do not exist locally, meaning that they may be pulled. Red commits do not exist locally and are not available to be pulled, as they exist only in clones that are not peers. Finally, gray commits belong to non-tracked branches, so they can neither be pushed nor pulled. Heads of these branches are not identified with thicker borders.

This visualization can easily have thousands of nodes, one for each commit in the topology. Nevertheless, despite the high number of nodes, users are usually interested in the most recent commits. As we show the commits following a chronological order, from left to right, most recent commits will be at the right part of the visualization. DyeVC positions the graph so that these commits are shown when opening the visualization.

3.3 Automatic collapsing

As previously discussed, the first version of DyeVC struggled when dealing with larger repositories (over 6500 commits). This limitation was mainly due to the memory used to represent commit nodes in the commit history graph. However, we observed that many of the commit nodes are unnecessary for comprehending the evolution and, in fact, were cluttering the visualization. For instance, a sequence of 20 commit nodes that are ordinary revisions and that belong to all clones (i.e., all have the same white color) could be collapsed into just one commit node, avoiding visualization cluttering and boosting performance. This observation motivated us to design and implement an automatic collapsing feature for DyeVC.

We identified two common node structures that can be automatically collapsed: sequential and parallel. The former contains a sequence of commits of the same type, where each of them has degree two, i.e., nodes with just one ancestor and one successor. This kind of structure can be collapsed because it does not represent any additional information besides the fact that some sequential work was performed. Figure 10 shows examples of sequences of commits, highlighted in red, which could be collapsed, producing the graph shown in Fig. 11 (still in red). On the other hand, the later contains one fork node and one merge node, with at most one (regular or collapsed) 2-degree node in each branch, between the fork and the merge nodes. Figure 11 shows examples highlighted in yellow of this parallel structure. The result of the collapse is shown in Fig. 12. The numbers inside the red and yellow circles refer to the number of collapsed nodes.

We implemented an iterative algorithm that works in phases to benefit from both sequential and parallel collapse strategies together. The algorithm is shown in Fig. 13.

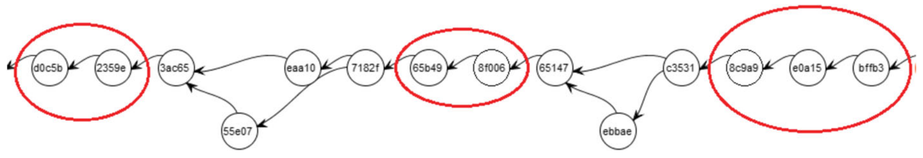


Fig. 10 Sequential structures before automatic collapsing

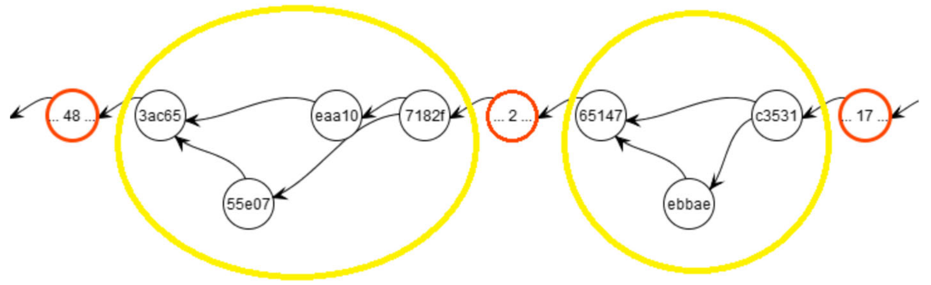


Fig. 11 Sequential structures after automatic collapsing and Parallel structures before automatic collapsing

The algorithm receives the commit graph and the number of iterations as parameters. Each iteration is executed in linear time complexity. The first phase collapses sequential structures (lines 2–11). The set of visited nodes is initialized as an empty set in line 2. Each node is inspected in a loop (lines 3–11). If the node is still not visited, the presence of a linear commit chain is tested in line 5 by examining predecessors and successors of the node. All found linear commit chain elements are marked as visited in line 6. If the linear commit chain has more than one node, it is collapsed (lines 7–9).

The second phase of the algorithm collapses parallel structures (lines 12–28). The *visited* set is reinitialized as an empty set in line 12. All nodes of the graph are examined again in a loop (lines 13–28). If the element is still not visited, and we are in the presence of a fork node (condition in line 14), the parallel structure generated by this fork is analyzed. Both nodes afterward the fork are saved in variables *a* and *b* (lines 15–16). Initially, a *group* set is initialized containing a single element, the fork node, in line 17. If the parallel structure resembles the 4-node group highlighted in yellow in Fig. 11, then the group to be collapsed is populated in lines 18–19. On the other hand, if the parallel structure resembles the 3-node group highlighted in yellow in Fig. 11, then the group to be collapsed is populated in lines 20–21. The visited set is updated in line 23. If the created node group has more than one node, it is collapsed in lines 24–26.

The phases of the algorithm can be repeated, as collapsing parallel structures may lead to new sequential structures. For instance, after applying parallel collapses over the graph shown in Fig. 11, a new sequential structure is formed, as illustrated in Fig. 12. The iteration would lead to a new collapse, and so on. As previously discussed, collapses are performed just for commits of the same type (same color, discussed in section 3.2.4), reducing the size of the graph without compromising the quality of the information shown in the graph.

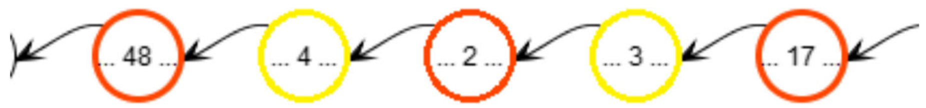


Fig. 12 Parallel structures after automatic collapsing

```
Algorithm 1 Automatic Collapse of Commit Graph.
Input:
    $graph$                                          ▷ The commit graph
    $iterations$                                     ▷ The number of collapse iterations
while $iterations > 0$ do
    $visited \leftarrow \emptyset$
    for each $node \in graph.nodes$ do               ▷ Sequential Collapse
        if $node \notin visited$ then
            $group \leftarrow chain(node, OUT) \cup \{node\} \cup chain(node, IN)$
            $visited \leftarrow visited \cup group$
            if $|group| > 1$ then
                $collapse(group)$
            end if
        end if
    end for
    $visited \leftarrow \emptyset$
    for each $node \in graph.nodes$ do               ▷ Parallel Collapse
        if $node \notin visited \wedge |node.in| = 2$ then
            $a \leftarrow node.in[0]$
            $b \leftarrow node.in[1]$
            $group \leftarrow \{node\}$
            if $linear(a) \wedge linear(b) \wedge a.in[0] = b.in[0]$ then
                $group \leftarrow group \cup \{a, b, a.in[0]\}$
            else if $(linear(a) \wedge a.in[0] = b) \vee (linear(b) \wedge b.in[0] = a)$ then
                $group \leftarrow group \cup \{a, b\}$
            end if
            $visited \leftarrow visited \cup group$
            if $|group| > 1$ then
                $collapse(group)$
            end if
        end if
    end for
    $iterations \leftarrow iterations - 1$
end while
return $graph.$

function CHAIN(node,direction)
    $chain \leftarrow \emptyset$
    while $linear(node)$ do
        $chain \leftarrow chain \cup node$
        if direction=IN then
            $node \leftarrow node.in[0]$
        else
            $node \leftarrow node.out[0]$
        end if
    end while
    return $chain$
end function

function LINEAR(node)
    return $|node.in| = 1 \wedge |node.out| = 1$
end function
```

Fig. 13 Automatic collapsing algorithm

3.4 Behind the scenes

The process underneath DyeVC can be formally defined using Set Theory, in order to describe how the data is structured and how DyeVC can play with this data to identify the repositories that are ahead or behind of other repositories, showing the commits that are missing or that belong to specific branches. We can define a project p as a tuple $(R, C, C^{database})$, where R is the set of all cloned repositories of p monitored by DyeVC, C is the set of all commits of p, and $C^{database}$ C is the set of commits of p in the DyeVC database. Each repository $r_i \in R$ is a tuple $\left(R_i^{push}, R_i^{pull}, C_i^{previous}, C_i^{current}, B_i\right)$, where $R_i^{push} \subseteq R$ is the set of repositories that r_i is allowed to push to, $R_i^{pull} \subseteq R$ is the set of repositories that r_i is allowed to pull from, $C_i^{previous} \subseteq C$ is the set of commits in r_i in the previous execution of DyeVC, $C_i^{current} \subseteq C$ is the set of commits in r_i in the current execution of DyeVC, and B_i is the set of named branches of r_i (Fig. 14a). It is worth noting that, as C is the set of all commits of project p (i.e., the domain of commits of p), any set of commits that belong to specific repositories $r_i \in R$, such as $C_i^{previous}$ and $C_i^{current}$, should also belong to C.

Each commit $c_j \in C$ has a set of parent commits $C_j^{parent} \subset C$. Commits are organized in a directed acyclic graph (Fig. 14b), where the first commit of the project has no parent (e.g., commit A in Fig. 14b), revision commits have only one parent (e.g., commit B in Fig. 14b), and merge commits have two or more parents (e.g., commit I in Fig. 14b). All reachable commits from c_j form its history, including c_j itself and the transitive closure over its parents (e.g., {A, B, E, F, H, I, J} is the history of commit J in Fig. 14b). The history of $c_j \in C$ is formally defined as:

$$H_j = \left\{ c \in C | c = c_j \vee \exists c_k : \left(c_k \in C_j^{parent} \wedge c \in H_k \right) \right\}$$

At this point, it is important to notice that ordering is not important for accounting which commits belong to each repository. The only situation in which ordering is important is when DyeVC plots the commit history graph. In this case, DyeVC accesses the tip of the branches (fast operation, as each branch has a reference to its tip) and traverses its transitive closure for plotting all previous commits (also fast, because each commit has a reference to its parents). Note that the commit graph is a directed acyclic graph (DAG), and this DAG is already represented in terms of pointers in C.

The sets of previous and current commits in a repository r_i are updated periodically, according to the monitoring frequency parameter defined by the DyeVC user. In the first execution of DyeVC over r_i, $C_i^{previous} = \varnothing$ and $C_i^{current}$ is populated with all commits obtained directly from Git. In the following executions, $C_i^{previous}$ is populated with the commits *in* $C_i^{current}$ of the previous execution and $C_i^{current}$ is again populated with all commits obtained directly from Git.

Each branch $b_k \in B_i$ is a tuple $(name, c_k)$, where $name$ is the name of b_k and $c_k \in C$ is the tip (i.e., head) of b_k. Consequently, H_k C contains all reachable commits of b_k.

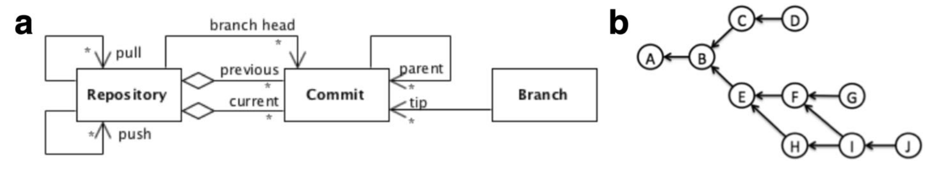


Fig. 14 UML class diagram representing the DyeVC formalization (**a**) and a directed acyclic graph of commits (**b**)

With this foundation established, we can now formalize the process of updating commits in the topology. For a local repository $r_i \in R$ being monitored by DyeVC, the rare situations where a commit is deleted can be formally defined as:

$$Del_i = C_i^{previous} \backslash C_i^{current}$$

Each locally deleted commit $c \in Del_i$ should be removed from $C^{database}$ if no other repository $r \in R$ still contains this commit. Conversely, the new commits in $r_i \in R$ since the previous monitoring cycle can be formally defined as:

$$New_i = C_i^{current} \backslash C_i^{previous}$$

Each locally added commit that is not already in the database ($c \in New_i \setminus C^{database}$) should be inserted in $C^{database}$. This verification is necessary because some of the locally added commits might have already been inserted into the database by another instance of DyeVC.

Moreover, we can formalize the identification of repositories that contain a specific commit and the repositories that are ahead or behind of a given repository. This information is necessary for building some of our visualizations. We formally define the repositories that contain a commit $c_j \in C^{database}$ as:

$$R_j = \left\{ r_i \in R | c_j \in C_{\mathrm{i}}^{current} \right\}$$

We formally define from which repositories r_i is ahead or behind as:

$$Ahead_i = \left\{ r_j \in R_i^{push} | \exists c \in C_i^{current} : c \notin C_j^{current} \right\}$$

$$Behind_i = \left\{ r_j \in R_i^{pull} | \exists c \in C_j^{current} : c \notin C_i^{current} \right\}$$

Finally, we can also formalize the commits that are ahead or behind two specific repositories and the branches in which a commit belongs. This relationship among commits and repositories/branches is also necessary for some of our visualizations. Considering two repositories $r_i, r_j \in R$, we formally define the commits ahead or behind r_i regarding r_j as:

$$Ahead_{i,j} = C_i^{current} \ C_j^{current}$$

$$Behind_{i,j} = C_j^{current} \ C_i^{current}$$

Considering a given repository r_i, we formally define the branches that a commit $c_j \in C$ belongs to as:

$$B_{i,j} = \left\{ b_k \in B_i | c_j \in H_k \right\}$$

The computation of R_j, $Ahead_i$ and $Behind_i$ is not expensive. The set of repositories R is usually small (one or few repositories per developer) and the complexity of the operation for checking if a commit belongs to a specific repository is O(1) (i.e., the complexity of checking if an element belongs to a hash-based set). So, we can say that the complexity of obtaining the relationship of commits and repositories is O(n), where n is the number of repositories in the project.

4 Implementation

We implemented our approach as a Java application launched via Java Web Start Technology. It currently monitors Git repositories, as it is the most used DVCS nowadays (Eclipse Foundation 2014). The source code and the link to download the tool via Java Web Start can be found at https://github.com/gems-uff/dyevc. The tool gathers information from repositories using JGit library,[2] which allows using our approach without having a Git client installed.

Gathered information is stored in a central document database running MongoDB. We hosted our database on a free MongoDB instance provided by MongoLab. We did not use MongoDB proprietary API, which would demand opening specific ports to connect to MongoDB. Instead, we opted to use MongoLab's RESTful (*Representational State Transfer*) API. RESTful APIs (Fielding 2000) have the advantage of being available using standard HTTP and HTTPS protocols. In this way, our approach can be used in environments protected with firewalls without major problems. We implemented a *MongoLab Provider* to use this RESTful API, which translates the application methods into RESTful commands and vice-versa. It also serializes/deserializes the application objects to/from JSON (*JavaScript Object Notation*) representations to be used through the RESTful commands.

A central document database was chosen because this way DyeVC instances can easily send and gather information. MongoDB was the chosen database because it is free, open-source and cross-platform. Besides, it has many features to improve performance and availability, such as document indexing, replication, and load balancing. Furthermore, it provides RESTful APIs, as cited before.

We present the gathered information as a series of graphs by using the JUNG (*Java Universal Network/Graph*) library,[3] from which DyeVC inherits the ability to extend existing layouts and filters. All graphs present similar behavior, allowing the window to be zoomed in or out, whether the user wants to see details of a particular area or an overview of the entire graph. By changing the window mode from *transforming* to *picking*, it is possible to select a group of nodes and collapse them into one node, or simply drag them into new positions to have a better understanding of parts with too many crossing lines.

5 Evaluation

To evaluate our approach, we first conducted a *posthoc* evaluation over the JQuery project,[4] an open-source project, aiming at checking if DyeVC can help answering questions Q1-Q3. Next, we conducted an observational evaluation involving four participants that used DyeVC. This evaluation also used the JQuery project. Finally, we ran DyeVC over some open-source projects of different sizes and from different sources, aiming at evaluating the scalability of our approach.

5.1 Posthoc evaluation

We conducted a *posthoc* evaluation using a real open source project to demonstrate that our approach can help in answering questions Q1-Q3. The selected project, JQuery, began in 2006 and had 6222 commits by the time of the evaluation. We reconstructed the repository history, simulating the actions that occurred in the past.

We do not replicate the repository history here, due to its size, but it is publicly available on GitHub. Automatically generated comments helped us to depict specific flows. For example, the comment "*Merge branch 'master' of https://github.com/scottjehl/jquery into scottjehl-master*" tells us that there was a user named "*scottjehl*" and that the merge operation was done at a branch called "*scottjehl-master*". Although one might perform a merge manually and insert a different text in the comment, this did not compromise our analysis because we had a focus on depicting some of the merge situations, and not all of them.

Due to the operating mode of Git, some details are missing, but these details do not compromise our analysis. The first one is the moment when a clone arises or deceases. This information does not exist anywhere in the repository. We inferred the creation of clones by looking at the commit messages (a commit by developer X led to the creation of a clone named X). Clones created at a given time stayed alive for the rest of the analysis.

The second missing detail is that, although we had the commit dates and times in the repository history, these dates and times were not guaranteed to be correct. This occurs because DVCSs do not have a central clock. Each commit is registered with the local time on the machine where the clone is located, which could lead to commits in the history with a predecessor in the future, depending on when and where each commit was performed. This missing detail is not relevant, because the order of commits is not depicted using their times, but using the pointers that Git maintains from a commit to its parents, as discussed in section 3.1. We can use these dates, but not as an authoritative information.

Finally, if rebases were conducted at the repository, this *posthoc* evaluation had no means to detect it, once a rebase consists on rewriting the local history for placing parallel commits on top of existing commits, consequently leaving no trail of the parallel work. This missing detail is not important for our evaluation as well, because this operation is done solely with the purpose of cleaning the repository, leaving its history easier to understand. However, other *posthoc* studies that intend to use DyeVC for finding all cases of parallel work should consider rebase as a potential threat to validity.

We chose a moment in time when three developers were involved, performing commits and merging changes in the repository. We created three clones for these developers, named after their usernames: *jeresig*, *adam*, and *aakosh*. Figure 15 shows the topology view on Sep 24 2010[5] when *aakosh* had 121 commits pending to be pushed to the central repository (hereafter called *central-repo*). Figure 16 shows part of *aakosh's* commit history and how DyeVC represents commits pending to be pushed (green nodes).

Later on, *aakosh* pushed his commits to *central-repo***.** In the meantime, both *adam* and *jeresig* committed some changes. Before they pushed their work to *central-repo*, *adam's* last commit was on Jun 21, 2010, and *jeresig's* on Sep 27 2010. At this moment, we registered them to be monitored by DyeVC. Figure 17 shows the topology view after this registration on Sep 27 2010.[6] Here, we can see that *aakoch* was synchronized with *central-repo*, whereas *adam* and *jeresig* had pending actions.

At this point, we can revisit questions Q1 and Q2:

Q1: *Which clones were created from a repository?* DyeVC's topology view (Fig. 17) shows all the clones where it is running, and also discovers other clones connected to them, even if it is not running there.

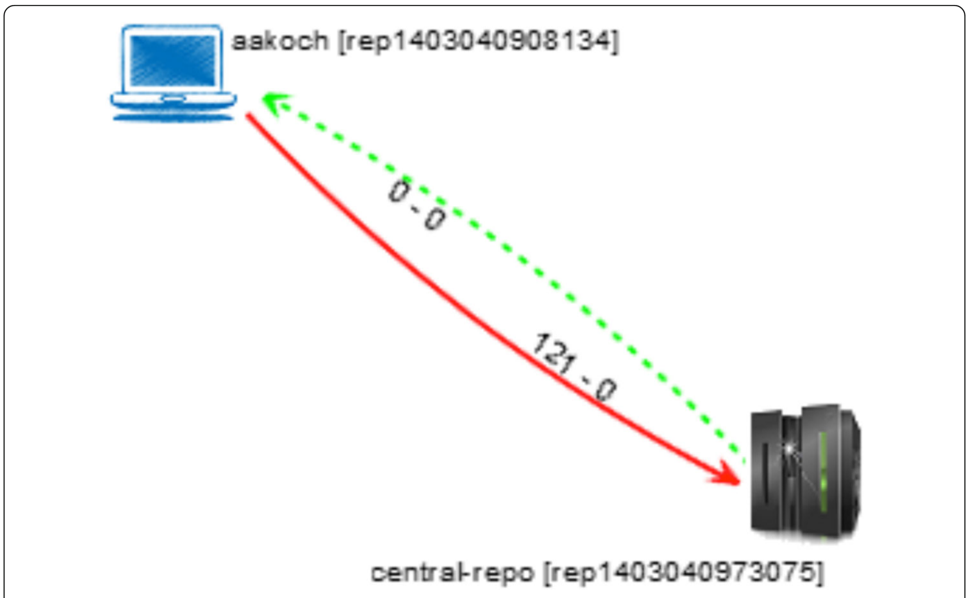


Fig. 15 First monitored repository in Topology view (Sep 24 2010)

Q2: *What are the communication paths among different clones?* DyeVC's topology view (Fig. 17) shows the dependencies between peers in the topology, as well as the number of commits ahead or behind in each of these clones.

Adam had 121 commits to pull from *central-repo*, what is corroborated by the details of his tracked branches (master branch in Fig. 18a). He also had a non-tracked commit pending to be pushed. Non-tracked commits are not shown in the tracked branches view, but we can see them in gray in the commit history views. Fig. 18b shows the collapsed commit history for *jeresig*, where we can see *adam*'s non-tracked commit with hash *a2bd8*.

The repository history leads us to think that *jeresig* is a core developer of this project because he performed most of the merges to the master branch. Looking at Fig. 17, we see that he had 26 commits pending to be pushed to *central-repo*. These 26 commits can be seen at *aakoch's* commit history (Fig. 19) as red commits since they could not be pulled by *aakoch* until *jeresig* has pushed them to *central-repo*. There was also a commit in central-repo pending to be pulled by *jeresig*. If we look back at Fig. 18b, we see that the only yellow commit is *a0887*, made by *aakoch*. This tells us that *jeresig* pulled changes from *central-repo* just before *aakoch* pushed commit *a0887*. If we look at Fig. 20, we see that all pending commits (those that were pending to be pushed and pulled) are related to the same branch (master). This tells us that, if *jeresig* wanted to push these commits to *central-repo*, he would have to perform a pull operation before.

This analysis helps us revisit and answer Q3:

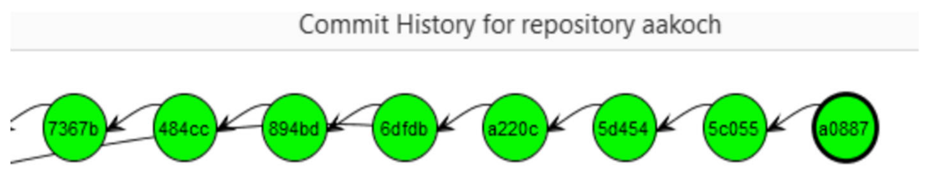


Fig. 16 aakoch's commit history showing commits pending to be pushed

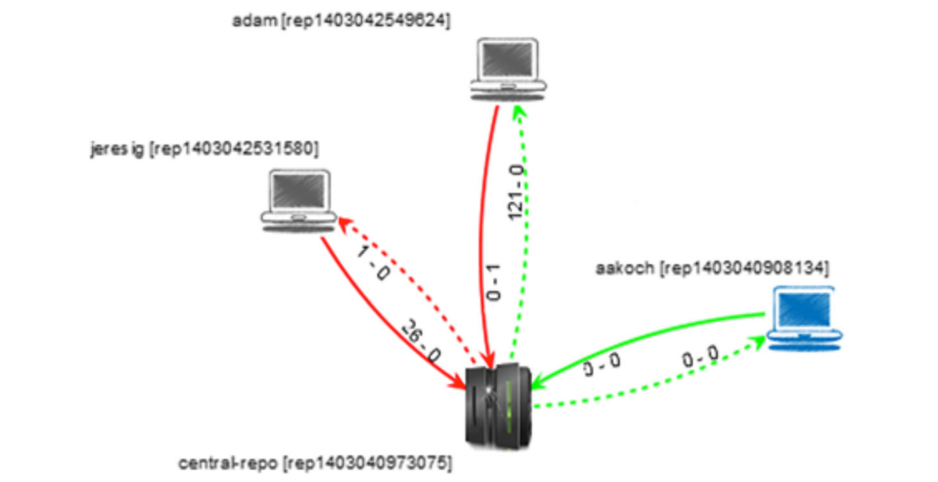
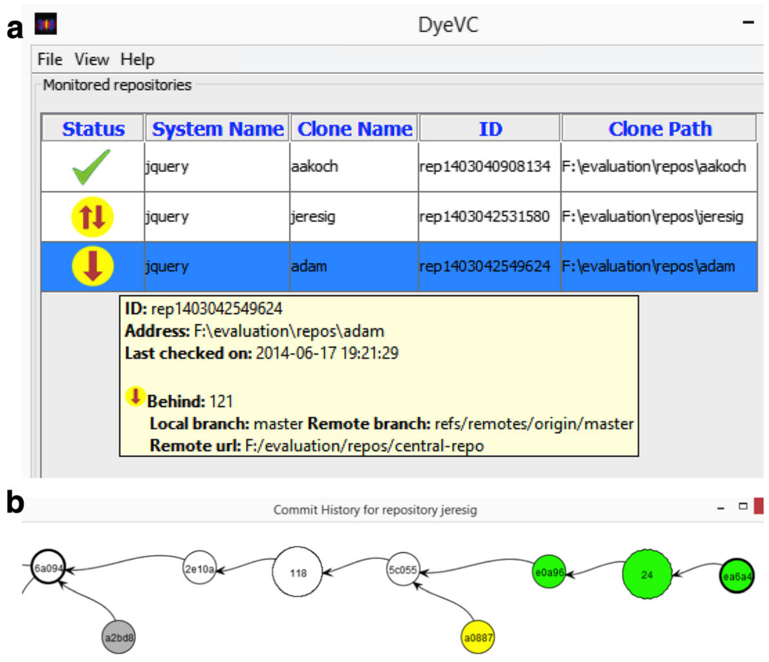

adam [rep1403042549624]
jeresig [rep1403042531580]
aakoch [rep1403040908134]
central-repo [rep1403040973075]
121 - 0
0 - 1
1 - 0
26 - 0
0 - 0
0 - 0

Fig. 17 Three monitored repositories in Topology view (Sep 27 2010)

Q3: Which changes are under work in parallel (in different clones or different branches) and which of them are available to be incorporated into others' clones? New commits in tracked branches of peers can be easily found by looking at Level 3 information (tracked branches, shown in Fig. 18a and Fig. 20). This view shows to which branch these commits are related and how many new commits exist. If we want to look at each commit individually, we can look at Level 4 information (commit history, shown in Fig. 16 and Fig. 19) and notice the yellow nodes. Additionally, Level 4 information can be used to find new commits in repositories that are not peers (red nodes), or new commits in non-tracked branches (gray nodes).

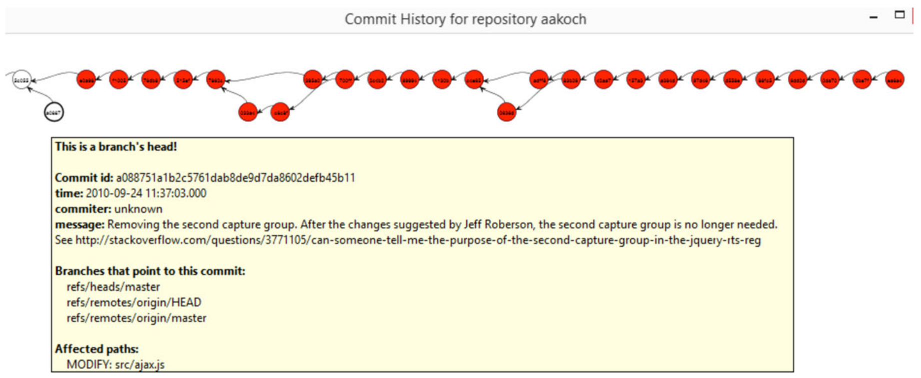


Status	System Name	Clone Name	ID	Clone Path
✓	jquery	aakoch	rep1403040908134	F:\evaluation\repos\aakoch
⇅	jquery	jeresig	rep1403042531580	F:\evaluation\repos\jeresig
↓	jquery	adam	rep1403042549624	F:\evaluation\repos\adam

ID: rep1403042549624
Address: F:\evaluation\repos\adam
Last checked on: 2014-06-17 19:21:29

↓ **Behind:** 121
Local branch: master **Remote branch:** refs/remotes/origin/master
Remote url: F:/evaluation/repos/central-repo

Fig. 18 Adam's tracked branches (**a**) and collapsed commit history for repository jeresig (**b**)



This is a branch's head!

Commit id: a088751a1b2c5761dab8de9d7da8602defb45b11
time: 2010-09-24 11:37:03.000
commiter: unknown
message: Removing the second capture group. After the changes suggested by Jeff Roberson, the second capture group is no longer needed. See http://stackoverflow.com/questions/3771105/can-someone-tell-me-the-purpose-of-the-second-capture-group-in-the-jquery-rts-reg

Branches that point to this commit:
refs/heads/master
refs/remotes/origin/HEAD
refs/remotes/origin/master

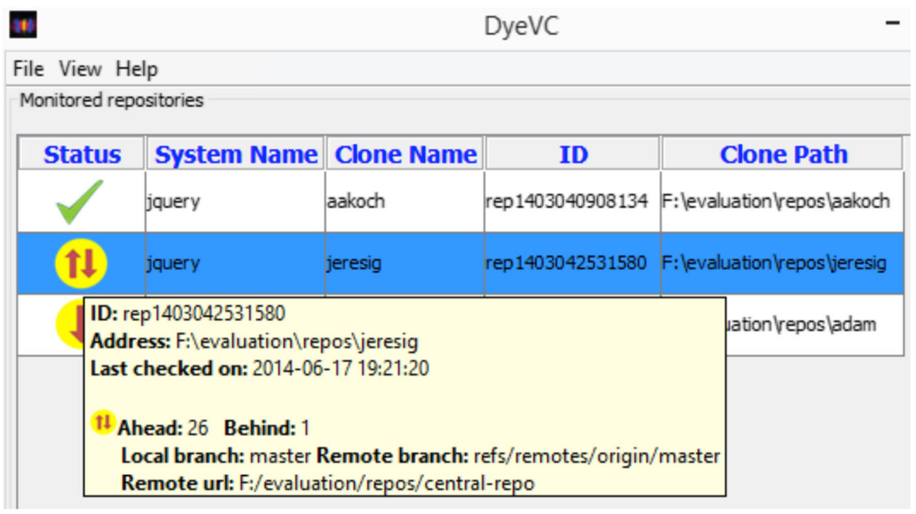
Affected paths:
MODIFY: src/ajax.js

Fig. 19 Aakoch's commit history

5.2 Observational evaluation

We conducted an observational evaluation over the same project used in the *posthoc* evaluation (JQuery) to assess the capability of the visualizations provided by DyeVC in supporting developers and repository administrators. The evaluation was conducted with four volunteers, which had previous experience with DVCS. They were graduate students from the Software Engineering research area at Universidade Federal Fluminense (UFF). Four sessions were conducted, each of them with one subject.

The goal of this observational evaluation was to analyze when DyeVC helps on understanding the project history better than existing tools. The evaluation was divided into two phases (without and with DyeVC), each one with two scenarios, where the subject had to answer questions related to usual work with DVCS. In Scenario 1, the subject played the developer role, working in a clone named *aakoch*. In Scenario 2, the subject played the repository administrator role. The following questions were posed: Q1.1 What is the status of your clone, compared to the central repository? Q1.2 Who else is working in the JQuery project (other clones)? Q1.3 Which files were modified in commit *5d454*? Q2.1 What are the existing clones for JQuery project? Q2.2 Which



Status	System Name	Clone Name	ID	Clone Path
✓	jquery	aakoch	rep1403040908134	F:\evaluation\repos\aakoch
⇅	jquery	jeresig	rep1403042531580	F:\evaluation\repos\jeresig
				ation\repos\adam

ID: rep1403042531580
Address: F:\evaluation\repos\jeresig
Last checked on: 2014-06-17 19:21:20

Ahead: 26 **Behind:** 1
Local branch: master **Remote branch:** refs/remotes/origin/master
Remote url: F:/evaluation/repos/central-repo

Fig. 20 Jeresig's tracked branches

clones are synchronized with the central repository? Q2.3 How many commits in tracked branches are pending to be sent to the central repository? Q2.4 Is there any commit in non-tracked branches? Where?

In Phase 1 (without DyeVC), DyeVC was not in place, and the subject answered the questions using any desired DVCS client among the ones available in the computer used in the evaluation: *gitk***,** *Tortoise Git***,** *Git Bash*, and *SourceTree*. Participants were allowed to access the Internet and search any other procedure or tool that could help in answering the questions. After that, the subject watched a 10-min video presenting DyeVC and started Phase 2 (with DyeVC), which consisted of answering the same questions with the help of DyeVC. The possible answers in Phase 2 were either "keep the answer of Phase 1", meaning that using DyeVC did not change the subject perception, or a different answer, otherwise.

Table 3 presents the time spent by each subject to answer each question in both scenarios and both phases. The values include the time to understand the question, investigate repositories with available tools, look for the answer, and write down the answers in the form. However, the values do not include the time spent filling the consent form and the characterization form, watching the video about DyeVC, and filling the exit questionnaire. It is possible to notice, by looking at Table 3, that all subjects took less time to complete Scenario 1 (developer role) in Phase 2 (with DyeVC). For Scenario 2 (admin role), none of the subjects managed to answer the questions in Phase 1 (without using DyeVC). For this reason, times shown in Phase 1 are the times spent by the subjects until they gave up finding an answer.

In Phase 1, each subject used different ways to look for the answers. In Phase 2, subjects correctly used DyeVC to find the answers. Question 1.1 was answered using DyeVC Level 3 visualization (Tracked branches). Question 1.3 was answered using Level 4 visualization (Commit History). Finally, questions 1.2 and 2.1 through 2.4 were answered using Level 2 visualization (Topology). Almost all subjects answered all the questions similarly, except for subject P4 in question 1.2 from Phase 1.

Subject P1 answered questions 1.1 and 1.3 in Phase 1 using the command line interface. To answer question 1.1, she looked at the log for both local and remote repositories, counted down how many hashes there were in each log and subtracted these numbers to find the answer. Question 1.3 was answered with *git show* command, which shows, for each affected file in the commit, what has changed. The answer to this question was easy to find because only one file was affected, but if many files had been affected, the subject would have trouble finding all affected files using this procedure. For questions 1.2 and 2.1 through 2.4, the subject tried to find a way to discover related clones by searching the Internet. After a few searches with no promising results, the

Table 3 Time spent (in minutes) to answer each question

Subject	Scenario 1		Scenario 2	
	Phase 1	Phase 2	Phase 1	Phase 2
P1	14	5	10	6
P2	13	6	4	5
P3	3	2	2	4
P4	10	2	6	10

subject gave up, and her answer was "I don't know". Once there was no answer to question 2.1, next questions in Scenario 2 could not be answered as well.

Subject P2 answered question 1.1 by issuing the *git status* command. To answer question 1.3, she used *Tortoise Git* and walked through the commit tree until finding the desired commit. For questions 1.2 and 2.1 through 2.4, the subject answered that she did not know a way to find an answer. When answering question 2.1, the subject commented that, as a repository manager, she should know which were the existing clones and their relationships, but she did not have any resources available to accomplish that.

Subject P3 answered question 1.1 by issuing a *git status* command (same as subject P2). To answer question 1.3, she used *Tortoise Git* but found the desired commit using the search feature of the tool, instead of walking through the commit tree. For questions 1.2 and 2.1 through 2.4, the subject answered that it was not possible to find an answer.

Subject P4 answered questions 1.1 and 1.3 using *SourceTree*. This subject answered question 1.2 differently from the others. She wrote down each different author of each commit as if it was a different clone. Although this is a valid interpretation, it may happen that authors commit changes in the same clone, and this would lead to a wrong answer for this question. For questions 2.1 through 2.4, the subject answered that it was not possible to find an answer.

The overall results of this evaluation were positive. In Phase 1 (without DyeVC), subjects were able to correctly answer questions Q1.1 and Q1.3 whether using DyeVC or not. Also, further questions were answered correctly only by using DyeVC.

The subjects also answered an exit questionnaire[7] (Cesario 2015). All subjects found easy to interact with DyeVC, to identify related repositories, and to use the operations available. They consensually elected the topology visualization as the most helpful visualization in DyeVC. Also, by using Product Reaction Cards, three out of the four subjects stated that DyeVC is helpful and easy to use. Product Reaction Cards (Benedek and Miner 2003) have a large set of words, both positive and negative, used to check the emotional response of a product or design.

5.3 Performance evaluation

We measured the time spent to perform the most common DyeVC operations to evaluate the scalability of our approach. We used projects of different sizes and hosted in different Git servers. Table 4 shows the monitored projects (name and hosting service), the repository metrics (the number of commits, disk usage, and the number of files) and the time spent to run some background and foreground operations in DyeVC. All measurements were taken in the same period of the day and from the same machine, a Core Duo CPU at 2.53 GHz, with 4GB RAM running Windows 8.1 Professional 64 bits, connected to the internet at 35 Mbit/s. Each operation was performed once for each repository, except for the repository registration, which was executed twice ("Insert 1st" and "Insert 2nd"), as detailed below.

We measured the main operations of our approach: "Insert 1st", invoked when the user registers the first repository of a given system to be monitored; "Insert 2nd", invoked when the user registers a repository to be monitored in a system that already has

Table 4 Scalability results of DyeVC for repositories with different sizes

Repository	Hosting	Repository metrics			Foreground operations			Background operations times (s)			
		Number commits	Size (MB)	Number files	Commit History		Topology	Insert 1st	Insert 2nd	Check Branches	Update Topology
					Time (s)	Memory Usage[a]	Time (s)				
DyeVC	github.com	187	1.0	539	3.5	15	2.7	12.4	16.1	1.7	4.4
SAPOS	github.com	702	7.0	685	5.6	19	3.2	20.8	22.6	1.8	5.2
JGgit	eclipse.org	2979	10.0	1595	18.4	512	3.4	42.4	46.0	5.9	6.8
EGit	eclipse.org	3775	27.0	1478	21.3	559	3.7	49.6	46.6	4.2	7.3
jQuery	github.com	5518	20.0	253	65.0	1121	4.1	40.0	37.4	1.4	9.4
Tortoise Git	code.google.com	6166	85.0	3220	68.0	492	4.2	39.0	36.0	1.6	9.6
Git Extensions	github.com	6417	448.0	1549	73.0	1529	17.0	155.8	129.0	1.6	10.6
Drupal	drupal.org	23,922	84.4	9290	-	-	18.0	102.0	95.0	2.0	18.0
ExpressoLivre	gitorious.org	25,822	141.0	20,729	-	-	18.2	110.0	102.0	2.1	19.3
Git	github.com	35,260	98.0	2656	-	-	19.4	196.0	158.6	3.4	40.0

[a]Memory usage was measured in MB during the execution of "Commit History" operation

registered repositories; "Commit History", invoked when the user requests to see the commit history of a given repository; "Topology", invoked when the user wants to see the topology of repositories of a given system; "Check Branches", invoked periodically to check all monitored repositories, searching for ahead or behind commits; and "Update Topology", invoked periodically to update topology information in the central database. This last operation updates the existing repositories, their peers, and the existing commits, marking in which repositories each commit is found.

It may be noted that the "Commit History" operation has no values for the last three repositories. This occurs because, as the number of commits increases, more memory is used to calculate the commit history graph. The current algorithm has an $O(x^2)$ space complexity (being x the number of commits). The computer used in this evaluation was configured with a 2 GB maximum Java Heap Size, which let us analyze repositories with up to 6 K commits. This limitation occurs mainly because of JUNG.

Table 5 shows the correlation between each repository size metric and the DyeVC operations' execution time, according to the Spearman's rank correlation coefficient (Spearman 1904). This correlation coefficient measures the monotonic relation between two variables and ranges from *−1* to *1*. Values of *1* or *−1* mean that each variable is a perfect (increasing or decreasing) monotone function of the other. A value of *0* means that there is no correlation between the variables.

Looking at Table 5, it is possible to notice that, except for the "Check Branches" operation, all other operation times are strongly correlated to the number of commits and repository size. This is due to the nature of these operations, which update or show information about all commits in the repository. On the other hand, except for the "Commit History" operation, all other operation times correlate with the number of files. This is also expected due to the nature of "Commit History" operation, which does not dig into the changed files.

However, it is possible to find some more tricky situations, which demonstrate that all three variables (number of commits, size, and number of files) should be taken into consideration when analyzing the performance of each DyeVC operation. One such situation is the one that occurs with *Git Extensions*, which has significantly fewer commits than repositories such as *Git*, but presents times for "Topology", "Insert 1st" and "Insert 2nd" operations in the same level of magnitude. This is because these operations are very I/O intensive. When a repository is registered to be monitored, DyeVC creates the working copy for that repository, as discussed in Section 3.1. Larger repositories will then take more time to perform these actions. Note in Table 4 that the size

Table 5 Spearman's rank correlation coefficient between repository size metrics and DyeVC operations time

Operation	Number commits	Size	Number files
Insert 1st	0.85	0.83	0.76
Insert 2nd	0.85	0.83	0.76
Check Branches	0.07	−0.05	0.72
Update Topology	1.00	0.88	0.52
CommitHistory	1.00	0.96	−0.04
Topology	1.00	0.88	0.52

of *Git Extensions* is notably bigger than any other of the repositories used in the evaluation.

Finally, it is worth mentioning that, even with all measurements taken in the same period of the day and from the same machine, short network latencies and processor usage peaks may have occurred, which affect the results.

All in all, although we cannot affirm that DyeVC is scalable to all possible repositories, our evaluation helped us to identify the scalability limits of DyeVC. Without automatic collapses, DyeVC was able to process repositories with around 6500 commits. To put this number in perspective, 99% of 50,012 projects analyzed by Rainer and Gale (2005) had less than 3137 commits. Moreover, Kalliamvakou et al. (2014) indicate that 90% of the projects in GitHub have less than 50 commits. This shows that DyeVC is scalable for a large number of projects, although we still see space for improvements, as presented in the following section.

5.4 Automatic collapsing evaluation

We also studied the impact of the automatic collapsing algorithm in the "Commit History" operation performance. This evaluation was performed at a later time in comparison with the results obtained in the previous section. Consequently, the repository metrics are slightly different. The repository size, number of commits, and number of files are higher, as shown in Table 6.

The design of the evaluation was as follows. First, the "Commit History" operation was performed without using the automatic collapsing. Afterward, sequential and parallel collapse strategies described in Section 3.2.5 were used to simplify the structure of the commit graph, collapsing the corresponding node structures. The execution of the sequential strategy was the first stage, and the parallel strategy was the second stage of each iteration of the automatic collapsing algorithm. Moreover, after each stage, running time and memory consumption were measured. The evaluation was executed in a Core i7 CPU at 2.00 GHz, with 16GB of RAM running Windows 7 64 bits.

We evaluated the capability of the automatic collapsing algorithm to reduce the number of nodes in the commit graph without compromising the quality of the information

Table 6 Characterization of the repositories used in the evaluation of the automatic collapsing algorithm

Repository	Characteristics		
	Size (MB)	Number files	Number commits
DyeVC	3.2	745	228
SAPOS	18.8	668	1245
JGit	39.3	1902	4741
EGit	63.6	1779	4983
jQuery	29.2	296	7291
Git Extensions	94.9	1710	8146
Tortoise Git	168	3518	8442
Drupal	176	10,285	38,047
ExpressoLivre	366	21,592	27,079
Git	104	3026	46,794

shown in the graph (collapses are performed just for commits of the same type, i.e., same color, as discussed in section 3.2.4).

Table 7 shows the reduction achieved after two iterations of the algorithm. The number of iterations was set to two due to empirical observation that there was almost no reduction after a second iteration. With two iterations, the algorithm can reduce the number of nodes by an average of 73% compared to the original graph. In some cases, such as Drupal or ExpressoLivre, which are repositories that we could not analyze before, the nodes reduction surpassed 90%, allowing us to visualize their commit history graph after the automatic collapsing process.

Furthermore, we analyzed the running time and memory consumption of the "Commit History" operation. In particular, data collected for repositories that were visualized before and after collapse are represented in Fig. 21 and Fig. 22 using boxplots. The figures show that the more collapse stages we execute, the less time is needed to represent the commit history, and the less memory is consumed for this purpose. This can be explained by the fact that the automatic collapsing algorithm is linear and very fast comparing to the subsequent visualization process, and speeds up the presentation of the commit graph. Using this method, significantly lower running times and memory consumption values are obtained, compared with values before the automatic collapsing. It was possible to visualize repositories with tens of thousands of nodes (Drupal and ExpressoLivre), which could not be represented before, without applying automatic collapsing process.

Git was the only repository that was not represented visually, even after the automatic collapsing. The main contributing reasons for this fact are its high number of nodes, its low nodes reduction rates, and its inherent complexity.

In the case of Drupal and ExpressoLivre repositories, high nodes reduction rates seem to be influenced by a somewhat more linear structure of the commit graph. There are long chains of 2-degree nodes, corresponding to sequential work stages performed by one contributor. Instead, Git repository showed resistant to collapse. To explain what we mean by "intrinsic complexity" of Git, we identified some structures that prevent commit graph's reduction. An example is shown in Fig. 23. Given the current definition of the collapse operations, the whole structure cannot be reduced because the

Table 7 Reduction of the number of nodes by the automatic collapsing algorithm

Repository	Before Collapse	Iteration 1			Iteration 2		
		1st stage	2nd stage	Reduction (%)	1st stage	2nd stage	Reduction (%)
DyeVC	228	73	47	67.98	32	32	85.96
SAPOS	1245	456	404	63.37	378	375	69.88
JGit	4741	3015	2751	36.41	2635	2635	44.42
EGit	4983	3007	2564	39.65	2347	2329	53.26
jQuery	7291	867	709	88.11	609	603	91.73
Git Extensions	8146	4083	3833	49.88	3702	3684	54.78
Tortoise Git	8442	1466	945	82.63	497	482	94.29
Drupal	38,047	903	697	97.63	563	557	98.54
ExpressoLivre	27,079	3008	2792	88.89	2669	2669	90.14
Git	46,794	24,459	24,216	47.73	24,094	24,094	48.51
Average				66.23			73.15

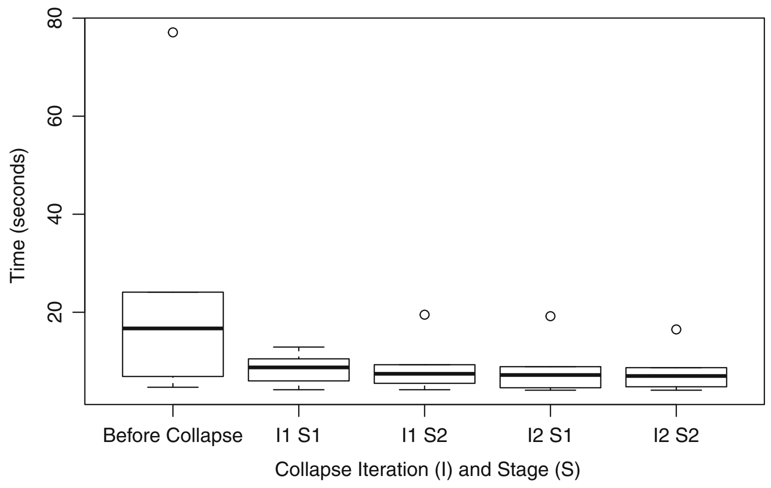


Fig. 21 Boxplot showing the running time of "Commit History" operation depending on the number of executed collapse stages

branches are not sequential chains of commits. Additional automatic collapsing heuristics that consider the possible dependencies between different branches seem necessary to accommodate these cases.

5.5 Threats to validity

While we have taken care to minimize threats to the validity of the evaluations, some factors can influence the results. The usage of a *posthoc* evaluation to assess a real project may not reflect the exact sequence of events that occurred, although the outcome did not change. For example, when we say that *aakosh*, at some moment, had 121 commits pending to be pushed to the central repository, these commits could have been pushed at once or by a series of smaller pushes. Moreover, only one project was selected to perform the analysis, what imposes limitations from a generalization

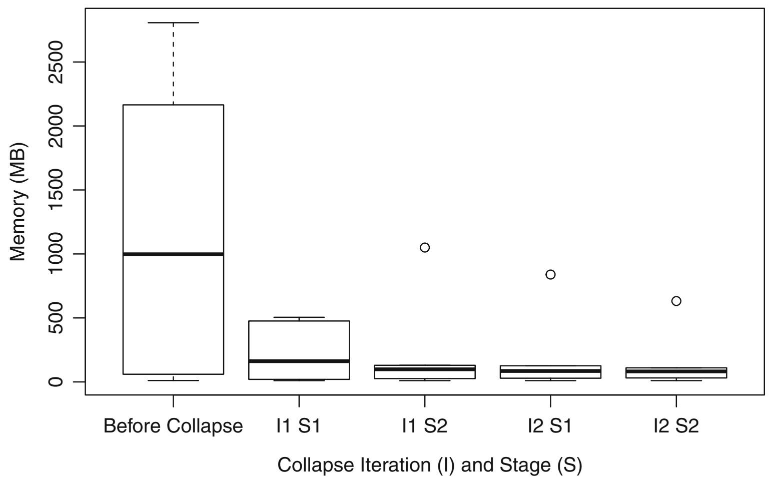


Fig. 22 Boxplot showing the memory consumption of "Commit History" operation depending on the number of executed collapse stages

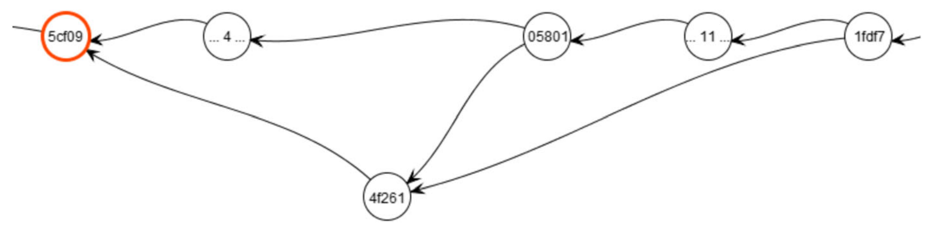


Fig. 23 Example of a structure that prevents automatic collapsing

standpoint. Furthermore, we used an open source project to perform the *posthoc* evaluation, but the *modus operandi* of peers may be different in academic or industrial contexts.

In the observation evaluation, the selection of subjects was made by asking for volunteers from students in the same research group of the author. This was necessary due to time and people restrictions. Therefore, this group might not be representative and can be biased. Moreover, there were few subjects in this evaluation. Thus, the results may have been influenced by the size and by specific characteristics of the group. Furthermore, subjects performed tasks involving DyeVC right after knowing the approach, giving no time to subjects to assimilate the tool. Results may have been influenced by this lack of time to mature the necessary knowledge to use the approach efficiently. Also, subjects could have answered questions in Phase 2 faster than in Phase 1 due to their learning regarding the scenario.

Finally, there is a risk regarding the instrumentation used to measure the response times during the performance evaluation. As we used a database stored over the Internet, connectivity issues and network instability may have affected the response times.

6 Related work

According to Diehl (2007), software visualization can be separated into three aspects: structure, behavior, and evolution. DyeVC relates primarily to the evolution aspect, more specifically with studies that aim at improving the awareness of developers that work with distributed software development. Steinmacher et al. (2012) present a systematic review of awareness studies, which we used to perform a forward and backward snowballing. The approaches obtained after the snowballing were divided into four groups. The first group ("*Commit notification*") includes approaches that notify commit activities. The second group ("*Awareness of concurrent changes*") comprises approaches that not only give the developer awareness of concurrent changes but also inform them about conflicts. The third group ("*Repository visualization*") includes approaches that visualize repository information. Finally, the fourth group ("*DVCS clients*") contains commercial and open source DVCS clients.

The first group contains tools such as SVNNotifier,[8] SCMNotifier,[9] Commit Monitor,[10] SVN Radar,[11] Hg Commit Monitor[12] and Elvin (Fitzpatrick et al. 2006). The primary focus of these approaches is on increasing the developer's perception of concurrent work by showing notifications whenever other developers perform actions. The approaches in this group do not identify related repositories and do not provide information on different levels of details, such as status, branches, and commits. DyeVC provides these different levels of details, as shown in Section 3.2.

The second group comprises approaches that give the developer awareness of concurrent changes, sometimes informing them if conflicts are likely to occur. This group includes tools such as Palantir (Sarma and van der Hoek 2002), CollabVS (Dewan and Hegde 2007), Crystal (Brun et al. 2011), Lighthouse (da Silva et al. 2006), FASTDash (Biehl et al. 2007), and WeCode (Guimarães and Silva 2012). Among these, only Crystal and FASTDash work with DVCSs. Crystal detects physical, syntactic, and semantic conflicts in Mercurial and Git repositories (provided that the user informs the compiling and testing commands), but does not precisely deal with repositories that pull updates from more than one peer. FASTDash does not detect conflicts directly, as the previously cited studies, but provides awareness of potential conflicts, such as two programmers editing the same region of the same source file in repositories stored in Microsoft Team Foundation Server. Although DyeVC primary focus is not to detect conflicts, it can be combined with such approaches to allow conflicts and metrics analysis over DVCS.

The third group includes approaches that visualize repository information. Each approach has a different visualization focus, such as program structures (Collberg et al. 2003), classes (Lanza 2001), lines (Voinea et al. 2005), authors (Gilbert and Karahalios 2006), and branch history (Elsen 2013)[13,.14] The latter have the same focus of DyeVC's Commit History visualization, but dealing only with the local repository, not showing, for example, where a given commit can be found in related repositories.

Finally, the fourth group includes commercial/open source DVCS clients, which allows one to execute operations on repositories/clones (push, pull, checkout, commit, etc.) and also visualizing the repository history, i.e., the commits, along with their attributes (comment, date, affected files, committer, etc.). For example, some Git clients include *gitk,*[15]*Tortoise Git,*[16] *EGit for Eclipse,*[17] and *SourceTree.*[18] The data about commits shown by these tools varies, but usually involves the committer name, message, date, affected files, and a visual representation of the history. These tools, though, have no knowledge regarding peers. For this reason, they do not present commits from other clones and do not include information about where each commit can be found. It is worth noticing that we could not find any similar work showing the dependencies among several clones of a DVCS.

Table 8 compares DyeVC with each group used to classify related work presented in this section, according to the following features: notifications (Which types of notification the approach supports?); CVCS (Does the approach support CVCS?); DVCS (Does the approach support DVCS?); related repositories (Does the approach identifies related repositories?); levels of detail (Does the approach present information in different levels of detail?); multiple peers (Does the approach support repositories with multiple peers, i.e., multiple pull / push destinations?); commits in peer nodes (Does the approach detects commits in peer nodes, i.e., nodes that have a direct communication path to each other?); commits in non-peer nodes (Does the approach detect commits in non-peer nodes, i.e., nodes that do not have a direct communication path to each other?); multiple branches (Does the approach support multiple branches in DVCS?); topology (Does the approach supply any topology visualization that shows dependencies among repositories?); and, finally, commit History (Does the approach allow visualizing only a partial commit history, showing only local commits, or does it allow visualizing a full commit history, including commits in other repositories that were not synchronized yet, or that are in non-tracked branches?).

Table 8 Comparing DyeVC features with related work

Feature	Commit notification	Awareness of concurrent changes	Repository visualization	DVCS clients	DyeVC
Notifications	New commits	Conflicts	No	No	Status change against peers
CVCS	Yes	Yes	Yes	No	No
DVCS	Some[a]	Some[b]	Some[c]	Yes	Yes
Related repositories	No	No	No	No	Yes
Levels of detail	No	No	No	No	Yes
Multiple peers	No	No	No	No	Yes
Commits in peer nodes	No	Some[d]	Some[e]	No	Yes
Commits in non-peer nodes	No	No	No	No	Yes
Multiple branches	No	No	No	Yes	Yes
Topology	No	No	No	No	Yes
Commit history	No	No	Some[f] / Partial[g]	Partial[g]	Full

[a]Exceptions are SCM Notifier and Hg Commit Monitor
[b]Exception is Crystal
[c]Exceptions are VisGi, Visugit, and GitHub's Network Graph
[d]Exception is Lighthouse
[e]Exception is GitHub's Network Graph
[f]Visugit and GitHub's Network Graph
[g]Approaches allow visualizing only local commits. Commits in other repositories that were not synchronized yet, or that are in non-tracked branches, are not shown

All in all, among related work, *Crystal* is the most similar to DyeVC and deserves a deeper comparison. Both approaches work with DVCSs (besides Git, *Crystal* also supports Mercurial) and use working copies to perform analyses, but there are major differences between them. *Crystal's* goal is to identify conflicts among pairs of repositories, whereas *DyeVC's* goal is to provide awareness regarding the existing peers and their synchronization, at different levels. To identify repositories, *Crystal* demands the user to point out all repositories they want to compare, whereas *DyeVC* requires that some of the repositories be registered and it automatically looks at configuration files to discover all the repositories that one pushes to or pulls from. The repository comparison in *Crystal* is from one repository against all the other together, whereas *DyeVC* analyzes each repository against each other, providing a pairwise view and a combined view of the history. Finally, the allowed actions in *Crystal* include the ability to *push*, *pull*, *compile*, and *test* a repository, whereas *DyeVC* allows one to visualize branches status, topology, and history. In this way, we see potential to have both tools working together to provide awareness and safety better when working with DVCS.

6.1 Trace reduction methods and automatic collapsing

Trace reduction is the compression of traces in some manner (either lossless or lossy) so that they can be stored and processed efficiently (Kaplan et al. 1999; Mohror and Karavanic 2009). The process of collapsing the commit graph can be seen as a particular case of trace reduction.

Program analysis and software visualization communities have already proposed trace reduction methods (Kuhn and Greevy 2006; Cornelissen et al. 2008; Noda et al. 2012; Jayaraman et al. 2017). In (Kuhn and Greevy 2006) and (Cornelissen et al. 2008), a trace reduction technique is mentioned, which assigns consecutive events that have equal or increasing nesting levels to the same group. Particularly, method call sequences are summarized in one method-call chain (Kuhn and Greevy 2006).

Also, compact sequence diagram generation is studied in (Noda et al. 2012) and (Jayaraman et al. 2017). To have a better sequence diagram representation of the program execution, Noda's method (Noda et al. 2012) abstracts the history of object interaction by grouping strongly correlated objects. These objects are compacted, achieving an appropriate reduction in the number of objects appearing in the sequence diagram, which results in compression of this diagram along the horizontal axis. Also, (Jayaraman et al. 2017) presents vertical and horizontal sequence diagram compaction techniques. For this end, one-to-one correspondence between call trees and sequence diagrams is used. A maximally compacted tree is obtained, generating smaller and more useful diagrams.

These approaches work with method-call sequences and sequence diagrams, both having a tree structure. Unlike these works, when applying automatic collapsing, we deal with a commit graph, which is a DAG, with a high number of commit nodes. The structure of a graph is more rich and complex than the structure of a tree.

7 Conclusion

In this paper, we presented DyeVC, an approach that identifies the status of a repository in contrast with its peers, which are dynamically found in an unobtrusive way. We have evaluated DyeVC on a real project, showing that it can be used to answer questions that arise when working with DVCSs. The observational evaluation results were promising: DyeVC was considered easy to use and fast for most repository history exploration operations while providing the expected answers. This provides initial evidence that DyeVC could effectively help developers and repository administrators by saving time and by supporting answering questions regarding DVCS usage that could not be answered before. We have also evaluated DyeVC's performance over repositories of different sizes, and we found out that the time and space complexity of the approach are directly related to the number of commits in the repository, especially in the view levels with finer granularity.

Some future research topics arise from this work. DyeVC could gather additional metadata, for example, to create a visualization showing conflicts that would happen when merging two or more branches. This data could also be used to mine information in the repositories, revealing usage patterns or presenting metrics. Moreover, the formalization of DyeVC mechanics could be used to prove correctness properties of our implementation. Finally, some optimization should be done to allow DyeVC work with larger repositories with more complex branch structures.

8 Endnotes

[1]Dye is commonly used in cells to observe the cell division process. As an analogy, DyeVC allows developers to observe how a Version Control repository evolved over time.

[2]http://www.eclipse.org/jgit/

[3]http://jung.sourceforge.net/

[4]https://github.com/jquery/jquery

[5]Considering the scenario just after commit *a088751a1b2c5761dab8de9d7da8602def b45b11*.

[6]Considering the scenario just after commit *ea6a4813b7d996f6f7af0b61a5f1bf4ab80b 291d*.

[7]The exit questionnaire can be found in Appendix G of the referenced Master's Thesis, which can be found at https://github.com/gems-uff/dyevc/blob/master/docs/dissertation.pdf.

[8]http://svnnotifier.tigris.org/ (2012)

[9]https://github.com/pocorall/scm-notifier (2012)

[10]http://tools.tortoisesvn.net/CommitMonitor.html (2013)

[11]http://code.google.com/p/svnradar/ (2011)

[12]https://bitbucket.org/dun3/hgcommitmonitor (2009)

[13]Visugit: https://github.com/hozumi/visugit

[14]GitHub's Network Graph: https://github.com/blog/39-say-hello-to-the-network-graph-visualizer

[15]http://git-scm.com/docs/gitk

[16]https://tortoisegit.org/

[17]http://eclipse.org/egit/

[18]http://www.sourcetreeapp.com/

Abbreviations
API: Application programming interface; CM: Configuration management; CPU: Central processing unit; CVCS: Centralized version control systems; DAG: Directed acyclic graph; DVCS: Distributed version control systems; HTTP: Hypertext transfer protocol; HTTPS: HTTP secure; JSON: JavaScript object notation; JUNG: Java Universal network/graph; MB: Megabyte; RAM: Random access memory; RESTful: Representational State transfer; UML: Unified modeling language; VCS: Version control systems

Acknowledgements
We thank CNPq and FAPERJ for the financial support.

Funding
CNPq and FAPERJ sponsored this work.

Availability of data and materials
The source code and the link to download DyeVC via Java Web Start can be found at https://github.com/gems-uff/dyevc. All projects used in the evaluations are available in their respective repositories, described in Table 4.

Authors' contributions
CC contributed in the design and implementation of DyeVC and the design of the automatic collapsing algorithm and was responsible for running the posthoc, observational, and performance evaluations. RI contributed in the design and implementation of the automatic collapsing algorithm and was responsible for running the automatic collapsing evaluation. LM contributed in the design of DyeVC, the automatic collapsing algorithm, and the evaluations, and was responsible for the DyeVC formalization. All three authors contributed to writing the paper. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

9 Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 9 January 2017 Accepted: 14 July 2017
Published online: 21 July 2017

References

Appleton B, Berczuk S, Cabrera R, Orenstein R (1998) Streamed lines: branching patterns for parallel software development, Pattern languages of programs conference (PLoP), p 98

Benedek J, Miner T (2003) Measuring desirability: new methods for evaluating desirability in a usability lab setting. In: Proceedings of usability professionals association (UPA), Orlando, p 57

Biehl JT, Czerwinski M, Smith G, Robertson GG (2007) FASTDash: a visual dashboard for fostering awareness in software teams. In: ACM conference on human factors in computing systems (CHI). ACM, San Jose, pp 1313–1322
Brun Y, Holmes R, Ernst MD, Notkin D (2011) Proactive detection of collaboration conflicts. In: ACM SIGSOFT symposium and European conference on foundations of software engineering (ESEC/FSE). ACM, Szeged, pp 168–178
Cederqvist P (2005) Version management with CVS. Free Software Foundation
Cesario C (2015) Awareness over distributed version control systems. Master's thesis, Universidade Federal Fluminense - UFF
Cesario CM, Murta LGP (2016) Topology awareness for distributed version control systems. In: Proceedings of the 30th Brazilian symposium on software engineering (SBES). ACM, Maringá, pp 143–152
Chacon S (2009) Pro Git, 1st edn. Apress, Berkeley
Collberg C, Kobourov S, Nagra J, Pitts J, Wampler K (2003) A system for graph-based visualization of the evolution of software. In: ACM symposium on software visualization (SOFTVIS). ACM, San Diego, pp 77–ff
Collins-Sussman B, Fitzpatrick BW, Pilato CM (2011) Version Control with Subversion. Compiled from r4849. O'Reilly Media, Stanford
Cornelissen B, Moonen L, Zaidman A (2008) An assessment methodology for trace reduction techniques. IEEE International Conference on Software Maintenance, In, pp 107–116
Dewan P, Hegde R (2007) Semi-synchronous conflict detection and resolution in asynchronous software development. In: European conference on computer-supported cooperative work (ECSCW). Springer London, Limerick, pp 159–178
Diehl S (2007) Software visualization: visualizing the structure, behaviour, and evolution of software. Springer, Berlin, New York
Eclipse Foundation (2014) 2014 annual eclipse community report. Eclipse Foundation, San Francisco
Elsen S (2013) VisGi: Visualizing Git branches. In: IEEE working conference on software visualization (VISSOFT). IEEE, Eindhoven, pp 1–4
Estublier J (2000) Software configuration management: a roadmap. In: International conference on software engineering (ICSE). ACM, Limerick, pp 279–289
Fielding RT (2000) Architectural styles and the Design of Network-Based Software Architectures. Thesis, University of California
Fitzpatrick G, Marshall P, Phillips A (2006) CVS integration with notification and chat: lightweight software team collaboration. In: ACM conference on computer-supported cooperative work (CSCW). ACM, Banff, pp 49–58
Gilbert E, Karahalios K (2006) LifeSource: two CVS visualizations. In: ACM conference on human factors in computing systems (CHI). ACM, Montreal, pp 791–796
Guimarães ML, Silva AR (2012) Improving early detection of software merge conflicts. In: Internation conference on software engineering (ICSE). IEEE Press, Zürich, pp 342–352
Gumm D-C (2006) Distribution dimensions in software development projects: a taxonomy. IEEE Softw 23:45–51
Jayaraman S, Jayaraman B, Lessa D (2017) Compact visualization of java program execution. Softw Pract Exp 47:163–191. doi:10.1002/spe.2411
Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D (2014) The promises and perils of mining GitHub. In: Proceedings of the 11th working conference on mining software repositories. ACM, New York, pp 92–101
Kaplan SF, Smaragdakis Y, Wilson PR (1999) Trace reduction for virtual memory simulations. In: Proceedings of the 1999 ACM SIGMETRICS international conference on measurement and Modeling of computer systems. ACM, New York, pp 47–58
Kuhn A, Greevy O (2006) Exploiting the analogy between traces and signal processing. In: 22nd IEEE international conference on software maintenance, pp 320–329
Lanza M (2001) The evolution matrix: recovering software evolution using software visualization techniques. In: International workshop on principles of software evolution (IWPSE). ACM, Tokyo, pp 37–42
Mohror K, Karavanic KL (2009) Evaluating similarity-based trace reduction techniques for scalable performance analysis. In: Proceedings of the conference on high performance computing networking, storage and analysis. ACM, New York, pp 55:1–55:12
Noda K, Kobayashi T, Agusa K (2012) Execution trace abstraction based on meta patterns usage. In: 19th working conference on reverse engineering, pp 167–176
O'Sullivan B (2009a) Mercurial: the definitive guide, 1st edn. O'Reilly Media, Sebastopol
O'Sullivan B (2009b) Making sense of revision-control systems. CACM 52:56–62
Perry DE, Siy HP, Votta LG (1998) Parallel changes in large scale software development: an observational case study. In: International conference on software engineering (ICSE). IEEE Computer Society, Kyoto, pp 251–260
Rainer A, Gale S (2005) Evaluating the quality and quantity of data on open source software projects. Proceedings of the 1st international conference on open source software
Rochkind MJ (1975) The source code control system. IEEE Trans Softw Eng 1:364–470
Sarma A, van der Hoek A (2002) Palantir: coordinating distributed workspaces. In: 26th computer software and applications conference (COMPSAC). IEEE, Oxford, pp 1093–1097
da Silva IA, Chen PH, Van der Westhuizen C, Ripley RM, van der Hoek A (2006) Lighthouse: coordination through emerging design. In: Workshop on eclipse technology eXchange (ETX). ACM, Portland, pp 11–15
Spearman C (1904) The proof and measurement of association between two things. Am J Psychol 15:72–101. doi:10.2307/1412159
Steinmacher I, Chaves A, Gerosa M (2012) Awareness support in distributed software development: a systematic review and mapping of the literature. In: ACM conference on computer-supported cooperative work (CSCW). ACM, Seattle, pp 1–46
Tichy W (1985) RCS: a system for version control. Soft Pract Exp 15:637–654
Voinea L, Telea A, van Wijk JJ (2005) CVSscan: visualization of code evolution. In: ACM symposium on software visualization (SOFTVIS). ACM, Saint Louis, pp 47–56
Walrad C, Strom D (2002) The importance of branching models in SCM. IEEE Comput 35:31–38
Wloka J, Ryder B, Tip F, Ren X (2009) Safe-commit analysis to facilitate team software development. In: International conference on software engineering (ICSE). IEEE Computer Society, Vancouver, pp 507–517

APPENDIX H – Curve and surface fitting by implicit polynomials: Optimum degree finding and heuristic refinement

Computers & Graphics 67 (2017) 14–23

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Curve and surface fitting by implicit polynomials: Optimum degree finding and heuristic refinement

Ruben Interian[a,*], Juan M. Otero[b], Celso C. Ribeiro[a], Anselmo A. Montenegro[a]

[a] *Institute of Computing, Fluminense Federal University, Av. Gen. Milton Tavares de Souza, Niterói, Rio de Janeiro 24210-346, Brazil*
[b] *Mathematics and Computer Science Faculty, Havana University, San Lazaro and L, Vedado, Havana 10400, Cuba*

ARTICLE INFO

Article history:
Received 23 December 2016
Revised 5 May 2017
Accepted 5 May 2017
Available online 17 May 2017

Keywords:
Implicit polynomial
Curve fitting
Surface fitting
Object representations
Optimization

ABSTRACT

Finding an implicit polynomial that fits a set of observations X is the goal of many researches in recent years. However, most existing algorithms assume the knowledge of the degree of the implicit polynomial that best represents the points. This paper presents two main contributions. First, a new distance measure between X and the implicit polynomial is defined. Second, this distance is used to define an algorithm able to find the degree of the polynomial needed for the representation of the data set. The proposed algorithm is based on the idea of gradually increase the degree, while there is an improvement in the smoothness of the solutions. The experiments confirm the validity of the approach for the selected 2D and 3D datasets.

1. Introduction

The problem of 2D and 3D object representation is present in different areas of research as computer graphics and computer vision. Modelling, 3D reconstruction and recognition tasks depend totally on a good representation of the observed objects.

Unfortunately, it is usual to receive noised, discrete or incomplete real world data. The models are obtained from images, videos, 3D scanners or another capture devices [11]. The nature of this devices allow to obtain a finite and discrete amount of data from the original object, commonly as a point set. The process of finding a model that fits better this observation set is a goal of many investigations in the last years [5,15,21,35].

Implicit polynomials (from now on IPs) are proved to be a powerful tool modelling real objects compared to other representation types, like explicit or parametric [34], with surprising variety of forms (Fig. 1, "Cherries" and "The pear") and good properties where required:

- A compact surface representation, i.e. concise, efficient description of the object, using few parameters [8]
- Algebraic and geometric invariants, for instance, area and volume [30]
- A fast way for classifying points as internal or external to the object
- Robust algorithms in presence of noise and occlusion [34]

1.1. Implicit polynomials

Implicit curve or surface is a zero set of a polynomial function f. For instance, in the case of a surface, the function is expressed as:

$$\begin{aligned} f_a(x,y,z) &= \sum_{0\le i+j+k\le n} a_{ijk}x^iy^jz^k \\ &= \underbrace{(1\ x\ y\ z\ x^2\ \dots\ )}_{m(\mathbf{x})^T}\underbrace{(a_{000}\ a_{100}\ a_{010}\ a_{001}\ a_{200}\ \dots\ )^T}_{\mathbf{a}} \\ &= m(\mathbf{x})^T\mathbf{a} \end{aligned}$$

IP is being seen as product between a monomial vector $m(\mathbf{x})^T$ that only depends on the point (x, y, z), and the parameters vector $\mathbf{a}$.

Formally, an implicit polynomial surface is the solution set of the following equation:

$$m(\mathbf{x})^T\mathbf{a} = 0 \qquad (1)$$

where $\mathbf{x}$ is the variable. This solution set is denoted as $Z(f_a)$. In both cases - 2D and 3D - usually implicit polynomial term is used.

The main contribution of this research is to develop an IP fitting algorithm *capable of discovering an optimal polynomial degree to be used* during the fitting process.

This paper presents two contributions. A new distance measure between X and the implicit polynomial is defined, and an

* Corresponding author.
E-mail addresses: rinterian@ic.uff.br, rubenus@yandex.ru (R. Interian).

http://dx.doi.org/10.1016/j.cag.2017.05.002
0097-8493/© 2017 Elsevier Ltd. All rights reserved.

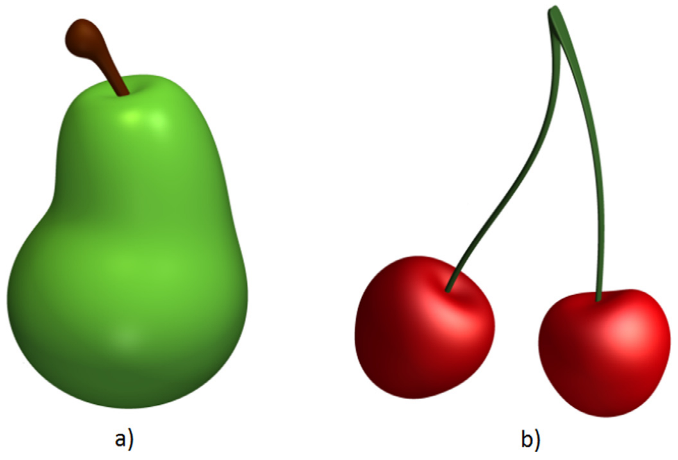


Fig. 1. Implicit Polynomials. (a) "The pear", 6*th* degree IP (b) "The cherries". Taken from [24].

algorithm able to find the optimal degree of the polynomial needed for the representation of the data set is proposed. The goal is to find an implicit polynomial that produces a compact and smooth representation of the point set with the lowest degree as possible, and at the same time minimizes the fitting error. The coefficients of the implicit polynomial can be used then as a descriptor for other tasks, for instance, shape analysis and recognition [7], in which complex and extensive representations are not useful.

1.2. Structure of present work

This work is organized as following. In the Section 2, main IP fitting methods are presented. In the Section 3, a novel distance measure between the point set and an IP is introduced. In the Section 4, fitting algorithm that uses the new measure in order to find IPs with certain desired properties, without previous knowledge of the best degree to be used in the fitting process, is proposed. In the Section 5, experimental results obtained from the proposed method are analyzed.

2. IP fitting methods

Fitting methods are classified as linear or nonlinear according to how the distance $dist(x_i, Z(f_a))$ between a point x_i and the zero set $Z(f_a)$ is defined.

2.1. Nonlinear methods

There is no simple way to find analytically the distance from any point to $Z(f_a)$. Approximate iterative methods are used instead [16,20,33].

The main idea of the nonlinear IP fitting methods is to use the Taubin first order approximation to the exact point-curve or point-surface distance [28,29,31]:

$$dist(\mathbf{x}, Z(f_a)) \approx \frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}$$

This kind of distance is usually named *geometric*, because it uses information from the partial derivatives of f_a.

2.2. Linear methods

The linear methods have been used most [6,36], because they do not require iterative approximations and are faster than nonlinear ones. The most important linear fitting algorithms are classic linear fitting and 3L algorithm.

The linear methods use the algebraic distance:

$$dist(\mathbf{x}, Z(f_a)) \approx f_a(\mathbf{x})$$

The following assumptions are made. By continuity, the value of f_a is close to zero near $Z(f_a)$. It is also assumed that far from $Z(f_a)$, f_a is growing.

In this case, the fitting problem is solved as a overdetermined system $M_{n\times k}\,\mathbf{a}_{k\times 1} = \mathbf{b}_{n\times 1}$ where M is the monomial matrix, which rows are the n monomial vectors of each point in the set X, and $\mathbf{b}$ is (*initially*) a zero vector.

The least squares solution of this overdetermined system [10] is:

$$\mathbf{a} = (M^T M)^{-1} M^T \mathbf{b} = M^+ \mathbf{b}$$

where M^+ is the *Moore–Penrose pseudoinverse* [3].

However, the linear algorithms suffer instability, because small changes in the observations can lead to completely different solutions [6]. Furthermore, since $\mathbf{b} = 0$, there is a trivial solution $\mathbf{a} = 0$ that should be avoid. Thereby this classical linear method is improved by other linear algorithms, like the 3L algorithm [6]. The stability of fitting is increased, and the vector $\mathbf{b}$ is substituted by another nonzero vector.

3. New distance measure to an IP

In order to obtain a good fitting algorithm, a new distance measure between a point and an IP is defined. In this regard, firstly two specific measures (called dissimilarity and smoothness measures) are deduced, with the aim of evaluating the "separation" and the "proximity" of the point set to $Z(f_a)$.

3.1. Dissimilarity measure

The dissimilarity measure evaluates *quantitatively* the real distance between the IP and the points.

First order Taubin approximation [31] is chosen for providing fast and analytically substantiated approximation to the point-IP distance:

$$dissim(\mathbf{x}, Z(f_a)) = \frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}$$

Therefore, for calculating the dissimilarity between a point set $\mathbf{X}$ and the IP, we propose to use the following function by averaging the Taubin approximations in each point:

$$dissim(\mathbf{X}, Z(f_a)) = \frac{1}{N}\sum_{i=1}^{N} \frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} \tag{2}$$

This function is called hereinafter *dissimilarity measure* between $\mathbf{X}$ and $Z(f_a)$.

3.2. Smoothness measure

In some of the latest works [21,27,34,35], the idea of taking control of some geometric characteristics of the IPs over the course of the fitting algorithms, is proposed. These characteristics could be: gradient vector norm near the point set, gradient direction relative to some estimated tangent vector in some points, as well as some others. All of them attempt to avoid the effects of overfitting, promoting those polynomials which have desired geometric properties.

For those reasons, "smoothness" measure is also used in order to provide a geometric proximity criterion between the IP and the dataset [34]. This measure evaluates *qualitatively* any IP as an approximation to the given set of points. The advantage of using this *quality* measure consists in obtain only IPs with certain desired topological properties, as shown below.

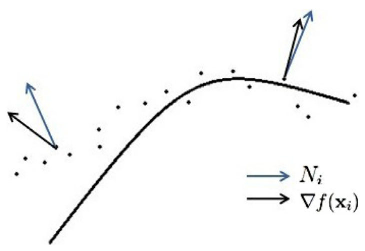


Fig. 2. Smoothness measure. If the directions of the estimated normal vector and the gradient vector are very different, the quality of the fitting is poor.

Given a dataset $\mathbf{X}$, an IP is smooth in $x_i \in \mathbf{X}$ if:

$$\frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|} N_i \approx 1$$

where N_i is an estimated normalized normal vector in $\mathbf{x}_i$. It can be obtained from physical models or estimated from $\mathbf{X}$.

Care should be taken when generating N_i vectors in a noisy point cloud, since inaccurate generation of these vectors can lead to poor fitting results. There are several algorithms addressing this problem [12,22,23]. In particular, we refer to Sahin method [22,23], in which this issue is tackled. In this work, the existence of a set of N_i vectors computed by any efficient method is assumed.

We use the following expression [35] as the *smoothness measure* of an IP regarding the point set $\mathbf{X}$:

$$smooth(\mathbf{X}, Z(f_a)) = \frac{1}{N} \sum_{i=1}^{N} \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|} N_i \tag{3}$$

We note that this function is bounded between -1 and 1 due to the normalization of both, gradient and normal vectors. Nevertheless, in practice, if the directions of the estimated normal vectors are determined accurately, the smoothness measure should never be close to -1. We note that smoothness can be interpreted as the average of cosine values of the angles between the gradient vector and the estimated normal vector, for every point of the dataset. If the cosine value is close to -1, that means the angle is close to 180 degrees. If that happens, it implies inversion of orientation which we should avoid because we are working with orientable surfaces. In the worst case, the gradient can be perpendicular to the estimated normal vector, causing smoothness be zero at this point.

In the Fig. 2, the intuitive idea of the smoothness measure is illustrated.

If the fitting result has good quality, the normalized gradient vector $\frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|}$ is close to the normal vector N_i in any point $\mathbf{x}_i$.

3.3. Penalization strategy

One of the contributions of this work is to propose a combination of the measures described above as an objective function (from now on OF) used in the fitting process. It is common to use *quantitative* OF in linear and nonlinear fitting methods, i.e. a function that describe an approximation to real distance from some IP to the points. Our approach include an objective function that penalizes non-smooth solutions during the optimization process, promoting higher *quality* IPs.

A strategy based on the idea of penalizing non-smooth IP solutions along the fitting process is described. Given a point set $\mathbf{X}$, we say that an IP is non-smooth if $smooth(\mathbf{X}, Z(f_a)) \approx 0$, that is:

$$1 - smooth(\mathbf{X}, Z(f_a)) \approx 1$$

Therefore, a new OF that penalizes non-smoothness is defined as the sum of two terms. The first term is the dissimilarity measure. The second term reflects how far the IP is from being smooth.

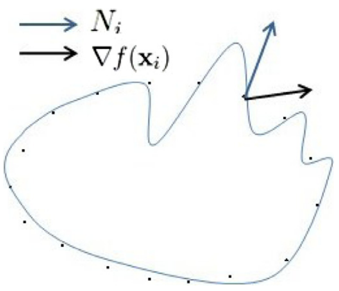


Fig. 3. Overfitting. The implicit polynomial pass close all the points in $\mathbf{X}$; it is well evaluated quantitatively, but has a poor qualitative evaluation, since it can be improved [27] by eliminating the turns it makes when passing through several points.

The positive constant δ has the role of penalty coefficient:

$$dist(\mathbf{X}, Z(f_a)) = dissim(\mathbf{X}, Z(f_a)) + \delta(1 - smooth(\mathbf{X}, Z(f_a)))$$

Substituting the expressions of dissimilarity and smoothness measures, and grouping the terms, we get:

$$dist(\mathbf{X}, Z(f_a)) = \frac{1}{N} \sum_{i=1}^{N} \frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} + \delta\left(1 - \frac{1}{N} \sum_{i=1}^{N} \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|} N_i\right)$$

$$dist(\mathbf{X}, Z(f_a)) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} + \delta\left(1 - \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|} N_i\right)\right) \tag{4}$$

This formulation of the distance function attempts to reconcile the two concepts we exposed in past sections: the proximity between the point cloud and the polynomial, and geometrical properties that the polynomial must have. The positive parameter δ indicates how strong is the penalty for not having those desired properties.

The δ parameter acts in a similar way to ridge regularization parameter α [22]. The aim of both parameters is to improve stability. However, there is no equivalence between the two. While in ridge regression the parameter works on global stability [22] (possibly causing local inaccuracy), in our method the value of δ acts locally by fixing gradient directions and improving local stability.

The distance function (4) is taken as the new OF that should be minimized during the fitting process:

$$\min_{f_a} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} + \delta\left(1 - \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|} N_i\right)\right) \tag{5}$$

Note that the IP can pass through all the points in $\mathbf{X}$ having positive OF value, for example, as in Fig. 3. We define now the fitting algorithm based on the distance measure (4).

4. An adaptive fitting algorithm

The main contribution of this work is to provide an algorithm capable of finding the degree of the polynomial needed for the representation of the points, without previous knowledge of the complexity of the dataset. This kind of algorithm is called hereinafter “adaptive fitting algorithm”.

Firstly, a fixed degree fitting algorithm is discussed. Subsequently, this algorithm is generalized to the variable degree case.

4.1. Fixed degree fitting

The minimization of the objective function defined above is proposed:

$$\min_{f_a} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{|f(\mathbf{x}_i)|}{\|\nabla f(\mathbf{x}_i)\|} + \delta\left(1 - \frac{\nabla f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|} N_i\right)\right)$$

Besides having not derivable nominator, it is well known that curve and surface fitting problems commonly lead to multimodal functional dependencies of the solution from data, even those associated to linear methods, as explained very well in [6]. This means that the objective function may have several local optimums. Exact methods are not efficient in this kind of problems, converging naturally to closest of them.

Therefore, metaheuristic algorithms are used in order to realize a better exploration of the search space and most important, *refine* several solutions provided by classical fitting methods, improving the quality of those. Those that were designed specifically for continuous problems are chosen, such as Particle Swarm Optimization and Differential Evolution [13,25]. Both have been previously used in curve fitting problems [18].

4.1.1. Optimization by PSO metaheuristic

The Particle Swarm Optimization metaheuristic (PSO) keep a set (swarm) of N particles traveling in a d-dimensional space [13]. Every particle represents a solution, and has associated a position and a speed vector, this last indicating the direction and the step of the movement. Each time some particle moves, it readjust its speed using information from the swarm, such that the search process is directed to promising search space regions. The parameter χ, called restriction factor, allows to bound speed values. In this way, the success of some particles affects the behavior of the others.

This algorithm has only two parameters: χ, the restriction factor, and N, the size of the swarm.

4.1.2. Optimization by DE metaheuristic

The Differential Evolution metaheuristic (DE) is an Evolutionary Algorithm and can be seen as a variation of the Genetic Algorithm [25]. Like any genetic algorithm, DE uses a population of N individuals represented by d-dimensional vectors. In summary, the algorithm generate new individuals using sum and difference operations over vectors in the population. In recent years this metaheuristic has been widely used in many practical problems, specially continuous ones [19].

Besides the population size N, DE has two parameters: weighting factor F, which controls the amplification of the variation obtained from the vector difference; and crossover constant CR, which indicates the percentage of the new individual vector components taken into account in the vectors of the next generation.

4.1.3. Seeded elements

Small number of seeded elements is included into the initial population of the iterative algorithms. Seeded elements typically have good values of the objective function, and is assumed that they increase the fitness of the population.

Exact algorithms solutions can be used as seeded elements. In particular, we use solutions of linear classic and 3L algorithms. The solutions of these linear methods can be obtained very quickly. The proposed number of seeded elements is $\frac{N}{10}$, where N is the total size of the population. The role of the metaheuristic is to improve the quality of the exact method solutions if they have good fitness respect to the distance measure defined above.

4.2. Adaptive fitting

The following question arises: what criteria should be followed for finding the optimum IP degree that best represents some dataset $\mathbf{X}$? In order to find an answer, several aspects of this problem are discussed below.

Table 1
Number of IP coefficients (problem dimension) from polynomial degree.

Degree	Coefficients, 2D	Coefficients, 3D
1	3	4
2	6	10
4	15	35
6	28	84
8	45	165
10	66	286
12	91	455
14	120	680
16	153	969

4.2.1. Maximum IP degree selection

Some practical considerations on IP degrees utilization are exposed.

In most research works addressing IP fitting problem, only even degrees are used in experiments or practical examples [6,14,22,34,35].

Taubin's work [32] has the explanation. Zero set of any *odd degree* polynomial is always unbounded[1]. Even degree polynomial zero sets can be bounded or unbounded. Therefore, are only considered even degree polynomials, due to finite nature of the observed datasets.

Furthermore, on the same researches, the most used IP degrees are in the range from 2 to 10. The use of superior degrees (12, 14, 16 and 18) is exceptional. In fact, the advantage of having a compact representation of the object is lost when these degrees are utilized (see the Table 1).

4.2.2. Smoothness measure behaviour analysis increasing IP degree

The key to determining the optimum IP degree is the behavior of the smoothness measure when this degree is increasing. The Fig. 4 shows the evolution of the above mentioned measure of the solutions of two fitting algorithms, for two distinct datasets.

In both cases, the smoothness measure increases, and then stabilizes its growth. In one case, the measure reaches the maximum, and then drops slightly. In the second case, asymptotic behavior is observed. The 3L algorithm solutions have higher quality than the solutions of the classical fitting algorithm.

These considerations lead us to use the change of the smoothness measure as a criterion of closeness to optimum IP degree. If the increment of the smoothness is very small or negative (the smoothness decreases) the “best” degree is reached.

4.2.3. Summary of the adaptive fitting algorithm

Taking into consideration all the above, the following fitting algorithm is proposed:

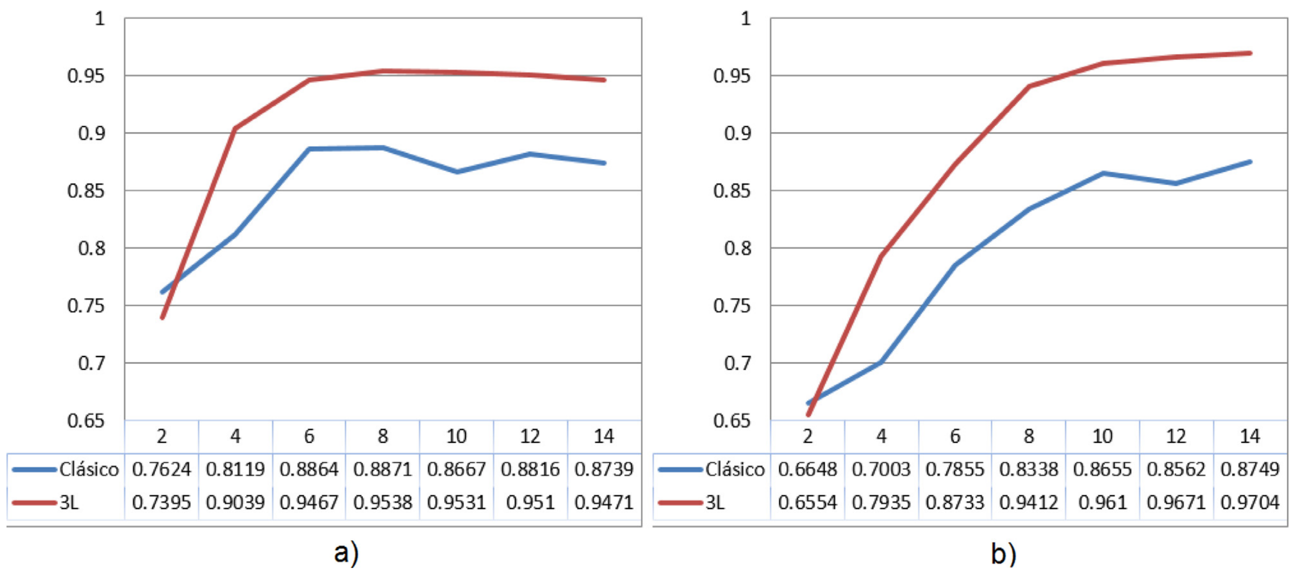
The stopping criterion of this algorithm is the non-increment of the smoothness measure in some ϵ. If $\epsilon = 0$, this is equivalent to a decrement of the smoothness.

One advantage of using this stopping criterion in the algorithm is that it has a clear geometric interpretation. The smoothness is the average of cosine values of the angles between the gradient vector and the estimated normal vector, for every point of the dataset. For instance, setting the value $\epsilon = 0.01$ in the stopping criterion represents a reduction of those angles in approximately 1°.

The following properties of the adaptive fitting algorithm can be exploited:

- It is able to find the degree of the IP that is necessary to get a good fitting result *without any previous knowledge of the complexity of the dataset.*

[1] For instance, any line in the plane, or any plane in the space, are zero sets of one-degree polynomials, and also unbounded.



a)

	2	4	6	8	10	12	14
Clásico	0.7624	0.8119	0.8864	0.8871	0.8667	0.8816	0.8739
3L	0.7395	0.9039	0.9467	0.9538	0.9531	0.951	0.9471

b)

	2	4	6	8	10	12	14
Clásico	0.6648	0.7003	0.7855	0.8338	0.8655	0.8562	0.8749
3L	0.6554	0.7935	0.8733	0.9412	0.961	0.9671	0.9704

Fig. 4. Smoothness measure behaviour increasing IP degree, datasets: (a) “Boot”, (b) “Horse” (*Large Geometric Models Archive*, *Georgia Institute of Technology* [1]).

- It introduces variability of the solutions, allowing to generate a set of them as output of the algorithm. With this we capitalize the multimodal nature of the IP fitting problem[2], which is an advantage over exact algorithms.
- It utilizes and improves the quality of the IPs that are solutions of the classic algorithms. This can be seen as a post-optimization process. The metaheuristics can generate new IPs that are unrelated to classic solutions, if they have desired geometric properties.

Algorithm 1 Adaptive fitting algorithm.

```
$G \leftarrow$ Maximum degree to be used for the current dataset
$g \leftarrow 2$
while $g \le G$ do
  $best\_elements_g \leftarrow FixedDegree(g)$
  if $g > 2$ and $smooth(best\_elements_g) - smooth(best\_elements_{g-1}) \le \epsilon$ then
    return $best\_elements_{g-1}$
  else
    $g \leftarrow g + 2$
  end if
end while
return $best\_elements_g$
```

5. Experimental results

Several issues concerning the practical implementation of the proposed algorithms are clarified.

One aspect that should be considered when applying fitting algorithms is the need of centering the dataset $\mathbf{X}$ at the coordinate origin. Furthermore, the dataset must be scaled (for example, dividing the points by their average distance from the coordinate origin). These transformations avoid numerical problems in fitting algorithms.

The experiments evaluate the performance of the adaptive fitting method in real 2D and 3D scenarios. Our main goal is to evaluate the quality of the solutions. The algorithm should generate curves and surfaces that are interpretable and similar to the original 2D and 3D objects.

[2] A multimodal problem has many local optima, which may have close values of the objective function.

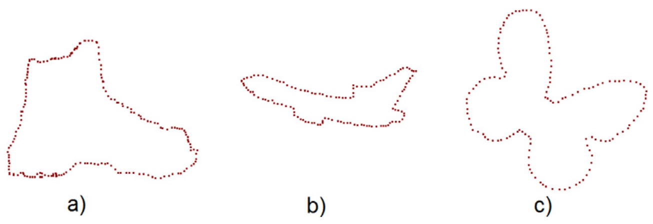


Fig. 5. 2D datasets. (a) Boot, (b) Airplane, (c) Butterfly.

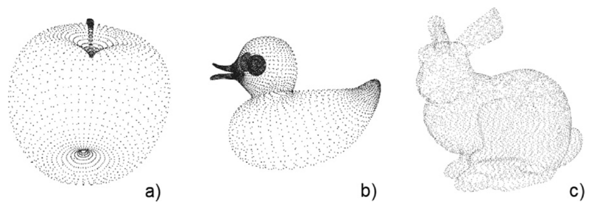


Fig. 6. 3D datasets. (a) Apple, (b) Rubber duck, (c) Stanford Bunny.

5.1. Datasets

The 2D and 3D datasets are selected as follows.

Point clouds are taken from widely known repositories, such as Stanford 3D Scanning Repository [2] and others (Fig. 6). In particular, we take the following models:

- Apple
- Rubber duck
- Stanford Bunny

In the case of 2D models, plane figures that appear frequently in implicit polynomial fitting research [6,27,35] are chosen (Fig. 5). There is not neither consensus nor repositories containing the ideal datasets for two-dimensional fitting. Moreover, 2D models are easily reproducible by researchers in the area.

5.2. Parameters of the algorithms

The following parameters were used in the experiments.

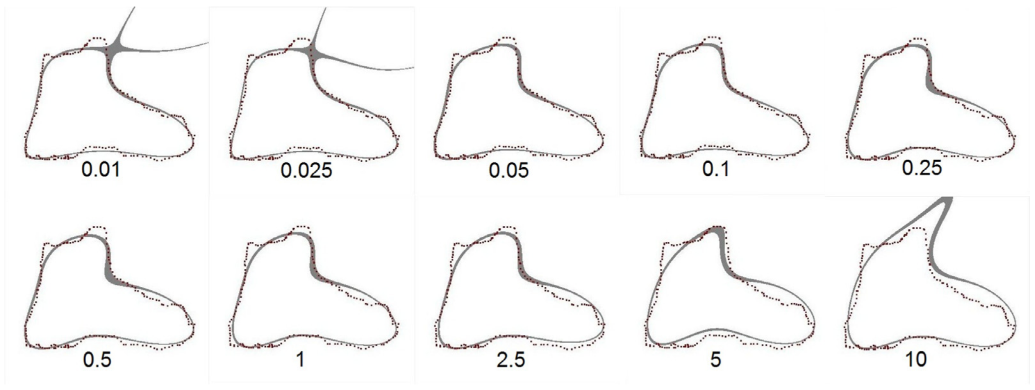


Fig. 7. Dataset “Boot”: setting parameter δ, 4th degree. Below each image is shown the value of δ.

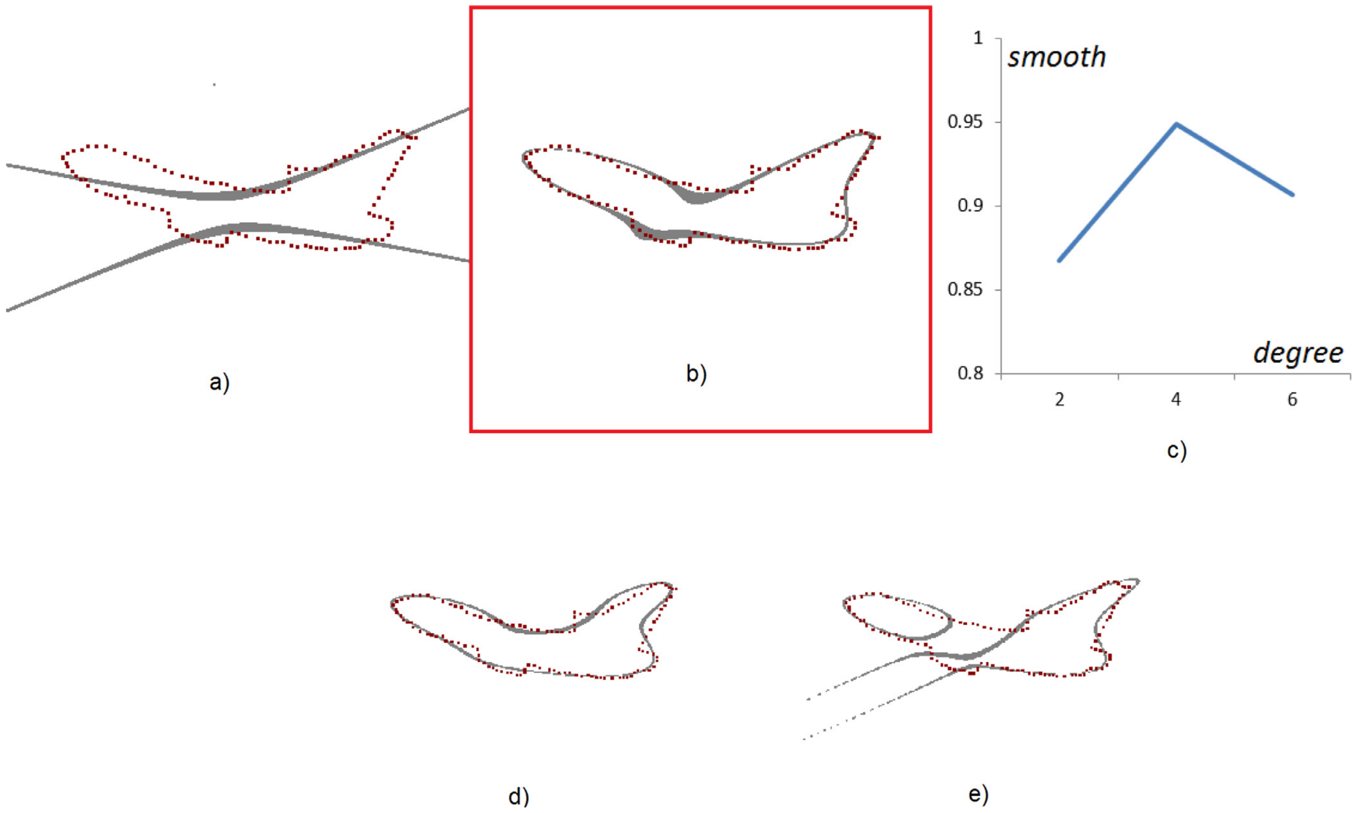


Fig. 8. Dataset “Airplane”, adaptive fitting run: (a) 2nd degree fit, (b) 4th degree fit (optimal degree), (c) evolution of the smoothness measure during the execution, (d) 3L fit for the 4th degree, (e) lineal classic fit for the 4th degree.

In DE and PSO metaheuristics, the population (swarm) size is set to four times the dimension of the problem [17], that is, four times the dimension of the parameters vector **a**. As stopping condition in both metaheuristics, the criterion of reaching 500 iterations is used. The run times of both algorithms are very similar.

In PSO metaheuristic, value $\chi = 0.729$ is taken, as suggested in [9].

In DE metaheuristic, values for weighting factor $F = 0.7$ and crossover constant $CR = 0.9$ are assumed, as recommended in [26].

The penalization parameter δ from the objective function (5) is estimated in Section 5.3.

Finally, for 3D models, fractions of the total number of points in the original dataset are taken, since the cardinality of these sets is very large, becoming 35,947 points in the dataset Stanford Bunny. This is because these models are often used in research in the field of computer graphics, where local accuracy is critical.

5.3. Setting parameter δ

The parameter δ introduced in the fitting algorithm, determines how strong is the penalty for having too low values of the smoothness measure.

Setting this parameter correctly is essential to obtain results that can be used and interpreted correctly, because a low value can eliminate the necessary effect of the penalty, and a high value can cause the implicit polynomial to stay away from the original set of observations.

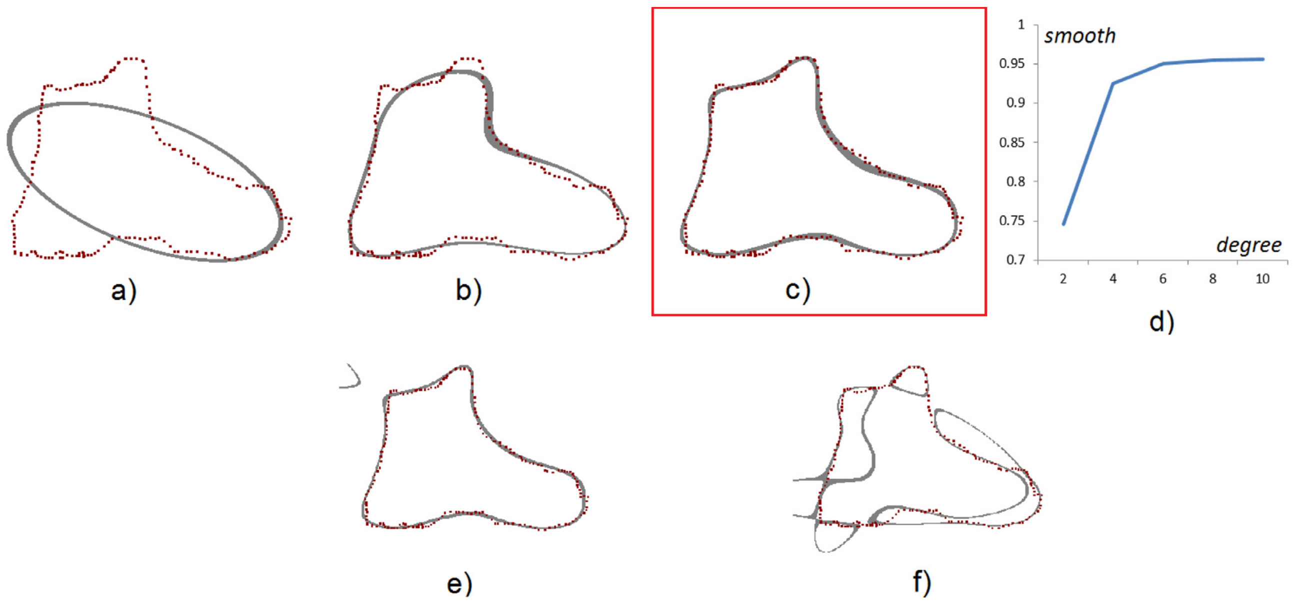


Fig. 9. Dataset "Boot", adaptive fitting run: (a) 2nd degree fit, (b) 4th degree fit, (c) 6th degree fit (optimal degree for $\epsilon = 0.01$), (d) evolution of the smoothness measure during the execution, (e) 3L fit for the 6th degree, (f) lineal classic fit for the 6th degree.

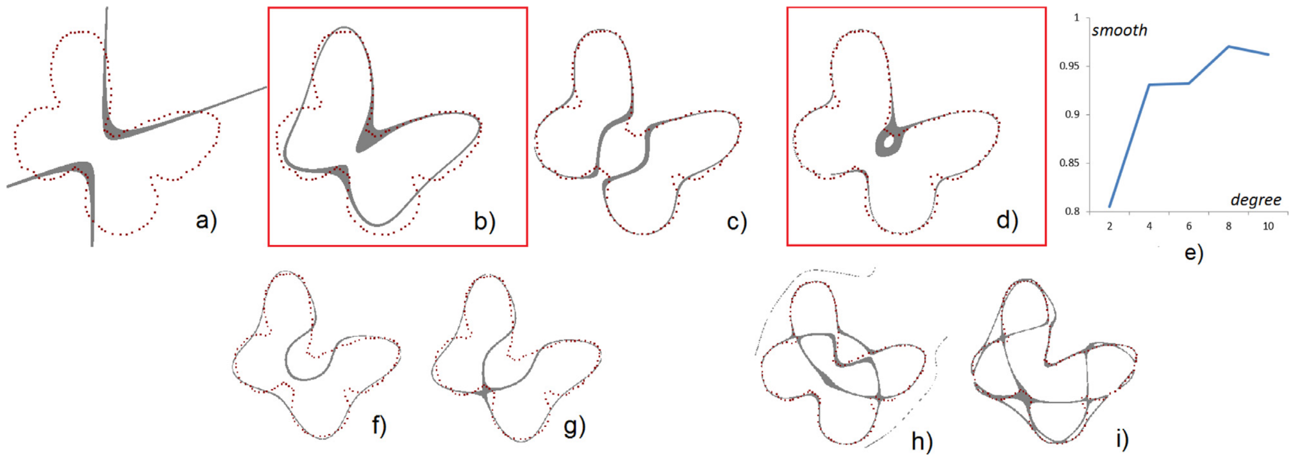


Fig. 10. Dataset "Butterfly", adaptive fitting run: (a) 2nd degree fit, (b) 4th degree fit (optimal degree for $\epsilon = 0.01$), (c) 6th degree fit, (d) 8th degree fit (optimal degree for $\epsilon = 0$), (e) evolution of the smoothness measure during the execution, (f) 3L fit for the 4th degree, (g) lineal classic fit for the 4th degree, (h) 3L fit for the 8th degree, (i) lineal classic fit for the 8th degree.

This fact is illustrated in the Fig. 7. Unstable behavior of the algorithm for small values of δ, as well as imprecise results for the large values, is observed. There is also a wide range of acceptable values for δ. We use the value $\delta = 0.25$ in the experiments described below.

5.4. Analysis of adaptive fitting results

The results of adaptive fitting algorithm for the different datasets are presented.

5.4.1. Fitting 2D curves

Adaptive fitting algorithm runs for 2D datasets are shown (Figs. 8–10), comparing the obtained results with classic linear and 3L algorithm results.

As observed, the adaptive algorithm can determine the degree of implicit polynomial needed to obtain an interpretable fitting.

5.4.2. Fitting 3D surfaces

Just as in 2D, experimental runs of the adaptive fitting algorithm for 3D datasets are executed. The Figs. 11–13 show the obtained results as well as the fits obtained by 3L and classic linear algorithms for the same datasets.

It is remarked that the algorithm is able to represent complex objects by relatively low degree implicit polynomials (degrees 2, 4, 6, 8).

5.4.3. Discussion of the graphical results

The results we presented in Figs. 8–13 can be divided into two groups. The first one contains datasets in which the smooth function has a strict maximum (Figs. 8, 11 and 12). In this cases, any nonnegative value for ϵ leads to easily identifiable optimum degree, in particular, $\epsilon = 0$ or $\epsilon = 0.01$.

The second group contains point clouds where the smoothness is a strictly increasing function (like Figs. 9 and 13). In order to reach correct results, the value of ϵ must be strictly positive, for example $\epsilon = 0.01$.

There is also a "Butterfly" dataset (Fig. 10), an example of a smoothness function with a strict maximum that has some unusual behavior. In this case, any of the aforementioned values of ϵ (0 and 0.01), leads to different but interpretable results.

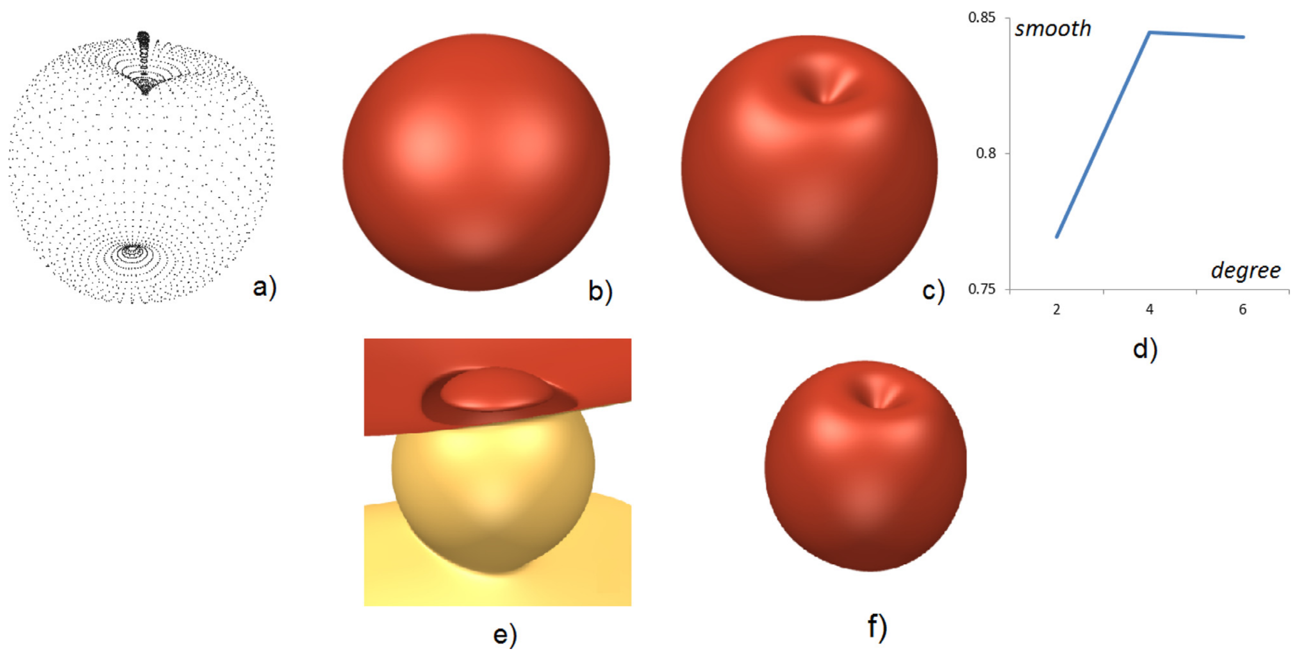


Fig. 11. Dataset “Apple”, adaptive fitting run: (a) original point cloud, (b) 2nd degree fit, (c) 4th degree fit (optimal degree), (d) evolution of the smoothness measure during the execution, (e) lineal classic fit for the 4th degree, (f) 3L fit for the 4th degree.

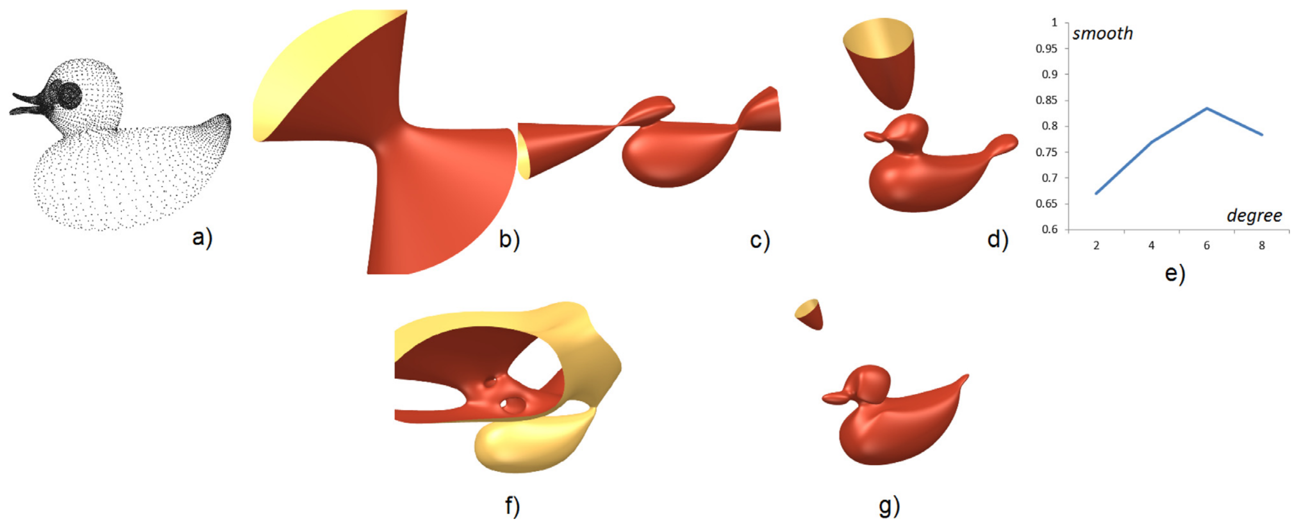


Fig. 12. Dataset “Rubber duck”, adaptive fitting run: (a) original point cloud, (b) 2nd degree fit, (c) 4th degree fit, (d) 6th degree fit (optimal degree), (e) evolution of the smoothness measure during the execution, (f) lineal classic fit for the 6th degree, (g) 3L fit for the 6th degree.

All these examples suggest that that the value of ϵ, however small, must be strictly positive. The value 0.01 seems to be a good strategy in all 6 cases we analyzed.

5.5. Comparison with 3L method plus ridge regression regularization

Ridge regression regularization (RRR) is a method that improves global stability of other linear methods, like 3L algorithm. It consists in adding the term kD to the M^TM matrix used in these methods (see Section 2 and [22] for further details). The variable k is the ridge regression parameter that should be small enough to preserve properties of the original matrix, but large enough to achieve global stability.

We now compare the performance of our method against 3L with RRR (3L+RRR) using some selected values of k parameter. The results are shown in Figs. 14 and 15. We use 3L+RRR with the same degree founded by our algorithm. Unlike the 3L method with ridge regression regularization, our method preserves important shape features, like a pronounced corner in the tail of the airplane, or the central part of the butterfly. The ridge regression is unable to achieve this kind of results for these datasets, since it only cares about global stability. Increasing k parameter, 3L+RRR fitting results tend to be more “circular”, eliminating corners and protrusions.

5.6. Fitting in presence of noisy data

To analyze the behavior of the algorithm when the data is noisy, we generated Gaussian noise in the original point clouds, with different standard deviation values. The results are presented in Fig. 16. Normal vectors N_i are generated using a simple algorithm presented in [12]. Our fitting algorithm always stopped in degree 4. The presented empirical evidence indicates that the method is

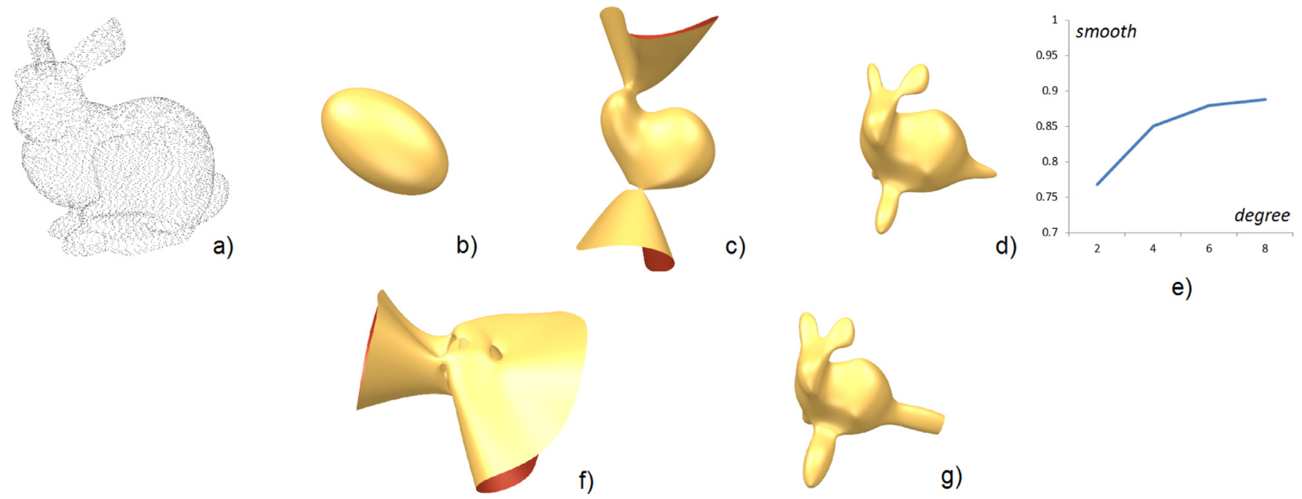


Fig. 13. Dataset "Stanford Bunny", adaptive fitting run: (a) original point cloud, (b) 2nd degree fit, (c) 4th degree fit, (d) 6th degree fit (optimal degree for $\epsilon = 0.01$), (e) evolution of the smoothness measure during the execution, (f) lineal classic fit for the 6th degree, (g) 3L fit for the 6th degree.

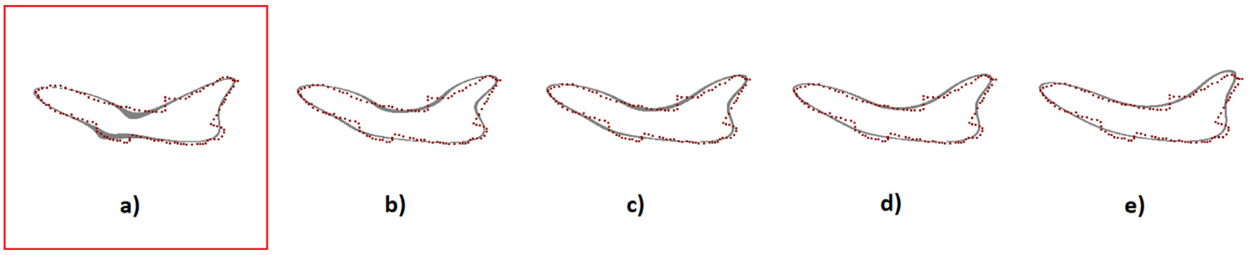


Fig. 14. Dataset "Airplane", comparing with 3L+RRR method: (a) Our method, (b) 3L method without RRR ($k = 0$), (c) 3L+RRR, $k = 10^{-8}$, (d) $k = 3 * 10^{-8}$, e) $k = 10^{-7}$.

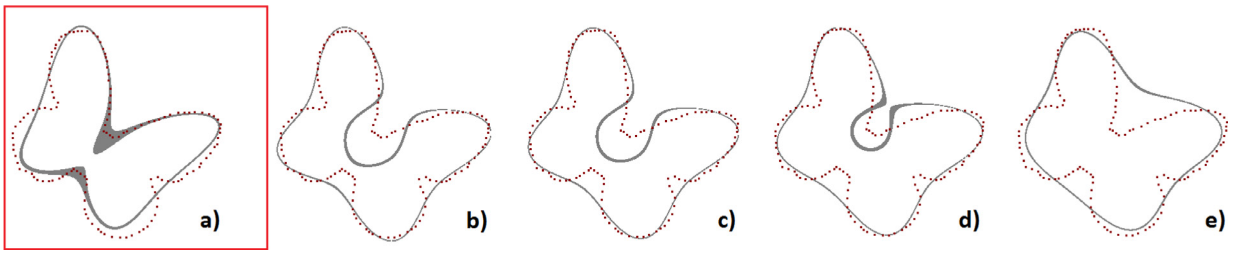


Fig. 15. Dataset "Butterfly", comparing with 3L+RRR method: (a) Our method, (b) 3L method without RRR ($k = 0$), (c) 3L+RRR, $k = 10^{-5}$, (d) $k = 3 * 10^{-5}$, (e) $k = 10^{-4}$.

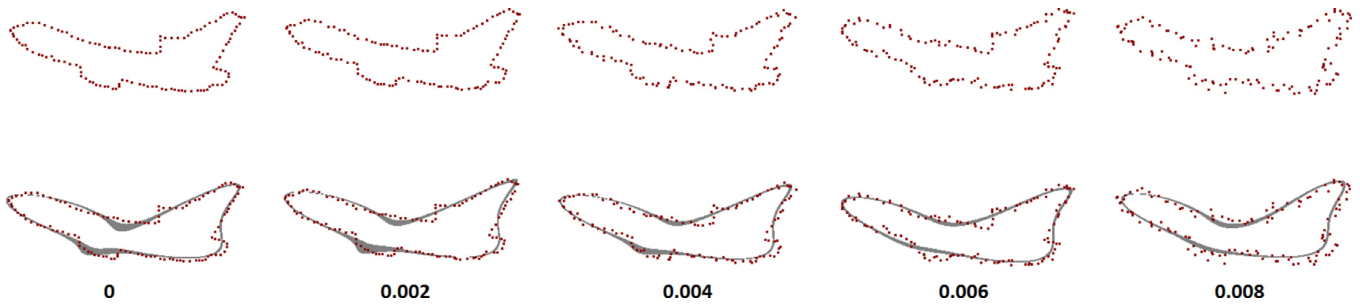


Fig. 16. Fitting in presence of noisy data. Below each image is shown the standard deviation value of Gaussian noise.

little sensitive to noise, which allows it to be used for recognition tasks.

6. Conclusions

Although the implicit polynomials are not the most locally precise representation scheme, they are very useful for applications requiring a compact registration of data from a complex real-world object, in order to perform a process of recognition [4] of the same, or other objects that correspond to the pattern. With the aim of finding an IP that properly represents some dataset, a fitting process is performed. The vast majority of fitting algorithms requires knowledge of the IP degree that best represents the points. This work presents an alternative to these fitting algorithms.

Firstly, an objective function to be used during the fitting process is defined. This function is characterized by including a penalty of undesired geometric properties (non-smoothness) in the polynomial.

Then, heuristic adaptive fitting algorithm is proposed, which is able to find the degree of the implicit polynomial that is necessary to represent the dataset. The algorithm is based on the idea of gradually increasing the degree of the IP, while there is an improvement in the smoothness of the solutions.

This algorithm is beneficial in comparison to others for three reasons: it can automatically find the required degree of the implicit polynomial, it can offer a set of solutions in contexts where variability (options) is required, and it can post optimize solutions of known fitting algorithms (classical linear and 3L), in order to obtain better quality solutions.

The experiments confirm the validity of the approach for the selected 2D and 3D datasets, since the fits made by the proposed algorithm are interpretable.

Acknowledgments

Ruben Interian has been sponsored by a CAPES scholarship. The work of Celso C. Ribeiro was partially supported by CNPq and FAPERJ research grants. Anselmo A. Montenegro is grateful for the fund from FAPERJ.

References

[1] Greg Turk and Brendan Mullins. Large geometric models archive, Georgia Institute of Technology. http://www.cc.gatech.edu/projects/large_models/ (accessed May 2017).
[2] The Stanford 3D Scanning Repository, Stanford Computer Graphics Laboratory. http://graphics.stanford.edu/data/3Dscanrep/ (accessed May 2017).
[3] Ben-Israel A. Generalized inverses: theory and applications. second. Springer; 2001.
[4] Ben-Yaacov H. Recognition of 3D objects based on implicit polynomials. Irwin and Joan Jacobs Center for Communication and Information Technologies; 2009.
[5] Berger M, Levine J, Nonato LG, Taubin G, Silva CT. An end-to-end framework for evaluating surface reconstruction. Sci Comput Imag Inst 2011.
[6] Blane MM. The 3l algorithm for fitting implicit polynomial curves and surfaces to data. IEEE Trans Pattern Anal Mach Intell 2000;22:298–313.
[7] Bronstein AM, Bronstein MM, Kimmel R. Numerical geometry of non-rigid shapes. Monographs in Computer Science. 1st. New York: Springer-Verlag; 2009.
[8] Campbell RJ, Flynn PJ. A survey of free-form object representation and recognition techniques. Comput Vis Image Underst 2001;81(2):166–210. http://dx.doi.org/10.1006/cviu.2000.0889.
[9] Eberhart R, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. Proc Congr Evol Comput 2000;1:84–8.
[10] Golub G. Matrix computations. third. The Johns Hopkins University Press; 1996.
[11] Gomes AJP, Voiculescu I, Jorge J, Wyvill B, Galbraith C. Implicit curves and surfaces: mathematics, data structures and algorithms. 1st. London: Springer-Verlag; 2009.
[12] Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Surface reconstruction from unorganized points. SIGGRAPH Comput Graph 1992;26(2):71–8.
[13] Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of the IEEE international conference on neural networks, 4; 1995.
[14] Landa Z. 2D object description and recognition based on contour matching by implicit polynomials. In: The 18th European Signal Processing Conference Proceedings; 2006. p. 1796–800.
[15] Liu T, Liu W, Qiao L, Luo T, Peng X. Point set registration based on implicit surface fitting with equivalent distance. Proceedings of the IEEE international conference on image processing; 2015.
[16] Mederos V, Estrada J. A new algorithm to compute the euclidean distance from a point to a conic. Invest Oper 2002;23.
[17] M. Pedersen, Good parameters for particle swarm optimization (2010), technical report no. HL1001 of Hvass Laboratories.
[18] Polo-Corpa M. Curve fitting using heuristics and bio-inspired optimization algorithms for experimental data processing in chemistry. Chemom Intell Lab Syst 2009;96:34–42.
[19] Price K, Storn R. Differential evolution - a practical approach to global optimization. Springer; 2006.
[20] Rouhani M. Implicit polynomial representation through a fast fitting error estimation. IEEE Trans Image Process 2012;21:2089–98.
[21] Rouhani M. The richer representation the better registration. IEEE Trans Image Process 2013;22:5036–49.
[22] Sahin T. Fitting globally stabilized algebraic surfaces to range data. Proceedings of the IEEE international conference of computer vision, 2; 2005.
[23] Sahin T, Unel M. Stable algebraic surfaces for 3D object representation. J Math Imag Vis 2008;32:127–37.
[24] Schenzel P. On the interactive visualization of implicit surfaces. Martin-Luther University Halle Press; 2012.
[25] Storn R, Price K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 1997;11:341–59.
[26] Talbi E. Metaheuristics from design to implementation. John Wiley & Sons, Inc.; 2009.
[27] Tasdizen T. Improving the stability of algebraic curves for applications. IEEE Trans Image Process 2000;9:405–16.
[28] Taubin G. Nonplanar curve and surface estimation in 3-space. Proceedings of the IEEE international conference on robotics and automation,; 1988. p. 644–5.
[29] Taubin G. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. Pattern Anal Mach Intell IEEE 1991;13:1115–38.
[30] Taubin G. Object recognition based on moment (or algebraic) invariants. Geometric invariance in computer vision. MIT Press; 1992. p. 375–97.
[31] Taubin G. Distance approximations for rasterizing implicit curves. ACM Trans Graph 1994;13.
[32] Taubin G. Parameterized families of polynomials for bounded algebraic curve and surface fitting. IEEE Trans Pattern Anal Mach Intell 1994;16:287–303.
[33] Taubin G, Cooper D. Symbolic and numerical computation for artificial intelligence. Academic Press; 1992.
[34] Zheng B. 2D curve and 3D surface representation using implicit polynomial and its applications. University of Tokyo; 2008. Ph.D. thesis.
[35] Zheng B. An adaptive and stable method for fitting implicit polynomial curves and surfaces. IEEE Trans Pattern Anal Mach Intell 2010;32:561–8.
[36] Zheng B. Breast MR image fusion by deformable implicit polynomial (DIP). IPSJ Trans Comput Vis Appl 2013;5:99–103.

APPENDIX I – Tumor growth modelling by cellular automata

MATHEMATICS AND MECHANICS OF COMPLEX SYSTEMS
Vol. 5, No. 3-4, 2017
dx.doi.org/10.2140/memocs.2017.5.239

TUMOR GROWTH MODELLING BY CELLULAR AUTOMATA

RUBEN INTERIAN, REINALDO RODRÍGUEZ-RAMOS,
FERNANDO VALDÉS-RAVELO, ARIEL RAMÍREZ-TORRES,
CELSO C. RIBEIRO AND AURA CONCI

Tumor growth is a complex process that requires mathematical modeling approaches for studying real-life cancer behavior. The use of cellular automata (CA) to represent tumor growth in its avascular stage is explained in this work, and a stochastic CA describing tumor growth is obtained, based on a differential equations system in the range of continuum mechanics. The novelty of this research is the deduction of the neighborhood structure and rules for a probabilistic CA from these differential equations that describe the evolution of the tumor growth. In addition, the influence of the stresses on tumor growth is captured by the CA.

1. Introduction

Cell-based and cell-centered approaches for the study of biological soft tissues have been widely used [Hoehme and Drasdo 2010; Merks and Glazier 2005]. In particular, *cellular automata* (CA) are one of the most successful models [Rejniak and Anderson 2011; Boondirek et al. 2010] and have been used in a large number of studies [Kansal et al. 2000b; Dormann and Deutsch 2002; Deutsch and Dormann 2005].

Cellular automata models have been widely used to simulate avascular tumor growth [Dormann and Deutsch 2002; Kansal et al. 2000a], tumor cell invasion [Anderson et al. 2009], and tumor interactions with various environmental factors [Rejniak and McCawley 2010]. A review of the main methodologies for CA models describing tumor growth is provided in [Boondirek et al. 2010], emphasizing that most researchers have attempted to consider a microscopic scale to describe the macroscopic characteristics of tumor morphology. In [Kansal et al. 2000b], a three-dimensional cellular automaton model of brain tumor growth is developed, simulating the Gompertzian model very precisely. A quantitative analysis of the growth of a subpopulation within a previously homogeneous tumor is presented in

Communicated by Francesco dell'Isola.
MSC2010: 37B15, 68Q80, 74A10.
Keywords: tumor growth, stresses, stochastic rules, cellular automata.

[Kansal et al. 2000a], applying a cellular automaton with a Delaunay triangulation lattice. In [Dormann and Deutsch 2002], avascular tumor growth is simulated using a hybrid lattice-gas cellular automaton, exhibiting self-organized formation of a layered tumor structure. A multiscale investigation focused on tumor cell invasion is given in [Anderson et al. 2009], using an evolutionary hybrid cellular automata model (EHCA). A comparison of relative strengths and weaknesses of various cell-based models, including CA, is presented in [Rejniak and McCawley 2010].

In this work, cellular automata are used for studying the process of tumor growth, specially in its avascular stage. The novelty of this research is the deduction of the neighborhood structure and the rules for a stochastic CA from continuous deterministic *differential equation* (DE) models. In addition, both the application of CA to study the evolution of tumor growth in time and the influence of the stresses in the growing of the tumor are considered in this work. The validation of the theoretical biomechanical model using CA is presented as well.

Having a CA model created from a system of differential equations allows one to describe more realistically the real-life situation of the tumor stage. The goals of the present work are summarized as follows:

(1) To visualize the tumor growth process represented by a system of differential equations.

(2) To obtain the evolution of the variables over time, mainly the tumor radius, in a new manner, more realistic and closer to the reality, where the tumor is not perfectly regular or circular. To reach this goal, a continuous deterministic DE model is transformed into a *probabilistic* CA.

(3) To create an alternative for studying the process of tumor growth in an interactive mode.

2. Cellular automata

Cellular automata have been seen mainly as discrete abstract computational systems. The Stanford Encyclopedia of Philosophy [Berto and Tagliabue 2012] defines cellular automata as follows:

> Firstly, CA are (typically) spatially and temporally discrete: they are composed of a finite or denumerable set of homogeneous, simple units, the atoms or cells. At each time unit, the cells instantiate one of a finite set of states. They evolve in parallel at discrete time steps, following state update functions or dynamical transition rules: the update of a cell state obtains by taking into account the states of cells in its local neighborhood (there are, therefore, no actions at a distance). Secondly, CA are abstract, as they can be specified in purely mathematical terms and implemented

in physical structures. Thirdly, CA are computational systems: they can compute functions and solve algorithmic problems. Despite functioning in a different way from traditional, Turing machine-like devices, CA with suitable rules can emulate a universal Turing machine, and therefore compute, given Turing's Thesis, anything computable.

Rigorously, the cellular automata is a quadruple (C, n, S, f) [Deutsch and Dormann 2005], where:

- C is a set of cells, not required to be finite.
- $n : C \times C \to \{0, 1\}$ is a neighborhood function that can been seen as a relationship (usually reflexive and symmetric) between the cells. This function shows which pairs of cells are neighbors, that is, the geometry of the cell organization. Furthermore, n must satisfy the *neighborhood size independence condition*: $|N(c_0)| = |\{c \in C : n(c_0, c) = 1\}| = N$ is a constant for every $c_0 \in C$; i.e., the size of the neighborhood is the same for all cells.
- S is a set of states. As discussed below, each cell will have an associated state, in each moment.
- $f : S^{|N|} \to S$ is a transition function. The transition function is a core of the CA dynamics and is commonly expressed with rules that define the state of the cell in the next time moment from the state of the cell neighbors.

The set of cells C with the neighborhood function n defines the structure of the cell space.

The simplest CA model can have binary cells (two states, "tumoral" or "normal" [Hu and Ruan 2003]). Commonly, the states are represented by a set of integer values $\{0, 1, 2, \dots\}$, each of these values having an appropriate physical or biological interpretation.

3. Linear elasticity tumor model

This work focuses on the mathematical model developed in [Ramírez-Torres et al. 2016], where the authors using linear elasticity generalize the Ngwa–Agyngi model [2012], which describes the evolution of growth-induced stresses in a spherical and isotropic growing tumor surrounded by an external medium.

The generalization of the model considering real cases of stresses is discussed in [Ramírez-Torres et al. 2016], where the dependence of tumor growth on the stresses is analytically derived. This particular model explores the avascular stage of a solid tumor.

This section briefly describes the model. We note that, in this article, we do not focus on its derivation but, instead, on obtaining from the model a CA that represents the tumor growth, as described in the next section.

Variable	Definition
t	Time
r	Radial coordinate
$R(t)$	Radius of the tumor at time t
R_0	Radius of the tumor at $t = 0$
$c(r, t)$	Nutrient concentration inside the tumor
c_b	Nutrient concentration at boundary
$\boldsymbol{u}(r, t)$	Tumor cell displacement
$\boldsymbol{v}(r, t)$	Tumor cell displacement velocity
$\boldsymbol{\sigma}$	Cauchy stress tensor of the tumor
σ_r	Cauchy stress tensor component in radial direction
σ_θ	Cauchy stress tensor component in transversal direction
$\boldsymbol{\sigma}^e$	Cauchy stress tensor of the external medium
E	Young's modulus
ν	Poisson's ratio
$\boldsymbol{e}$	Strain tensor of the tumor
$\boldsymbol{e}^e$	Strain tensor of the external medium
ρ	Tumor cell density
γ_r	Tumor growth anisotropy parameter (radial direction)
γ_θ	Tumor growth anisotropy parameter (transversal direction)
η_1	Dependence of cell proliferation on stresses
η_2	Dependence of cell death on stresses

Table 1. Model variable definitions.

The variables and constants of the model are shown in Table 1.

The mathematical model obtained by [Ramírez-Torres et al. 2016] is a generalization of [Ngwa and Agyingi 2012] as a result of the followings assumptions:

(i) Tumor cells form a homogeneous population that is considered a continuum.

(ii) There is adhesion between tumor cells at the boundary, which maintains the tumor's solid shape and is in equilibrium with the expansive forces exerted by the internal cell proliferation.

(iii) The tumor has a spherical shape, and its symmetry is maintained at all times.

(iv) The tumor is in a state of diffusion equilibrium.

(v) The nutrient consumption rate is proportional to the nutrient concentration and to the tumor cell density. Without stresses, the cellular proliferation rate is proportional to the nutrient concentration and to the tumor cell density, while cell death is proportional to the cellular density.

(vi) The tumor material is assumed to be incompressible and responds to stress in a purely elastic and isotropic form.

(vii) There is a constant nutrient concentration in the tumor boundary.

(viii) There is an external medium, which is supposed to be elastic, isotropic, and incompressible.

The tumor is modeled as a solid in the three-dimensional space, and the forces on it are considered acting per volume unit. As a result of the spherical symmetry hypothesis, the problem is treated in one dimension with respect to the radial coordinate r.

3.1. ***Kinematics and equilibrium equations.*** Because of the radial symmetry (hypothesis (iii)), the surface of the tumor is given by $S = r - R(t)$ and the velocity field has the form $\boldsymbol{v} = (v_r, 0, 0)$, leading to

$$\frac{dR}{dt} = v_r(R, t). \tag{1}$$

This equation represents the growth rate of the tumor. Removing the inertial factors and considering hypothesis (i), the equilibrium equation is

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{F} = 0, \tag{2}$$

where $\boldsymbol{F}$ is the vector of body forces, which is considered null.

3.2. ***Constitutive equation.*** The constitutive relation, which associates the stress σ_{ij} with the material strain e_{ij}, represents a material with a linear elastic response subject to an anisotropic growth:

$$e_{ij} = g(\delta_{1i}\gamma_r + (\delta_{2i} + \delta_{3i})\gamma_\theta)\delta_{ij} + \frac{1+\nu}{E}\sigma_{ij} - \frac{\nu}{E}\delta_{ij}\sigma_{kk} \quad \text{with } i, j, k = r, \theta, \phi, \tag{3}$$

where g is the growing factor, δ_{ij} is Kronecker's delta, $\gamma_r, \gamma_\theta \in \mathbb{R}_+$, and $\gamma_r + 2\gamma_\theta = 1$. Parameters γ_r and γ_θ represent the proportions of the tumor growth in the radial and transversal directions, respectively.

Assuming small deformations, $\boldsymbol{e} = \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)$, using the material incompressibility (i.e., $\nu = \frac{1}{2}$) from hypothesis (iv), and applying a Jaumann derivative in (3), we obtain the relationship between the rates of deformation and stress

$$\tfrac{1}{2}(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T)_{ij} = (\nabla \cdot \boldsymbol{v})(\delta_{1i}\gamma_r + (\delta_{2i} + \delta_{3i})\gamma_\theta)\delta_{ij} + \frac{1}{2E}\frac{D}{Dt}(3\boldsymbol{\sigma} - \boldsymbol{I}\,\mathrm{tr}(\boldsymbol{\sigma}))_{ij}. \tag{4}$$

3.3. ***External medium.*** From assumption (viii), the external medium satisfies the generalized Hooke's law

$$\sigma^e_{ij} = \frac{E\nu}{(1+\nu)(1-2\nu)}\delta_{ij}e^e_{kk} + \frac{E}{1+\nu}e^e_{ij}, \tag{5}$$

since the material is incompressible, $\nu = \frac{1}{2}$, and (5) becomes

$$\sigma^e_{ij} = -p\delta_{ij} + \frac{2E}{3}e^e_{ij}, \tag{6}$$

where p is the isotropic pressure.

3.4. ***Growth equation.*** For a living tissue, growth can be interpreted as the difference between cell production and cellular death. Then, from the mass-conservation continuity equation and hypothesis (v), we have

$$\underbrace{\frac{\partial \rho}{\partial t} + \nabla \cdot (\boldsymbol{v}\rho)}_{\text{growth}} = \underbrace{\alpha c\rho(1+\eta_1 \operatorname{tr}(\boldsymbol{\sigma}))}_{\text{cellular proliferation}} - \underbrace{k\rho(1-\eta_2 \operatorname{tr}(\boldsymbol{\sigma}))}_{\text{cellular death}}, \tag{7}$$

where $\eta_1, \eta_2 \in \mathbb{R}_+$ are constants representing the dependence of cellular proliferation and death on stress.

As a consequence of tumor incompressibility (assumption (vi)), we get from (7)

$$\nabla \cdot \boldsymbol{v} = \alpha c(1+\eta_1 \operatorname{tr}(\boldsymbol{\sigma})) - k(1-\eta_2 \operatorname{tr}(\boldsymbol{\sigma})). \tag{8}$$

3.5. ***Nutrient concentration.*** The nutrient concentration variation is determined by nutrient diffusion through the boundary of the tumor and its consumption by tumor cells in the interior. From assumptions (iv) and (v), it is noticed that

$$\frac{\partial c}{\partial t} + \boldsymbol{v} \cdot \nabla c = \underbrace{D_c \nabla^2 c}_{\text{diffusion}} - \underbrace{A_c c\rho}_{\text{consumption}}, \tag{9}$$

where c represents the nutrient concentration, D_c is the diffusion rate (which is assumed to be constant), and A_c is the nutrient consumption rate. Moreover, assuming that the nutrient concentration variation is much smaller than its diffusion and consumption, (9) can be written as

$$D_c \nabla^2 c = A_c c\rho. \tag{10}$$

3.6. ***Nondimensionalization of the model.*** An important step in modeling is to work with nondimensional variables. With this purpose, we define the constants

$$L \equiv \sqrt{\frac{D_c}{A_c \rho}}, \quad \tau \equiv \frac{1}{\alpha c_b}, \quad c_b, \quad \text{and} \quad \epsilon \equiv \frac{k}{\alpha c_b},$$

which represent the length scale, time scale, constant nutrient concentration at the boundary, and ratio between the cellular death and cellular proliferation rates. We use asterisks to identify the nondimensional variables:

$$r^* = \frac{r}{L}, \quad \sigma^*_{ij} = \frac{\sigma_{ij}}{E}, \quad p^* = \frac{p}{E}, \quad v^* = \frac{v}{\alpha c_b L}, \quad t^* = \frac{t}{\tau}, \quad \text{and} \quad c^* = \frac{c}{c_b}.$$

These new variables are placed in (1), (2), (4), (6), (8), and (10). For simplicity of notation, the asterisks are removed.

The equation of nutrients, derived from (10), can be solved analytically. We thus get

$$c(r,t) = \frac{R \sinh r}{r \sinh R}. \tag{11}$$

Then taking into account the radial symmetry $\boldsymbol{v} = (v_r, 0, 0)$ and placing (11) in the equation resulting from the nondimensionalization of (8) leads to

$$\frac{\partial v_r}{\partial r}(r,t) = \frac{R\sinh(r)}{r\sinh(R)}[1+\eta_1 E(3\sigma_r(r,t)-2\beta(r,t))] \\ -\epsilon[1-\eta_2 E(3\sigma_r(r,t)-2\beta(r,t))] - 2\frac{v_r(r,t)}{r}. \tag{12}$$

Now, in matrix form, (4) reads

$$\begin{pmatrix} \frac{\partial v_r}{\partial r} & 0 & 0 \\ 0 & \frac{v_r}{r} & 0 \\ 0 & 0 & \frac{v_r}{r} \end{pmatrix} = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 v_r)\begin{pmatrix} \gamma_r & 0 & 0 \\ 0 & \gamma_\theta & 0 \\ 0 & 0 & \gamma_\theta \end{pmatrix} \\ + \frac{1}{2E}\left(\frac{\partial}{\partial t} + v_r\frac{\partial}{\partial r}\right)\begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix}, \tag{13}$$

with

$$\sigma_1 = 2\sigma_r - \sigma_\theta - \sigma_\phi,$$
$$\sigma_2 = 2\sigma_\theta - \sigma_r - \sigma_\phi,$$
$$\sigma_3 = 2\sigma_\phi - \sigma_r - \sigma_\theta,$$

where σ_r, σ_θ, and σ_ϕ are the diagonal components of the Cauchy stress tensor $\boldsymbol{\sigma}$. In (13) the diagonal elements are the only nonzero elements because of the symmetry assumptions. If the second diagonal element is subtracted from the third in the matrix equation, then

$$\left(\frac{\partial}{\partial t} + v_r\frac{\partial}{\partial r}\right)(\sigma_\theta - \sigma_\phi) = 0,$$

suggesting $\sigma_\theta = \sigma_\phi$ because the material derivative of $\sigma_\theta - \sigma_\phi$ is zero. Substituting this result in the first diagonal element of (13), we have

$$\left(\frac{\partial}{\partial t} + v_r\frac{\partial}{\partial r}\right)(\sigma_r - \sigma_\theta) = 2\left(\gamma_\theta\frac{\partial v_r}{\partial r} - \gamma_r\frac{v_r}{r}\right). \tag{14}$$

As in (4), the equilibrium (2) represents a system of partial differential equations, i.e., three partial differential equations for three unknown functions (σ_r, σ_θ, and σ_ϕ), and the only nonzero equality is that corresponding to the radial direction, i.e.,

$$\frac{\partial \sigma_r}{\partial r} + \frac{2(\sigma_r - \sigma_\theta)}{r} = 0. \tag{15}$$

Finally, from (1), (12), (14), and (15), the model is given by the system of first-order partial differential equations

$$\frac{dR}{dt}(t) = v_r(R,t) \qquad \text{for } t \in \mathbb{R}_+^*, \tag{16}$$

$$\begin{aligned}\frac{\partial v_r}{\partial r}(r,t) = {}& \frac{R\sinh(r)}{r\sinh(R)}[1+\eta_1 E(3\sigma_r(r,t)-2\beta(r,t))] \\ & -\epsilon[1-\eta_2 E(3\sigma_r(r,t)-2\beta(r,t))] \\ & -2\frac{v_r(r,t)}{r} \qquad \text{for } t \in \mathbb{R}_+^* \text{ and } r \in (0,R],\end{aligned} \tag{17}$$

$$\frac{\partial \sigma_r}{\partial r}(r,t) = -\frac{2\beta}{r} \qquad \text{for } t \in \mathbb{R}_+^* \text{ and } r \in [0,R), \tag{18}$$

$$\varpi(r,t) = \left(\frac{\partial}{\partial t} + v_r(r,t)\frac{\partial}{\partial r}\right)\beta(r,t) \quad \text{for } t \in \mathbb{R}_+^* \text{ and } r \in (0,R), \tag{19}$$

with

$$\begin{aligned}\varpi(r,t) &= 2\left(\gamma_\theta\frac{\partial v_r}{\partial r} - \gamma_r\frac{v_r}{r}\right) \\ &= 2\gamma_\theta\left(\frac{R\sinh(r)}{r\sinh(R)}[1+\eta_1 E(3\sigma_r(r,t)-2\beta(r,t))]\right. \\ &\qquad \left. -\epsilon[1-\eta_2 E(3\sigma_r(r,t)-2\beta(r,t))]\right) - 2\frac{v_r(r,t)}{r},\end{aligned}$$

subject to the initial and boundary conditions

$$\begin{aligned}R(0) &= R_0,\\ \sigma_r(r,0) &= 0,\\ \beta(r,0) &= 0,\\ v(0,t) &= 0,\end{aligned}$$

where $\beta = \sigma_r - \sigma_\theta$. Two conditions are still needed for β and σ_r.

(i) *Condition for β at $r = 0$.* Since $v_r = 0$ at $r = 0$ and the derivatives of β are bounded (because β is assumed C^1 in $[0, R]$ with respect to r), then

$$\frac{\partial \beta}{\partial t}(0,t) = \varpi(0,t).$$

Therefore, the first boundary condition for β is

$$\begin{aligned}\frac{\partial \beta}{\partial t}(0,t) = (2\gamma_\theta - 1)&\left(\frac{R}{\sinh(R)}[1-\eta_1 E(3\sigma_r(0,t)-2\beta(0,t))]\right. \\ &\left. -\epsilon[1+\eta_2 E(3\sigma_r(0,t)-2\beta(0,t))]\right) + \frac{\partial v_r}{\partial r}(0,t).\end{aligned}$$

(ii) *Condition for σ_r.* This condition may be determined if the constitutive equation (6) of the external medium is used assuming the continuity of the stresses at the tumor boundary. Because of the spherical tumor symmetry from hypothesis (iii) and the incompressibility of the external medium (hypothesis (viii)),

$$\operatorname{tr}(\boldsymbol{e}) = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 u_r) = 0.$$

Substituting the solution of the above equation in (6), we have

$$\sigma_r|_{r=R} = -\frac{4(R-R_0)}{3R}.$$

4. Cellular automata model definition

A cellular automaton based on a linear elasticity tumor model [Ramírez-Torres et al. 2016] is defined. The model described in [Ramírez-Torres et al. 2016] and summarized in the previous section is taken as an example of a system of differential equations (DE) that represents tissue growth, i.e., tumor growth in this case.

The present continuum-mechanical model considers the tumor as a solid in the three-dimensional space, and the forces on it are considered acting per volume unit. As a result of the spherical symmetry hypothesis, the body deformation is the same in two of the three principal directions and no shear deformations are accounted for. Moreover, from the symmetry condition, all the fields depend only on time t and on one spatial variable r. The three principal stresses are σ_r, σ_θ, and σ_ϕ, with (as shown in the text after (13)) $\sigma_\theta = \sigma_\phi$. In this sense, we only need to find σ_r and σ_θ, given that the stress in the direction orthogonal to the plane where they are contained, corresponding to the third principal stress, is equal to σ_θ and therefore not necessarily zero. Hence, the theoretical model is addressed as neither a plane strain nor as a plane stress problem [Sokolnikoff 1956].

On the other hand, for representing the cellular automaton model, a cross section passing through the origin of the tumor in the three-dimensional model is taken. It does not matter which cross section because, given the spherical symmetry of the continuous theoretical model, all the cross sections are equivalent. Therefore, the cellular automaton is constructed in a two-dimensional setting, although the model is easily extended to three dimensions. Indeed, the rules of the model are given for any spatial dimension. Hence, the stresses and deformations in the cellular automaton model are the same stresses and deformations which are completely contained in a cross section of the continuum-mechanical model.

The transformation of the deterministic DE linear elasticity tumor model into a cellular automaton is required. The neighborhood structure and the rules for a probabilistic CA are deduced from continuous differential equations. The CA solution presented in this work can be extended to other similar continuous DE models.

4.1. ***General model definition.*** In the present study, a simple two-state cellular automaton is considered, where a 0-state represents a normal cell and a 1-state exemplifies a tumoral cell.

The automaton is considered infinite because it is defined in a borderless square lattice in which there exists some central cell with coordinates (0, 0). The initial state assigned to each cell is 0, except for a finite number of them. Practical implementations of this model must maintain a finite matrix representation of the relevant part of the lattice and expand it on demand.

The neighborhood structure and a rule set that describes the continuous model behavior are explained in the following.

4.2. ***Neighborhood structure selection.*** There is an equivalence (1) between the tumor radius growth speed and the tumor cell displacement speed in the continuous model described in [Ramírez-Torres et al. 2016]:

$$\frac{dR}{dt} = v(R, t).$$

The speed v could be used in the definition of the transition rules of the CA because the "displacement speed" oriented outside the tumor is closely related to the tumor cell propagation. In addition, we know that v does not depend explicitly on the spatial coordinates of a point but instead on the radial one. This fact implies that the "influence" of a tumor is equally distributed in all directions from the center of the model.

In order to select the appropriate neighborhood structure, we should take into account this property of the tumor growth speed. Since all the discrete time steps are equal, the increments of the tumor cell coordinates should also be equal, for the same moment of time, in all directions. In the CA, the increment of the tumor cell coordinate is described as the propagation of some tumor cell into another normal cell.

Therefore, the influence zone (neighbor) of any cell c must have cells with the same distance to c. We can choose some constant distance q, and for any cell c_0, the neighborhood is defined as

$$\{c \in C : d(c_0, c) = q\},$$

where $d(c_1, c_2)$ is the common Euclidean distance in the unit square lattice.

We choose $q = 1$, which generates a well known von Neumann neighborhood [Deutsch and Dormann 2005]. It is possible to choose other values of q, such as $q = \sqrt{2}$ or $q = 2$, generating other unexplored neighborhoods.

4.3. ***Rule inference.*** The main principle used in the area of rule inference from continuous models [Guinot 2002] consists of creating a stochastic rule with the structure

$$s_c(t+\Delta t) = \begin{cases} s_c(t)+1, & X \le g(s_c(t), N(c), \Delta t), \\ s_c(t), & X > g(s_c(t), N(c), \Delta t), \end{cases} \tag{20}$$

where X is a random variable with uniform distribution in (0, 1) and Δt is the length of the time step. The hardest part of rule inference is precisely the correct and meaningful definition of g. Actually, function g expresses the idea of the "speed" of growing in some place of the lattice where conditions around the cell are expressed by the state of the cell itself $s_c(t)$ and by its neighborhood $N(c)$.

From the continuous model, the differential equation (1) shows the relationship between the tumor radius growth speed and the radius itself:

$$\frac{dR}{dt} = v(R, t).$$

Initially, we assume that $v(R, t) \ge 0$, i.e., the size of the tumor is nondecreasing.

We interpret $\frac{dR}{dt}$ as *the average number of new tumor cells created from one tumor cell* per unit of time. If $\frac{dR}{dt} < 1$, this speed is seen as the probability of appearance of a new tumor cell during the unit time step, i.e., function g. Otherwise, some normalization process over $\frac{dR}{dt}$ becomes necessary.

In the case when $\eta_1 = \eta_2 = 0$ (stress dependence is not considered), a closed-form expression for v can be obtained from (16)–(19):

$$\frac{dR}{dt} = \frac{\cosh R}{\sinh R} - \frac{1}{R} - \frac{\epsilon R}{3}, \tag{21}$$

where ϵ is a constant. The fact that $v(R, t) = v(R)$, depending only on the maximum tumor radius, is particularly important for the rule definition process because cellular automaton inherently does not have any notion of time: the next state depends only on the current state.

However, if $\eta_1 \ne 0$ or $\eta_2 \ne 0$, then the stress influence is taken into account and no closed-form expression can be obtained for v. In this case, $v(R, t)$ is approximated using numerical methods, like Euler's method or the Lax–Wendroff scheme from finite differences.

4.3.1. *Case without stress dependence.* If $\eta_1 = \eta_2 = 0$, stresses are not taken into account, and a closed-form expression (21) for v is obtained from the differential equations (16)–(19). The properties of this equation are analyzed in order to appropriately define g.

Stability analysis. Equation (21) has the form $R_t' = v(R)$, where R_t' is the derivative of the radius function with respect to time. In the classical numerical methods, we can use the fact that, for a small h,

$$R_t' \approx \frac{R_{t+1} - R_t}{h} \quad \text{and} \quad R_{t+1} \approx R_t + hv(R).$$

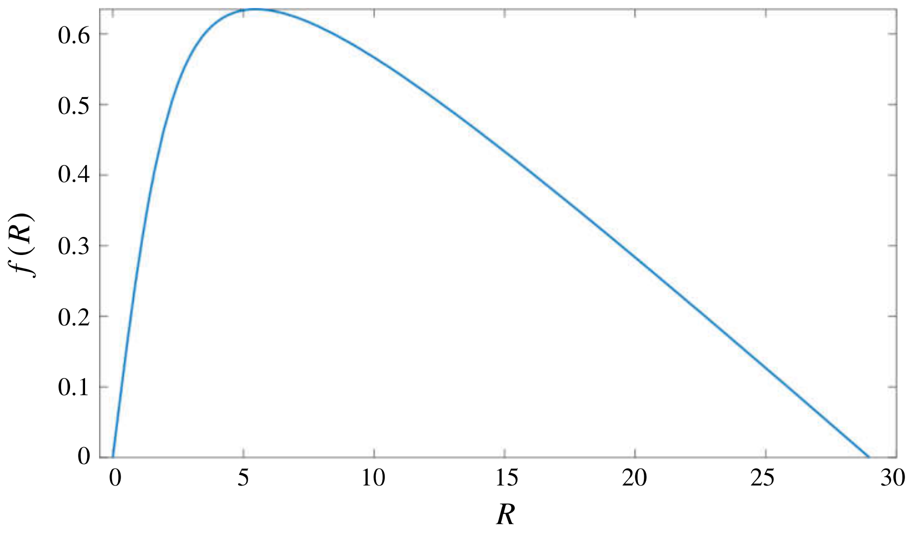


Figure 1. Case without stress dependence, stability analysis: increment function $(\cosh R)/(\sinh R) - 1/R - \epsilon R/3$ for $\epsilon = 0.1$.

It is clear that, for a discrete time step h, $v(R)$ represents the increment in the tumor radius from a time moment to the next. In terms of cellular automata, it represents the magnitude of propagation of a tumoral cell c into its normal neighbor. However, since CA are discrete computational systems in space, the tumoral cell cannot be expanded in a fractional amount of space.

In Figure 1, the increment $v(R)$ reaches a local maximum close to $v = 0.63$ and has a root close to $R = 29$. The tumor radius starts its growth at $R = R_0 \geq 0$. Since the increment is always positive, it grows asymptotically to the root value, where the increment is null.

We analyze the rule inference process, where the function g is appropriately defined. In rule (20), g indicates the probability of expansion of a tumoral cell into a normal neighbor. In this sense, the desired expansion probability is exactly the magnitude of propagation of the tumor cell. Since the increment $v(R) \in [0, 1]$ is bounded, it can be used as the probability g, simply setting $g = v$. If $v(R) \notin [0, 1]$, it would have been necessary to perform some normalization process over $v(R)$.

To summarize, if the stress dependence is not considered, the following stochastic rule emerges:

$$s_c(t + \Delta t) = \begin{cases} s_c(t) + 1, & X \leq v(r), \\ s_c(t), & X > v(r), \end{cases} \tag{22}$$

where X is a random variable with uniform distribution in $(0, 1)$,

$$v(r) = \frac{\cosh r}{\sinh r} - \frac{1}{r} - \frac{\epsilon r}{3},$$

and $r = \sqrt{i^2 + j^2}$ is the radial coordinate of a cell. Although the angular coordinate is not present in this rule, this does not mean that the model is symmetric in all polar directions. The asymmetry is present indirectly through random variable X whose realizations must be generated at every time step and for each cell [Guinot 2002], leading to an irregular shape.

4.3.2. *Case with stress dependence.* The assumption that $\eta_1 \neq 0$ or $\eta_2 \neq 0$ leads to a scenario where the stresses' influence is considered. In particular, no closed-form expression can be obtained for v in this case, making the use of numerical methods unavoidable. Therefore, v is approximated using Euler's method or the Lax–Wendroff scheme from finite differences [Ramírez-Torres et al. 2016]. More precisely, numerical approximations of $R(t)$ and $v(R, t)$ are obtained. This means that pairs $(R_i, v(R_i))$ are computed for each moment in time t_i, where

- R_i is the tumor radius at t_i and
- $v(R_i)$ is the tumor growth speed at any point of the tumor border at t_i.

There is no guarantee of the existence of a functional dependency between v and R, as in (21). In fact, there are cases where for the same value of R, more values of v are obtained. One example is shown in Figure 2, where no $v(R)$ can be defined because $R(t)$ is not injective, with two distinct growth speeds at the same radius, at two distinct moments of time.

We first analyze the case where the function $v(R)$ is approximately definable. Next, we examine, the case where the function $v(R)$ cannot be defined.

(i) *Definable* $v(R)$. If $R(t)$ is a strictly increasing function, then it is also injective and an ordered sequence of pairs $(R_i, v(R_i))$ is obtained, where all radius values

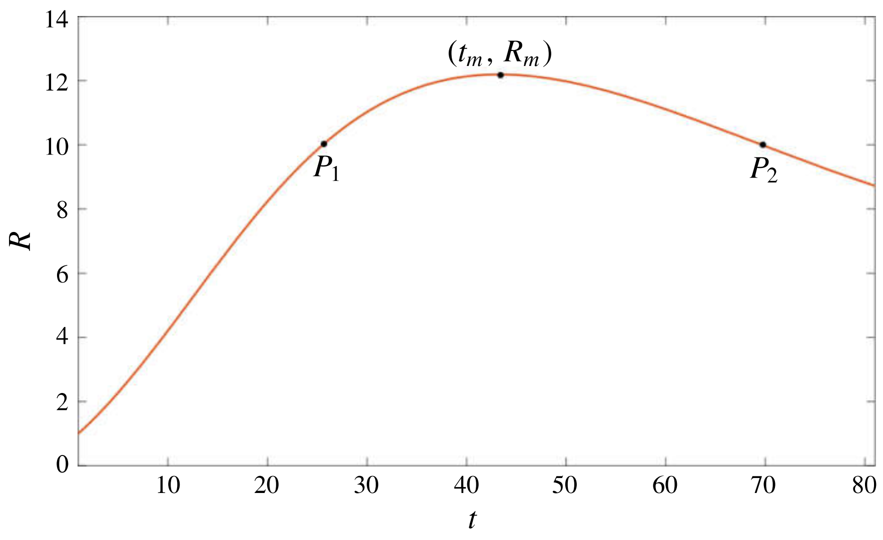


Figure 2. Tumor radius growth in time, theoretical model, strict extremum case.

are different. In this case, function $v(R)$ is approximately defined in the following way. To evaluate $v(R)$ for some $R = R_j$, *the speed value that corresponds to the closest to* R_j *among all the radius values in the sequence* is selected.

The smaller the time step becomes, the better the approximation is, since there is a higher density of available R values.

Given this definition of $v(R)$, the stochastic rule for the CA is defined in a similar way as in the case without stress dependence (22):

$$s_c(t+\Delta t) = \begin{cases} s_c(t)+1, & X \le v(R_i), \\ s_c(t), & X > v(R_i). \end{cases} \tag{23}$$

Here, $v(R_i)$ is the evaluation of the radius R_i that is *closest to the radial coordinate of the cell value*, from the available radius-speed evaluations.

The sequence of pairs $(R_i, v(R_i))$ is called from now on the *guide set* and represents an approximation to the actual v function. It is used as a guide during the CA evolution process.

(ii) *Undefinable* $v(R)$. If $R(t)$ is not a strictly increasing (or strictly decreasing[1]) function, the injectivity of $R(t)$ cannot be guaranteed and no simple functional dependency between v and R can be defined.

In particular, the special case where $R(t)$ has a strict extremum is considered. In Figure 2 an example is shown. The existence of some pair $(R_m, v(R_m))$ is assumed, where R_m is a maximum over the R values available in the guide set.

The impossibility of defining a single CA that represents the continuous model behavior is explained below. Choosing two time moments with the same tumor radius (points P_1 and P_2 in Figure 2), similar CA configurations are expected in both cases. Then since the cellular automaton is a memoryless system (no time variable or history can be present), it is not possible to distinguish between both states, in order to determine when it should grow or when a reduction should take place, since the next state of any CA only depends on its previous state.

In response to the need of having the tumor growth process representation in this case, a solution based on the use of two cellular automata is proposed. Having a sequence of the $(R_i, v(R_i))$ pairs ordered by i, the sequence is divided into two subsequences. The first is the sequence s_1 that starts with $(R_0, v(R_0))$ and ends with $(R_m, v(R_m))$; the second sequence s_2 contains the rest of the radius-speed pairs up to the last (see Figure 2).

It was shown that a guide set like s_1 can define an approximation to the *positive* v function needed to define the rule (23). Another rule that reflects tumor reduction

[1]The assumption that $v(R,t) \ge 0$ is relaxed here; i.e., the tumor may decrease with negative values of v.

is then required. In this sense, the following rule structure is proposed:

$$s_c(t+\Delta t) = \begin{cases} s_c(t) - 1, & X \le |v(R_i)|, \\ s_c(t), & X > |v(R_i)|. \end{cases} \tag{24}$$

This rule is applied to the tumoral cells with normal neighbors, where X is a random variable with uniform distribution in (0, 1) and $v(R_i)$ is the evaluation of the radius R_i from s_2 that is closest to the radial coordinate of the cell r.

Finally, the complete model is defined as an ordered pair (A_1, A_2), where:

- A_1 corresponds to the first of the two automata, with stochastic rule (23), and guide set s_1 and
- A_2 corresponds to the second automaton, with its initial state being the final state of A_1, stochastic rule (24), and guide set s_2.

4.3.3. *Rule definition summary.* To summarize, a new stochastic model of tumor growth obtained from a differential equation system is provided. A random variable X with uniform distribution in (0, 1) is considered. If a closed-form expression of the growth speed v in the tumor border can be found, then the following rule is defined:

$$s_c(t+\Delta t) = \begin{cases} s_c(t) + 1, & X \le v(r), \\ s_c(t), & X > v(r), \end{cases}$$

where $v(r) = (\cosh r)/(\sinh r) - 1/r - \epsilon r/3$ and $r = \sqrt{i^2 + j^2}$ is the radial coordinate of the cell c in the lattice.

If a closed-form expression for v cannot be found, then a guide set is defined as a collection of pairs $(R_i, v(R_i))$ *ordered by the time moment at which the approximation is made*. If $v \ge 0$, then the following first CA rule can be applied to the normal cells with at least one tumoral cell in its neighborhood:

$$s_c(t+\Delta t) = \begin{cases} s_c(t) + 1, & X \le v(R_i), \\ s_c(t), & X > v(R_i), \end{cases}$$

where $v(R_i)$ is the evaluation of v at the radius R_i from the guide set that is closest to the radial coordinate of the cell r. We remark that this rule is applied to the cells with tumoral neighbors and, in particular, to the cells on the border of the tumor, where $r \approx R$.

If $v < 0$, then a second CA rule is applied to the tumoral cells with normal neighbors:

$$s_c(t+\Delta t) = \begin{cases} s_c(t) - 1, & X \le |v(R_i)|, \\ s_c(t), & X > |v(R_i)|, \end{cases}$$

where, once again, $v(R_i)$ is the evaluation of v from the guide set at the closest point to the radial coordinate of the cell r.

5. Results

One important issue in any model is the validation of its consistency with the original phenomenon or another reference model. In this case, a comparison between the behavior of the discrete CA model described above and the continuous model is provided.

In this section, experimental results are discussed for all three cases of tumor growth: without stress dependence, with stress dependence and strictly increasing tumor radius, and with stress dependence and a tumor radius that has a strict extremum.

The CA model described above was implemented using the C# programming language on an Intel Core i7 machine, with a 2.0 GHz processor and 16 GB of random-access memory. All executions were very fast, taking a few seconds of running time.

Common values of the parameters $\epsilon = 0.1$, $R_0 = 1$, and $E = 64.0439$ were fixed in all computations. An important clarification is that the tumor radius value in the cellular automaton is computed as the average between the tumoral border cell distances to the center of the tumor.

5.1. ***Tumor growth without stress dependence.*** In the case where $\eta_1 = \eta_2 = 0$, the stresses are not taken into account in the mathematical model. Then the cellular automaton with rule (22) behavior is compared with the continuous differential equation model. The results are shown in Figure 3.

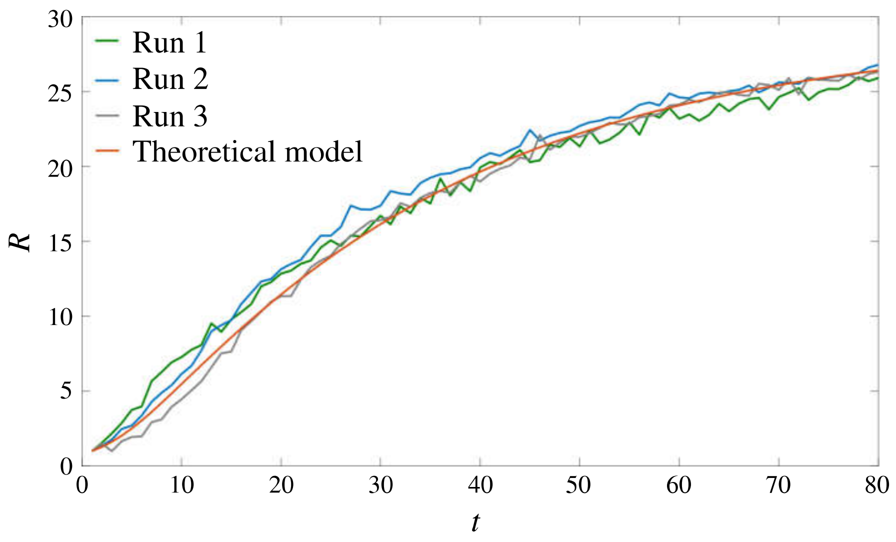


Figure 3. Graphic illustrating the evolution of the tumor radius with time, case without stress dependence. Comparison between several runs of the discrete CA model and the continuous (red line) model. The continuous model is obtained from (21), as a particular case of the model defined by (16)–(19).

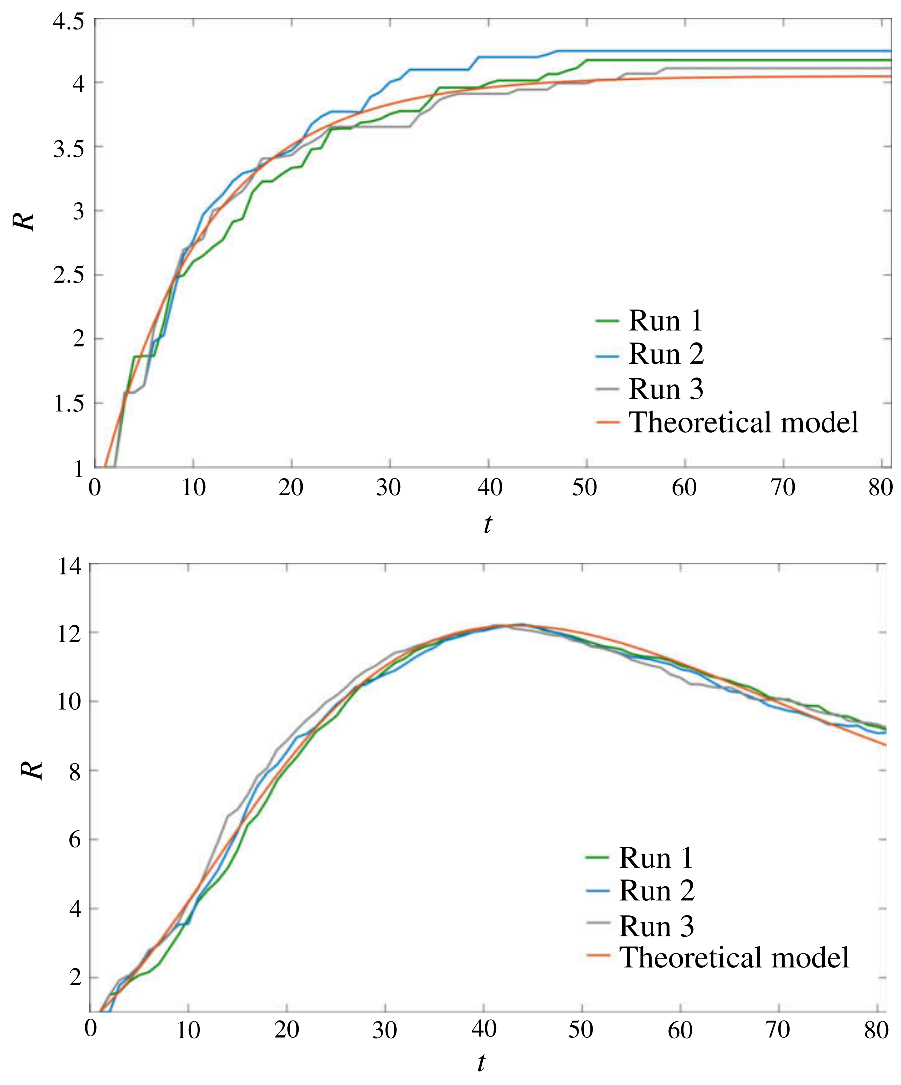


Figure 4. Graphics illustrating the evolution of the tumor radius with time, cases with stress dependence when the tumor radius is strictly increasing (top) and when the tumor radius has a strict extremum (bottom). Comparison between several runs of the discrete CA model and the continuous (red line) model.

The radius-time dependence graphics are similar and close to each other. A small difference between them is due to the dissimilar nature of the models.

5.2. ***Tumor growth with stress dependence.*** Similar experiments are performed in the case of stress-dependent tumor growth. Results are shown in Figure 4.

The case where the tumor radius is a strictly increasing function is obtained with $\gamma_\theta = \frac{1}{3}$ and $\eta_1 = 0.004$ (Figure 4, top). A cellular automaton with rule (23) is used.

The case where the tumor radius has a strict extremum is obtained with $\gamma_\theta = \frac{1}{10}$ and $\eta_1 = 0.002$ (Figure 4, bottom). In this case, two cellular automata with rules (23) and (24), respectively, are used.

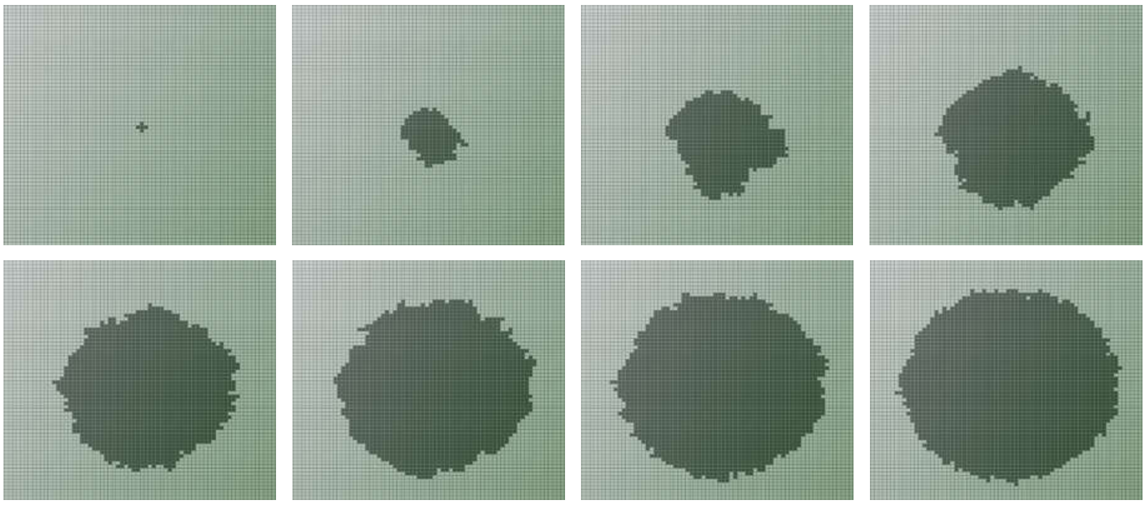


Figure 5. Visualizations of the CA tumor growth for time moments 0, 10, 20, 30, 40, 50, 60, and 70, respectively.

In both cases, close radius-time dependence graphics are observed. Stabilization of tumor radius growth is detected in the strictly increasing function case (Figure 4, top) in all runs. An excellent degree of proximity is achieved in the strict extremum case (Figure 4, bottom) with two cellular automata.

5.3. ***Tumor growth visualization in time.*** An example of the tumor growth visualization is shown in Figure 5. It corresponds to the case without stress dependence described in Section 5.1.

In particular, we notice an exponential, fast growth at the beginning, followed by an asymptotic behavior when the radius of the tumor stabilizes around a constant value. Further visualizations are not shown due to their similarity with those corresponding to $t = 60$ and $t = 70$. There is almost no change when the running time is increased because of the strong convergence of the tumor radius in the CA model.

This corresponds to the stability results obtained in Section 4.3.1 and shown in Figure 1, where the radius increment function has a zero at a radius value slightly below 29.

5.4. ***Dispersion of the executions.*** Finally, the dispersion of computational executions is evaluated. As an example, the case without stress dependence is considered, whose visualizations were shown in the previous section. Firstly, the *average execution* is computed as a sequence of averages of tumor radius values obtained from 100 CA model runs, for each time step. The resulting graphic is shown in Figure 6. It compares the CA *average execution* with the continuous model: both functions are very close to each other.

Figure 7 displays the coefficient of variation. Most of the chart is below the

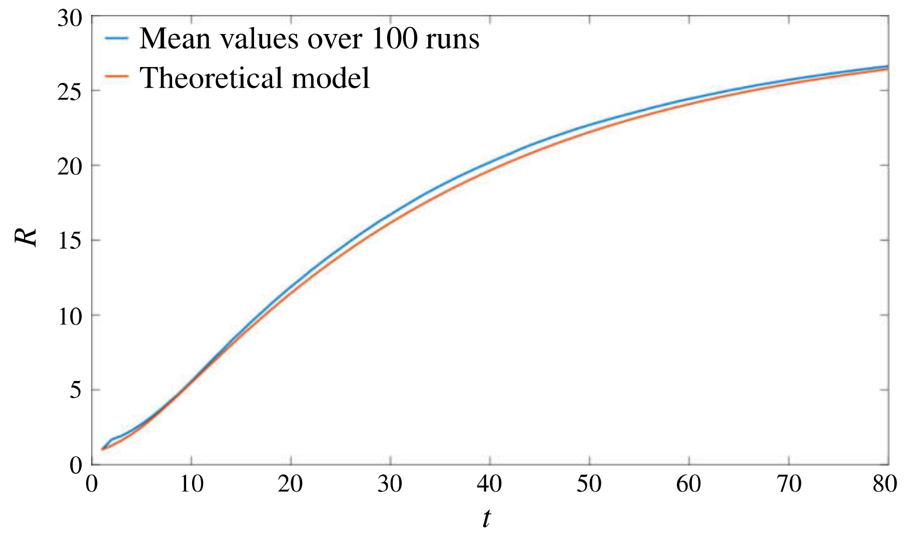


Figure 6. The blue line shows average values of the tumor radius variable over 100 CA model runs. Each point is obtained as the average of 100 tumor radius values obtained at that moment of time by 100 independent executions of the cellular automaton model. The red line shows the continuous model obtained from (21) as a particular case of the model defined by (16)–(19).

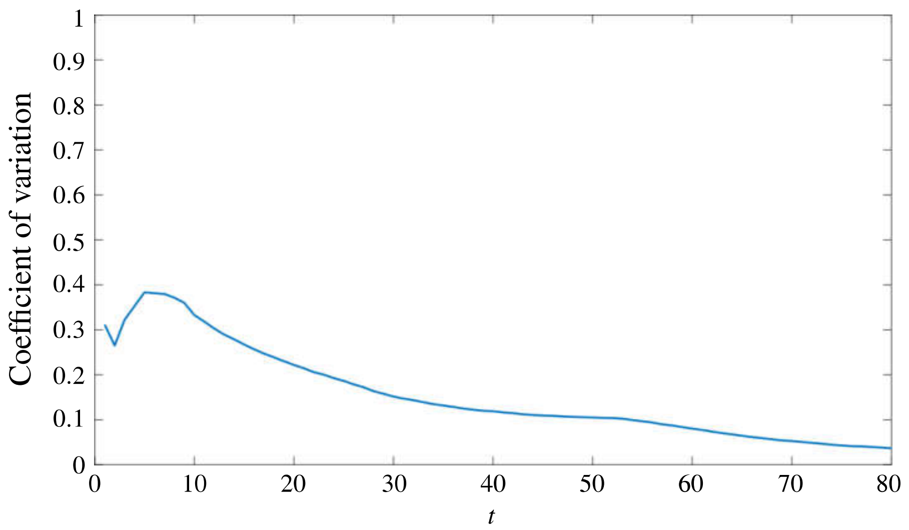


Figure 7. Values (in percent) of the coefficient of variation of the tumor radius variable for the CA model. Each point is obtained from 100 tumor radius values reached at that moment of time by 100 independent executions of the CA model.

20% line, indicating small variations of the tumor radius. Larger variations are observed only near the start of the executions. This shows a stable behavior of the computational model beyond its stochastic nature.

Conclusions

A new stochastic cellular automata model for the process of tumor growth is proposed. From a linear elasticity deterministic continuous tumor model described in [Ramírez-Torres et al. 2016], a neighborhood structure and stochastic automata rules are deduced. The results allow one to visualize the growth process described in the continuous model in a more realistic manner since tumors are neither regular nor perfectly circular. Moreover, stress influence in the growing of the tumor is taken into account at the time when the rules of the CA model are derived. The differences in the tumor radius between the two models are small and are actually due to their different nature. Validation tests confirmed that the CA model accurately captures the hypothesis of the described phenomena. The methodology exposed in this work can be applied to other continuous DE models in order to represent the growth processes in a nonidealized and nondeterministic way.

Acknowledgments

Ruben Interian has been sponsored by a CAPES scholarship. Ariel Ramírez-Torres thanks the postdoctoral funding of the Program of Postdoctoral Scholarships DGAPA from UNAM and professors Federico J. Sabina and Catherine García-Reimbert for their support. Work of Celso C. Ribeiro was partially supported by CNPq research grant 459575/2014-7 and by FAPERJ research grant E-26/201.198/2014. Aura Conci is partially supported by FAPERJ project SIADE2 and by CNPq project number 303240/2015-6.

References

[Anderson et al. 2009] A. R. A. Anderson, K. A. Rejniak, P. Gerlee, and V. Quaranta, "Microenvironment driven invasion: a multiscale multimodel investigation", *J. Math. Biol.* **58**:4–5 (2009), 579–624.

[Berto and Tagliabue 2012] F. Berto and J. Tagliabue, "Cellular automata", electronic reference, 2012, Available at http://plato.stanford.edu/entries/cellular-automata/.

[Boondirek et al. 2010] A. Boondirek, W. Triampo, and N. Nuttavut, "A review of cellular automata models of tumor growth", *Int. Math. Forum* **5**:61 (2010), 3023–3029.

[Deutsch and Dormann 2005] A. Deutsch and S. Dormann, *Cellular automaton modeling of biological pattern formation: characterization, applications, and analysis*, Birkhäuser, 2005.

[Dormann and Deutsch 2002] S. Dormann and A. Deutsch, "Modeling of self-organized avascular tumor growth with a hybrid cellular automaton", *In Silico Biol.* **2**:3 (2002), 393–406.

[Guinot 2002] V. Guinot, "Modelling using stochastic, finite state cellular automata: rule inference from continuum models", *Appl. Math. Model.* **26**:6 (2002), 701–714.

[Hoehme and Drasdo 2010] S. Hoehme and D. Drasdo, "A cell-based simulation software for multicellular systems", *Bioinformatics* **26**:20 (2010), 2641–2642.

[Hu and Ruan 2003] R. Hu and X. Ruan, "A simple cellular automaton model for tumor-immunity system", pp. 1031–1035 in *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing* (Changsha, China, 2003), vol. 2, Inst. Electrical and Electronics Engrs., 2003.

[Kansal et al. 2000a] A. Kansal, S. Torquato, E. A. Chiocca, and T. S. Deisboeck, "Emergence of a subpopulation in a computational model of tumor growth", *J. Theor. Biol.* **207**:3 (2000), 431–441.

[Kansal et al. 2000b] A. Kansal, S. Torquato, G. R. Harsh, IV, E. A. Chiocca, and T. S. Deisboeck, "Simulated brain tumor growth dynamics using a three-dimentional cellular automaton", *J. Theor. Biol.* **203**:4 (2000), 367–382.

[Merks and Glazier 2005] R. M. H. Merks and J. A. Glazier, "A cell-centered approach to developmental biology", *Physica A* **352**:1 (2005), 113–130.

[Ngwa and Agyingi 2012] M. Ngwa and E. Agyingi, "Effect of an external medium on tumor growth-induced stress", *IAENG Int. J. Appl. Math.* **42**:4 (2012), 229–236.

[Ramírez-Torres et al. 2016] A. Ramírez-Torres, F. Valdés-Ravelo, R. Rodríguez-Ramos, J. Bravo-Castillero, R. Guinovart-Díaz, and F. J. Sabina, "Modeling avascular tumor growth via linear elasticity", pp. 1739–1740 in *Contributions to the foundations of multidisciplinary research in mechanics* (Montreal, 2016), vol. 3, edited by J. M. Floryan, Proc. Int. Congress Theoretical and Applied Mechanics **24**, Int. Union Theoretical Applied Mechanics, 2016.

[Rejniak and Anderson 2011] K. A. Rejniak and A. R. A. Anderson, "Hybrid models of tumor growth", *Wiley Interdiscip. Rev. Syst. Biol. Med.* **3**:1 (2011), 115–125.

[Rejniak and McCawley 2010] K. A. Rejniak and L. J. McCawley, "Current trends in mathematical modeling of tumor-microenvironment interactions: a survey of tools and applications", *Exp. Biol. Med.* **235**:4 (2010), 411–423.

[Sokolnikoff 1956] I. S. Sokolnikoff, *Mathematical theory of elasticity*, 2nd ed., McGraw-Hill, 1956.

Received 4 Dec 2016. Revised 23 Apr 2017. Accepted 6 Jun 2017.

RUBEN INTERIAN: rubenus@yandex.ru
Institute of Computing, Fluminense Federal University, Niterói, Brazil

REINALDO RODRÍGUEZ-RAMOS: rerora2006@gmail.com
Mathematics and Computer Science Faculty, Havana University, Havana, Cuba

FERNANDO VALDÉS-RAVELO: fernandov@cim.sld.cu
Molecular Immunology Center, Havana, Cuba

ARIEL RAMÍREZ-TORRES: arielrt02@gmail.com
Institute for Research in Applied Mathematics and Systems, National Autonomous University of Mexico, Mexico City, Mexico

CELSO C. RIBEIRO: celso@ic.uff.br
Institute of Computing, Fluminense Federal University, Niterói, Brazil

AURA CONCI: conci.aura@gmail.com
Institute of Computing, Fluminense Federal University, Niterói, Brazil