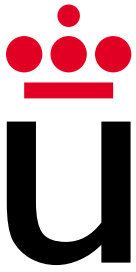


Universidad
Rey Juan Carlos

TESIS DOCTORAL

# *Design and Implementation of Metaheuristic Algorithms for Social Network Analysis*


**Autor:**

***Sergio Pérez Peló***

**Directores:**

***Abraham Duarte Muñoz***

***Jesús Sánchez-Oro Calvo***


**Programa de doctorado en Tecnologías de la Información y las Comunicaciones**

**Escuela Internacional de Doctorado**

**2022**

Esta Tesis Doctoral ha sido desarrollada bajo la financiación de los fondos asociados al proyecto PGC2018-095322-B-C22 de la Comunidad de Madrid, el proyecto PID2021-126605NB-I00 del Ministerio de Ciencia e Innovación, el proyecto PID2021-125709OA-C22 del Ministerio de Ciencia e Innovación y los Fondos Estructurales de la Unión Europea con referencia P2018/TCS-4566.

ID DOCUMENTO: CBXFN0psic
Verificación código: https://sede.urjc.es/verifica

El Dr. D. Abraham Duarte Muñoz, Profesor Catedrático de Universidad del Departamento de Ciencias de la Computación, Arquitectura de Computadores, Lenguajes y Sistemas Informáticos y Estadística e Investigación Operativa de la Universidad Rey Juan Carlos, y el Dr. D. Jesús Sánchez-Oro Calvo, Profesor Titular de universidad Departamento de Ciencias de la Computación, Arquitectura de Computadores, Lenguajes y Sistemas Informáticos y Estadística e Investigación Operativa de la Universidad Rey Juan Carlos, directores de la Tesis Doctoral Design and Implementation of Metaheuristic Algorithms for Social Network Analysis realizada por el doctorando D. Sergio Pérez Peló,

HACEN CONSTAR:

que esta Tesis Doctoral reúne los requisitos necesarios para su defensa y aprobación.

En Móstoles, septiembre de 2022,

Dr. D. Abraham Duarte Muñoz

Dr. D. Jesús Sánchez-Oro Calvo



| FIRMADO POR | FECHA FIRMA |
|---|---|
| SANCHEZ-ORO CALVO JESÚS | 30-09-2022 11:10:01 |
| DUARTE MUÑOZ ABRAHAM - DIRECTOR DE LA ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA | 30-09-2022 17:01:49 |

Documento firmado digitalmente - Universidad Rey Juan Carlos - C/Tulipan, s/n - 28933 Mostoles

*Se está siempre muy lejos de la solución de un problema hasta que uno realmente tiene la respuesta*

—Stephen Hawking

# Agradecimientos

Durante esta estapa de aprendizaje y crecimiento personal, que culmina en el presente manuscrito, han sido (y son) muchas las personas a las que tengo que agradecer en cierto modo su ayuda, apoyo o, simplemente, su presencia. Todas ellas han aportado su grano de arena, consciente o inconscientemente, para que yo haya podido llegar hasta aquí.

Gracias a Elena, mi mayor apoyo para todo en la vida, pero especialmente durante todo este proceso. Desde que nos conocimos siempre has estado ahí para animarme cuando lo necesité, aportarme tu punto de vista (siempre tan positivo) y corregirme con amabilidad cuando he estado equivocado. En estos casi cinco años hemos pasado por muchas situaciones diferentes (incluida una pandemia), pero siempre has mantenido esa esencia que te caracteriza, siendo el punto de luz en la oscuridad y el lugar al que aferrarse ante la incertidumbre. Gracias por cambiarme las gafas de mirar el mundo, convirtiéndome en una mejor versión de mí mismo. No puedo más que agradecerte el haberme aportado tanto pidiendo tan poco a cambio.

Gracias a mis padres, que siempre han apoyado cada una de mis decisiones, me han animado cuando he estado a punto de tirar la toalla y siempre han confiado en mí, a ciegas. No es fácil avanzar en el camino de la educación en líneas paralelas, pero creo que lo habéis conseguido con bastante éxito. A mi abuela, que aunque todavía hoy me siga preguntando a qué me dedico sin entenderlo demasiado bien, siempre aprovecha cualquier ocasión para presumir de nieto. No puedo estar más orgulloso de tu orgullo. A mis hermanos, que sin saberlo y sin quererlo me han aportado las capacidades que me han permitido afrontar con la responsabilidad y la paciencia necesarias todas las adversidades que se han ido presentando durante mi vida. Gracias a mi "otra" familia (otra entre comillas, porque para mí ya es la mía): Pedro, Carmen, Ani y David (y el resto de los Gavilán y los García), que siempre me han acogido como uno más, ayudándome en todo lo posible y sin ponerme jamás un impedimento para sentirme parte de algo tan importante como es la familia. No tengo más que agradecimiento hacia ellos.

Gracias también a todos mis amigos de toda la vida: Moha, Joseda, Julio, Pablo, Álvaro, Jesús, Nacho, Fran, Franco y todos los que no enumero aquí, pero saben que son igual de importantes. Cada uno de vosotros ha ayudado a forjar la persona que soy hoy, y no podía haber tenido mejores influencias. También a mis amigos que no son de toda la vida, pero a día de hoy podría decir que lo son sin ningún tipo de diferencia: Edu, Alba, Andrea, Marina, Irene y Patri. Durante estos cinco años hemos vivido mil experiencias, buenas y malas, que nos han unido lo suficiente como para no volver a separarnos, aunque estemos lejos los unos de los otros. Todos me habéis ayudado a descansar el cerebro después de jornadas interminables de no parar de pensar durante horas, e incluso algunos me habéis escuchado explicaros mis problemas de optimzación pidiéndoos ideas, aún sin saber de optimización y aún a riesgo de aburriros soberanamente. Ese es el tipo de amistad que no tiene precio.

En el ámbito laboral, el primer agradecimiento debe ir sin duda hacia Jesús y Abraham. Desde el día en que Jesús "me engañó" para comenzar esta aventura que es la carrera universitaria siempre he podido contar con vosotros para todo, incluso cuando otras tareas mucho más importantes requerían vuestra atención habéis sacado un hueco para mí. Me siento muy orgulloso de tener unos mentores de tal nivel, es un verdadero privilegio trabajar con vosotros. Gracias por enseñarme de forma tan didáctica y positiva desde que fuisteis mis profesores en la carrera hasta hoy, que aún sigo aprendiendo día a día de vosotros. Esta tesis es tan vuestra como mía, espero estar a la altura de vuestras espectativas y de vuestra calidad docente e investigadora. Gracias por el privilegio de aprender con vosotros, nunca podré pagarlo (ni siquiera en cervezas). Gracias también a Eduardo y Chema, que aunque no han estado implicados directamente en mi tesis, siempre me han dado buenos consejos sobre cómo afrontar este trabajo con éxito, permitiéndome mejorar día a día. También a Antonio, que se unió a nosotros cuando yo llevaba aproximadamente la mitad del camino recorrido, pero que sin dudarlo ni un instante se unió a mí en la investigación en la que se centra esta tesis, depositando su confianza en mí sin apenas conocerme.

El camino no es tan divertido si se hace en soledad, por eso tampoco puedo dejar de agradecer a todos mis compañeros becarios que me han acompañado durante este periodo. A Juan David, que fue el primero en acogerme en este lado de la trinchera (nunca olvidaré tu boda). A Raúl, el tipo más listo que he conocido y el mejor compañero de estancia que podía haber tenido. Tendremos que volver a Bruselas a visitar a *Carlota* y sus *tartines*. A Cavero, Yuste e Isaac, no recuerdo otra cosa que no sean risas con vosotros. Gracias por aportarme siempre un punto de vista diferente, ayudarme con los problemas que hoy forman parte de esta tesis, y ser parte de momentos tan importantes como la victoria

en la National Cyberleague. Espero que sigamos compartiendo momentos dentro y fuera de los despachos siempre. A Ale, a la que considero una amiga de las de verdad (aunque te empeñes en decirme que no somos amigos y siempre estés pinchándome). Siempre dispuesta a decirte lo bueno y lo malo, sin tapujos, eres la amiga crítica que todo el mundo necesita en su vida. A Enrique, el compañero de carrera, máster, empresa y doctorado que cualquiera querría tener. No has fallado ni una sola vez de las que te he necesitado, quizá la persona que más me ha apoyado y me ha enseñado del mundo de la informática desde el día cero. No olvidaré nunca esos años de 12 horas seguidas en la universidad, las prácticas, los exámenes y las horas de estudio juntos. Si hoy estoy donde estoy, es en gran parte gracias a tu inestimable ayuda.

Al resto de compañeros del grupo GRAFO: Nico, Alf y Alberto. Cada uno forma parte de los engranajes de la maquinaria del grupo que me ha acogido desde el principio, y no concibo otra familia laboral con mejores componentes. Merecen también una mención especial todos los compañeros de la Universidad Rey Juan Carlos y, en concreto, del departamento de Ciencias de la Computación (y todos los demás que se agrupan bajo las siglas CALE) que, en algún momento u otro, me han ayudado con mis gestiones o con los problemas que me han podido surgir: Almudena Sierra, Antonio Sanz, Juanjo Pantrigo, David Concha, Belén Vela, Alberto Sánchez, Marta Beltrán, Carlos Cuesta y todos aquellos buenos compañeros y docentes que he tenido durante todos estos años.

Gracias también a Rafa Martí, que nos abrió el camino a todos. Siempre recordaré sus charlas contándonos como siguió la pista a un geólogo llamado Kuby para acabar trabajando en el problema de la Máxima Diversidad que tanto juego ha dado. También nuestro interés común por la música, aún tenemos pendiente una sesión de estudio conjunta. A Anna Gavara y Ana Dolores, las mejores colaboradoras que he podido tener. Muchas gracias por extender la amistad que compartís con Jesús hacia mí desde el primer día. Recordaré siempre con mucho cariño mis viajes a Valencia y Sevilla, que espero seguir haciendo durante mucho tiempo.

No me olvido de todas las personas que he conocido durante mi estancia en Bruselas: Carlos, Gonzalo, Carmen, Clara, Marieta, Laura, Gloria, María, Claudia y Josema. Pero sobre todo, tengo que agradecer a Fede hacer de nexo de unión entre vosotros y yo. Mil gracias por hacer de la experiencia una vivencia única, de la que me llevo muchos recuerdos, lugares... pero, sobre todo, amigos. Volveremos pronto a plux.

Gracias a Thomas Stützle por su hospitalidad, generosidad y atención durante los meses que estuvimos en Bruselas, ha sido un placer trabajar con un

investigador de su renombre y calidad. Una experiencia fructuosa en todos los sentidos. Mi agradecimiento también a todo el laboratorio de IRIDIA. El aprendizaje a través de los seminarios, la acogida de todos vosotros y eventos como las *IRIDIAlympics* son cosas que nunca olvidaré.

Mencionar también a todos esos alumnos que ahora son amigos: Diego, Andrea, Álex, Ismael, y todos a los que haya podido aportar un poquito de mí en su formación. Mucho éxito en todo lo que os propongáis, tenéis un gran futuro por delante.

A todas las personas que, de una forma u otra, han formado parte de mi camino hasta aquí. Esta tesis es el final de una etapa, pero el principio de otra que me depara muchos retos, éxitos y fracasos. Espero contar con todos vosotros como he podido hacerlo hasta ahora.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**ACO** Ant Colony Optimization.

**CDP** Community Detection Problem.
**CL** Candidate List.
**COP** Combinatorial Optimization Problem.
**CPM** Clique-Percolation-Method.

**DCDP** Dynammic Community Detection Problem.

**EADP** Extended Adaptive Density Peaks.
**EC** Evolutionary Computation.

**GRASP** Greedy Randomized Search Procedure.

**IG** Iterated Greedy.

**MOCDP** Multi-objective Community Detection Problem.
**MODCDP** Multi-objective Dynamic Community Detection Problem.

**NRA** Negative Ratio Association.

**OCDP** Overlapping Community Detection Problem.

**PR** Path Relinking.
**PSO** Particle Swarm Optimization.

**RC** Ratio Cut.
**RCL** Restricted Candidate List.

**SCC** Strong Connected Components.
**SNA** Social Network Analysis.

**TSP** Travelling Salesman Problem.

**VNS** Variable Neighborhood Search.

# Chapter 1

# Abstract

In a society where the immediacy is becoming more and more established, there is a need to obtain solutions to real-world problems quickly and accurately. With respect to precision, or more specifically optimality, the scientific discipline that deals with this issue is optimization. This knowledge area can be seen as a meeting point between various disciplines, such as operations research, statistics, computer science or artificial intelligence. There are a lot of interesting real-life problems that can be approached from the optimization point of view: calculating the fastest route to go from home to work, finding the best way to place the maximum number of containers in a ship to send products around the world, or knowing the weakest point of a computer network in order to dedicate more resources to protect it. All these problems can be solved by two different approaches: exact algorithms or approximated algorithms.

The main problem of exact algorithms is that, when the space of solutions to be explored is too large, the computation time required to provide an optimal solution to the problem is unacceptable. In this context, approximated algorithms emerge as an alternative, with the main disadvantage that they do not guarantee to reach an overall optimal solution. However, if the adequate technique is selected, a high-quality solution can be assured. This is the case of heuristic and metaheuristic algorithms.

The Community Detection Problem (CDP) is an $\mathcal{NP}$-hard problem that belongs to the Social Network Analysis (SNA) family of problems. The main objective in CDP is to group the users within a social network depending on their characteristics, their relations and other properties of the network itself. It is said that a good solution for the CDP is characterized by a good community structure. Community structure is considered good when the resulting groups

contain nodes that are highly connected among them and sparsely connected with nodes in other groups. There are different variants of the same problem in which different constraints are considered. For example, the number of communities that a solution contains is fixed *a priori*, or nodes can be assigned to different groups simultaneously. The CDP can also be faced optimizing different objective functions simultaneously, and taking into account single or multiple objectives. Despite all of the above, the ultimate goal is always the same: to obtain solutions with a good community structure.

In this Thesis different heuristic and metaheuristic algorithms are proposed to solve the most extended variants of CDP. The problems have been tackled by considering a wide variety of methodologies, such as Variable Neighborhood Search (VNS), Iterated Greedy (IG) or Scatter Search (SS). Each proposal has been evaluated against both synthetic and real-world networks to check their utility and application in these contexts. Obtained results overcome the state of the art proposals in all studied variants of CDP.

# Part I

# PhD Dissertation

# Contents

# Chapter 2

# Introduction

*This Doctoral Thesis belongs to the field of optimization. More specifically, this document is focused on solving problems included in the Social Network Analysis (SNA) from the optimization point of view. In particular, the solved problems belong to a family of problems that is commonly known as Community Detection Problem (CDP). In this chapter, the Thesis context is presented, introducing the optimization area and the tackled problems characteristics. Finally, the main hypothesis of the work and the objectives derived from it are presented.*

## 2.1 Optimization

In the last decades, the need to obtain solutions to real-world problems has drastically increased. Every day new problems appear since mankind is evolving more and more rapidly, and with it the problems we must face. In this context, optimization problems appear as an important tool to solve these new real-life problems, aiming to find feasible solutions to them. Optimization can be seen as the confluence of several disciplines such as Operational Research, Applied Mathematics, or Computer Science [3]. It tries to solve problems from different scientific research areas since the time of Heron of Alexandria, who is credited with one of the first optimization problems in history [4].

Generally, an optimization problem could be described as a concrete task that has two general characteristics. The first one is that there are usually many solutions to this problem. The second one is that there is a clear method for comparing these solutions to each other [5].

In mathematical terms, an optimization problem can be defined by an objective function that must be minimized (or maximized), being subject to a set

of constraints that must be satisfied in order to obtain a feasible solution. More formally, an optimization problem (OP) can be described as follows:

$$OP = \begin{cases} Minimize\ f(\sigma) \\ subject\ to\ \sigma \in \phi \end{cases} \tag{2.1}$$

where $\sigma$ is a solution to the incumbent problem, $f(\sigma)$ is the objective function that will be optimized and $\phi$ is the set of feasible solutions. When tackling an optimization problem, the solution that is tried to be found is the one known as the optimal solution. A feasible solution is an optimal solution if and only if it provides the minimum or maximum value for the objective function under evaluation (depending on whether a minimization or maximization problem is being addressed, respectively).

Scientific community has a widely interest in optimization problems, since solving problems in this area implies solving them also in many other areas of knowledge. Developing methodologies and algorithms capable to solve these kind of problems derives in the generation of knowledge that is useful to solve real-world difficulties, such as decide where to locate a new hospital, what is the most ecological route between two points or how a large amount of orders must be shipped to maximize the profit in the shortest time possible.

If more granularity to the definition of an optimization problem is added, it can be defined as a set of decision variables that must be assigned to a certain value that provides the best final value to the objective function in which they are included, always satisfying the given restrictions. In terms of the type of variables involved, the objective function to be optimized and the constraints defined, optimization problems can be divided into different families. More specifically, we can distinguish among three well-known categories: Non-linear Programming (or Global Optimization), variables that compose the problem take real values and neither objective function nor constraints are restricted to be linear; Linear Programming, where variables take real values and both objective function and constraints must be linear; Integer programming, which differs from linear programming in that the variables take integer values. However, this categorization includes only those problems where constraints and objective function are clearly known, but it is not the habitual situation in real-life problems. Indeed, a lot of daily problems are difficult to them formulate mathematically due to their high combinatorial character. Problems like Travelling Salesman Problem (TSP) [6, 7], or Knapsack problem [8] belong to this family. In this work, all tackled problems are Combinatorial Optimization Problems (COP) where solutions are modeled

with integer numbers [9, 10]. Combinatorial Optimization Problems have a dimensionality $n$ associated that is useful to represent a single solution $\sigma$, where $\sigma = (x_1, x_2, ..., x_n)$. Having defined a solution, a value $v_i$ is assigned to each variable $x_i$ of the problem, with $v_i$ belonging to the domain $D_i, i \in [1, n]$. This ensures that the solution addresses all the constraints of the problem. Mathematically,

$$COP = \begin{cases} \sigma = \{x_1, ..., x_n\} \\ D_1, ..., D_n \\ f : D_1 \times D_2 \times ... \times D_n \rightarrow \mathbb{R}^+ \\ \sigma = \{\{(x_1, v_1), ..., (x_n, v_n)\} | v_i \in D_i\} \end{cases} \tag{2.2}$$

Another way to classify optimization problems is attending to how difficult is for a machine (typically a computer) to solve the problem. In other words, the classification is based on its computational complexity [11]. Given this classification, algorithms can be also classified relating to the time that it takes for them to solve a problem. This time can be sized using the Bachmann-Landau notation [12, 13], also known as Big-O notation ($O$-notation), which defines the time required by an algorithm to solve certain problem in the worst case. Using this notation, there are certain problems that can be solved by an existing algorithm in polynomial time. It means that there exists a direct relation between the algorithm's input and the maximum time that it needs to solve a problem, and this relation can be expressed using a polynomial function. The required time by an algorithm to solve the problem in the worst case is denoted as $\mathcal{O}(n^t)$, being $t$ the maximum exponent in the polynomial function. Problems that meet this characteristic are said to belong to the family of $\mathcal{P}$ problems. Examples of $\mathcal{P}$ problems can be the Strong Connected Components (SCC) problem, solved by Tarjan [14] and Kosaraju-Sharir [15], the Topological Sorting problem, solved by Kahn [16], or the Shortest Path Problem, solved by Dijkstra[17].

Although the set of $\mathcal{P}$ problems is quite large, the problems for which no polynomial solution is known are numerous. In fact, most of today's real-world problems belongs to the $\mathcal{NP}$ problems class. Although there are not known solutions for them, it is possible to probe if an obtained value conforms a solution for the problem under study in polynomial time. Given this definition, all $\mathcal{P}$ problems are also $\mathcal{NP}$ problems, because probing that a given solution is a solution for the problem requires polynomial time too.

If real-world problems are analyzed, it is easy to find a large amount of them that can be categorized as $\mathcal{NP}$ problems. In view of this circumstance, the need to develop efficient methods to solve them emerges. Furthermore, some problems require of high-quality solutions in short computational times, even if optimality

cannot be guaranteed. In this context, heuristic methods [18] appear as a good alternative to solve problems in a fast way, to the detriment of the guarantee of obtaining an optimal solution. This is the main difference between exact and heuristic methods. The former are developed to find the optimal solution for a given problem [10, 19], while the latter does not give this guarantee.

On the one hand, the main difficulty related to exact algorithms is that there are a large quantity of real-life problems that they cannot solve, either because of the complexity of the problem itself or because an exact algorithm to solve it is not yet known. On the other hand, the main problem faced by heuristic algorithms is that they easily get stuck in a local optimum of the solution space. Under this circumstance, metaheuristics emerge as a new kind of algorithms that guides the heuristics during their search, with the aim of escaping from local optima and improving the solutions found. Metaheuristic concept has been introduced by Glover in 1986 [20] and was defined as an "strategy that guides and modifies other heuristics to explore solutions beyond local optimality". In the same definition, Glover explains that "the heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule".

Attending to this definition, metaheuristics are considered as a set of approximated algorithms that guides the heuristic procedure exploration in a smart way, combining intensification (exploitation) and diversification (exploration) of the search space of the problem under solution. By the intensification, a limited but promising region of the search space is explored, looking for improvements in the incumbent solution. This procedure is traditionally associated with local search procedures. Regarding exploration, a large region of the search space is explored, with the aim of increase the diversity among explored solutions. Metaheuristics have been successfully applied in a large number of optimization problems, reaching high-quality solutions in a reasonable computational time.

In this PhD Thesis, some of the most widely used metaheuristics have been explored to solve optimization problems, providing high quality solutions for real-life problems. Some of them are *Greedy Randomized Search Procedure* (GRASP) [21, 22], *Variable Neighborhood Search* [23], *Path Relinking* [24, 25] and *Scatter Search* [26]. Concretely, these metaheuristics are used in order to provide high quality solutions to an optimization problem that belong to the Social Networks Analysis (SNA) topic: the Community Detection Problem (CDP).

## 2.2 Optimization problems in Social Network Analysis

The evolution of social networks in the last decades has aroused the interest of scientists from different fields, from psychology to computer science. Millions of people are constantly sharing all their personal and professional information on different social networks [27]. In addition, social networks have become one of the most widely used sources of information, mainly due to their ability to provide the user with real-time content. Social networks are not only a new form of communication, but also a powerful tool that can be used to gather information related to relevant questions, for example: which is the favorite political party for the upcoming elections, what are the most talked about movies of the past year, which is the best rated restaurant in a given area, etc. Extracting relevant information from social networks is a topic of interest, mainly due to the enormous amount of potential data available. However, traditional network analysis techniques are becoming obsolete due to the exponential growth of social networks, in terms of the number of active users and the relations among them. Social Network Analysis (SNA) has become one of the most popular and challenging tasks in data science [28]. One of the most addressed problems in social networks is the analysis of the relevance of users in a given social network [29]. The relevance of a user is commonly related to the number of followers or friends the user has in a given social network. However, this concept can be extended, as a user can be relevant not only if they is connected to a large number of users, but also if they is connected to relevant users. Several metrics have been proposed to analyze the relevance of a user in a social network, with PageRank emerging as one of the most widely used [30]. Moreover, it is interesting to know which users will be the most relevant before they become influential [31]. Finally, in the field of marketing analytics, there is a special interest in generating the profile of a user given a set of tweets written by that user [32].

Assessing the relevance of a user has become a more complex problem that consists of detecting specific users (often referred to as influencers) with certain attributes that may be personal (credibility or enthusiasm) or related to their social networks (connectivity or centrality).

These attributes allow them to influence a large number of users, either directly or indirectly [33]. Another important issue related to the influence of people on other users is sentiment analysis in social networks. This analysis focuses on discovering what people think about a given topic by analyzing the information they post on social networks. A complete survey on sentiment analysis techniques can be found in [34].

The problems described above deal only with individual users. However, there are also some problems related to the network structure, dedicated to finding specific attributes and properties that can help to infer additional information from the whole social network. In this context, community detection emerges as one of the most studied problems, which is the main topic that concerns this Thesis.

## 2.3 Hypothesis and objectives

Once the problem to be solved has been identified, the next point to be addressed during the development of a research project is the formulation of an initial hypothesis. This hypothesis is a tentative proposal that seeks to formulate a solution to the problem posed. The hypothesis will represent a fundamental element in the research process, since it will serve as a guide.

The hypothesis proposed for the development of this Doctoral Thesis can be summarized in the following terms: the Community Detection Problem in social networks is a task with practical interest in different scientific disciplines, but with a high computational cost. Therefore, it is interesting to develop algorithms that are able to solve it efficiently, obtaining optimal solutions if possible or, at least, of high quality, in a reasonable amount of time. For this reason, heuristic algorithms become relevant to solve this type of problems. In recent years, bioinspired heuristics have been proposed to avoid the computational requirements of exact implementations. In the area of bioinspired computation, evolutionary approaches are the most popular algorithms. It is important to highlight the review performed by Pizzuti in [35] about Evolutionary Computation (EC) techniques to detect communities in networks. An interesting work about EC is the work published by Said et. al [36], where authors designed a clustering coefficient-based genetic algorithm able to detect cohesive groups from dense graphs and also, communities in sparse networks. Other relevant work is [37] that presents a genetic algorithm that uses a multi-individual ensemble learning-based crossover function. The algorithm is improved with a local search strategy to speed up its convergence.

Other well-known bioinspired algorithms are the ones belonging to swarm intelligence. In this new group, the most popular algorithms are Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). These two algorithms are inspired by the social behaviour of birds within a flock, and the behaviour of ants seeking a path from the nest to the source of food, respectively. PSO has been successfully used for CDP in [38], where a discrete PSO algorithm is used to extract the communities in large-scale social networks by optimizing the modularity. Regarding ACO algorithm, this algorithm has been used to extract high-quality communities in Ego Networks [39].

However, this family of algorithms has the disadvantage that they do not leverage specific information obtained from the network. In this Thesis, the development of algorithms based on traditional metaheuristics, performing an in-deep study of the networks under evaluation opens a new perspective for future research for community detection problems.

The proposed heuristic algorithms will make use of metaheuristic techniques, which have been shown to be effective procedures when dealing with optimization problems. In particular, population-based metaheuristics constitute a subset of this type of techniques, characterized by taking into consideration more than one solution simultaneously and providing combination mechanisms among them. Techniques such as Scatter Search are a clear example of this type of metaheuristics. On the other hand, the so-called trajectory-based metaheuristics start from an initial solution and are capable of generating a trajectory in the solution space. Examples of this type of metaheuristics are GRASP (Greedy Randomized Adaptive Search Procedure) or Path Relinking. The main objective of the Doctoral Thesis is to develop algorithms that solve several optimization problems related to the analysis of social networks and the detection of communities in them, approaching them from a heuristic point of view, both from the mono-objective and multi-objective perspective.

To achieve the main objective described above, it is necessary to cover the following partial objectives:

- **To study the state of the art of the problem.** An exhaustive literature review must be performed to analyze both the proposed algorithms and the instances to be evaluated.
- **To design and develop a metaheuristic algorithm for solving the problem.** At this point, one or more metaheuristic algorithms are developed, modeling the initial constructive procedure (typically an heuristic), the improvement method (normally a local search) and the metaheuristic framework that guides the heuristic.
- **To validate the heuristic algorithm.** Once the proposed algorithm is designed, it must be implemented and validated. This validation process includes to check if it is generating feasible solutions and, if it is, to analyze the relevance of each stage of the algorithm and its contribution to the final solution.
- **To experimentally compare the proposed algorithm with the state of the art algorithms.** In this stage, proposal is compared against the state of the art methods using a reference dataset. Through this experimentation,

the strengths and weaknesses of the developed algorithm and its contribution to the literature can be studied. In this comparison, it will be necessary to use statistical tests to confirm that the proposed algorithm is a relevant scientific contribution to the area.

- **Submit the partial results of the research work to review processes by independent institutions.** The results obtained during the whole process are sent to conferences and journals of high impact in the research area for their possible publication.

## 2.4 Memory structure

This document summarizes the research that has been conducted and is organized as follows:

- In **Part I** , the developed research is shown, describing the addressed problem, as well as the followed methodology and the obtained results. Finally, the conclusions derived from the work are exposed.
  - In **Chapter 2**, optimization research area, Combinatorial Optimization Problems, heuristics and metaheuristics are described. Hypothesis and objectives for this work are also defined.
  - In **Chapter 3**, the Community Detection Problem (CDP) is presented, defining it in a detailed way, explaining each one of its variants.
  - In **Chapter 4**, the followed methodology applied to the previously described problem is explained. The main metaheuristic and all the other techniques that lead it to achieve the obtained results are exposed.
  - In **Chapter 5**, the obtained results for each variant of the problem and the main contributions made are analyzed and discussed.
  - In **Chapter 6**, the conclusions derived from the research and possible future works are presented.
- In **Part II** the published articles associated with this Thesis are summarized and gathered, together with information about the journal in which they are published.
- In **Part III** additional articles that are not directly related with CDP, but that are the results of using the contributions derived from this Thesis are listed and replicated, together with information about the journal in which they are published.

- Finally, **Part IV** contains the abstract of the dissertation and its conclusions in Spanish language.

# Chapter 3

# Community Detection Problem

*In this chapter the classical Community Detection Problem is introduced. Then, the main exact and heuristic algorithms, as well as the most extended metrics used in the literature to solve them are described. Finally, the variants of the problem studied in this dissertation are exposed and defined.*

## 3.1 Community Detection Problem

In recent years, the importance of social networks has experimented an exponential growth. Every day new users and connections among them appear in the context of social networks, generating a large amount of information that can be analyzed in different ways. Many research fields can be derived from social networks, that are joint in the Social Network Analysis area. In this field, many different sub-areas can be found, such as sentiment analysis, radicalization detection or influence analysis. This Thesis is focused on one of these sub-areas: the community detection problems. Most of the social networks present a common characteristic known as community structure [40]. It is said that networks with a good community structure can be divided into groups in such a way that connections between users in the same group are dense, while connections between users in different groups are sparse. The connections between users can represent different characteristics depending on the nature of the social network and the profile of the users, i.e., from professional relationships to friendships or common hobbies. The community detection tasks are dedicated to find and analyze these groups with the main objective of better understanding and visualizing the structure of the network and the relationships between its users.

Running community detection algorithms in modern social networks is a

computationally intensive task, mainly due to their continuous growth. Moreover, since these networks are constantly changing (new friendships, mentions to users, viral information, etc.), it is interesting to perform community detection in the shortest possible computational time, producing information in real time. These characteristics make exact methods unsuitable for the current size of social networks, requiring heuristic algorithms to speed up the process without losing quality. Recent works have addressed the community detection problem from a non-exact perspective to generate high quality solutions in small computational time [39]. Several studies are devoted to reduce the computational effort to detect communities in social networks [41, 42]. When comparing traditional algorithms in large modern social networks, it can be observed that some of the algorithms require more than 40 000 seconds for networks with approximately 10 000 nodes, and cannot provide a solution after 24 hours of computation for networks with more than 50 000 nodes. The growth of social networks makes more difficult their representation and understanding. Communities in a social network generally summarize the entire network, but reducing its size and, thus, making it easier to analyze. In addition, community detection in social networks has several practical applications. Recommendation systems analyze the data of similar users to suggest content that may be of interest to them. To find similar users in a network, we can simply perform community detection across the network [43], improving the results of the recommendation system. Communities in social networks also identify people with similar interests, allowing us to evaluate the popularity of a political party [44], or even detect radicalism in social networks [45]. While there are several community detection algorithms that have been proposed with the objective of identifying similar users in networks, most of the available procedures have been designed to optimize a specific objective function, making it difficult to adapt the algorithm to a different one. However, the constant evolution of this area results in a continuous proposal of new metrics that better represent the community structure of a given network. In this Thesis, efficient and versatile algorithms are proposed that can be easily adapted to different optimization metrics.

In mathematical terms, a social network can be represented as a graph $G = (V, E)$, where $V$ is the set of vertices (users) and $E$ is the set of edges (connections among users). It is worth to mention that graphs representing social networks can be directed or undirected, depending on the nature of the social network that is being analyzed. In social networks like Twitter, where a follower-followed relationship between two users is given, a directed graph emerges as the natural representation of the network. However, in networks like Facebook, where a friendship relation occurs, an undirected graph is a better option, given that if user `A` is a friend of user `B`, then `B` is also a friend of `A`. Given this definition, the main objective of Community Detection Problems (CDP) is to find subgraphs

(communities) with a good community structure. As it was aforementioned, a good community structure is reached when the found communities have nodes densely connected among them and sparsely connected to nodes in others subgraphs. The ideal situation is the one in which each community conforms a different connected component in the whole graph, but this is a circumstance that rarely occurs in real-world social networks. However, there are different approaches that use this idea as starting point, trying to solve the CDP by identifying cliques in the graph that represents the network [46, 47, 48]. Despite of this, there are several approaches based on completely different ideas, such as obtaining the centrality metrics of each node and joining in the same community those that are close to a central node [49], or taking into account the number of edges that connect different nodes to generate the different communities [50]. In this sense, Section 3.2 presents a first approach to the community detection.

Attending to the evaluation of the quality of a given solution for the CDP, there are different metrics that have been designed with the aim of measuring how good the community structure of the reached solution is. In Community Detection Problems, two or more different metrics must be used, since the same metric cannot be applied to generate high-quality solutions and to evaluate them. On the one hand, one (or more) must be selected as objective function. On the other hand, different metrics must be considered for evaluating the performance of the algorithms, with the aim of providing a fair comparison among algorithms. There are many metrics that focuses in the same goal: evaluating the community structure of a solution. Section 3.1.1 compiles and describes some of them.

### 3.1.1 Metrics used in the Community Detection Problem

As it has been stated, there is no consensus on what is the best metric to be optimized when facing a Community Detection Problem. There is a general agreement on what shaped communities should be like, but not on exactly how to measure that quality. In this scenario, many authors working in the Community Detection Problem have proposed new metrics, all of them pursuing the same goal: to establish a metric that, when optimized, allows to generate communities with individuals densely connected to each other and sparsely connected to individuals belonging to other communities.

The most widely used metric for both evaluation and optimization is the modularity. This metric was formally proposed in [51] and slightly reduced in [52]. Modularity evaluates the quality of a network structure based on the number of edges that are included in a community minus the expected number of them in a network where these edges are randomly placed. Mathematically, given a graph $G$ and a solution $S$ for this graph, modularity $Q(G, S)$ can be described as:

$$Q(G,S) = \sum_{k=1}^{\mathcal{K}} \left( \frac{|E_{C_k\leftarrow}|}{|E|} - \left( \frac{sd_{C_k}}{2 \cdot |E|} \right)^2 \right) \tag{3.1}$$

where $\mathcal{K}$ is the number of communities in a solution, $|E|$ is the number of edges in the whole graph $G$ that identifies the network under evaluation, $E_{C_k\leftarrow}$ represents the set of intra-community edges in community $k$ (it is, edges that connect nodes in the same community), and $sd_{C_k}$ represents the sum of degrees of all nodes belonging to community $k$. Following this definition, it seems clear that a high modularity value is obtained when nodes in the same community are highly connected among them and slightly connected with nodes in others communities. Modularity value is always in the $[-1, 1]$ range. A value of 1 is obtained when all nodes are in the same community and the network represents a complete graph, it is, each node is connected to all other nodes.

Another good metric used in community detection problems is the conductance [53, 54, 55]. This metric tries to size how well-connected are the nodes that belong to a community, considering that good community structure is reached when splitting a community in more has no sense. Given a network $G$, a solution $S$, and a specific community $C_k$, its conductance, $C_o(C_k, S, G)$, is calculated as the number of inter-community edges (it is, the number of edges that connects one node assigned to community $k$ with a node assigned to a different community) divided by the minimum between the number of edges in one community (intra-community and inter-community edges) and the number of edges that connect nodes that do not belong to the community $C_k$. More formally,

$$C_o(C_k, S, G) = \frac{|E_{C_k\rightarrow}|}{\min\{|E_{C_k\leftarrow} \cup E_{C_k\rightarrow}|, |E \cap (E_{C_k\leftarrow} \cup E_{C_k\rightarrow})|\}} \tag{3.2}$$

Having given this definition, the conductance of a complete solution $C_o(S, G)$ is evaluated as the average conductance for all the communities in the graph. In mathematical terms,

$$C_o(S, G) = \frac{\sum_{k=1}^{\mathcal{K}} C_o(C_k, S, G)}{\mathcal{K}} \tag{3.3}$$

where $\mathcal{K}$ is the number of communities in the incumbent solution $S$. With these definitions, it is clear that a lower value of conductance represents a better connectivity among members of a community, so the higher conductance, the better.

Community Score [56] is another good objective function to be established when solving the CDP. This metric sizes the number of edges belonging to a

single community with respect to the size of the community, evaluating how dense is the subgraph generated by the community under evaluation. More formally, community score can be defined as follows:

$$CS(S) = \sum_{k=1}^{\mathcal{K}} \left( \frac{2 \cdot |\{v \in (V_k \setminus \{u\}) : (u,v) \in E_{C_k}\}|}{|C_k|} \right)^2, \forall u \in C_k \tag{3.4}$$

where $E_{C_k}$ represents the set of edges in $C_k$. As it can be derived, a high value of $CS$ implies a high degree of connectivity in the detected communities. Therefore, a high value of community score indicates that generated groups are considered good communities, so when it is used as objective function a maximization problem is being tackled.

Average Out-Degree function metric [57] sizes the number of edges in a single community that connect nodes belonging to different communities. In this case, the metric evaluates if the found communities in a given solutions have a large amount of inter-cluster edges, which is a bad feature in the context of community detection. Mathematically,

$$AVG\_ODF(S) = \sum_{k=1}^{\mathcal{K}} \left( \frac{1}{|C_k|} \sum_{u \in C_k} \frac{E_{C_k \to (u)}}{d(u)} \right) \tag{3.5}$$

where $E_{C_k \to (u)}$ is the number of inter-cluster edges in community $C_k$ with $u$ as an endpoint, and $d(u)$ is the degree of node $u$. Minimizing this metric, well interconnected groups are reached, so a lower value of $AVG\_ODF$ indicates a good community structure.

Ratio Association (RA) [58] is another used metric that measures the percentage of edges connecting nodes in the same community that exists with respect to the size of the community under evaluation. In mathematical terms,

$$RA(C_k) = \frac{|E_{C_k \leftarrow}|}{|C_k|} \tag{3.6}$$

Maximizing this metric, the number of edges that relates users in the same community is maximized too, so higher values of RA indicates better community structure in the obtained solutions.

Ratio Cut (RC) [59] is the opposite of RA, given that it evaluates the percentage of edges connecting nodes in different communities for each community present in certain solution with respect to its size.

$$RC(C_k) = \frac{|E_{C_k\rightarrow}|}{|C_k|} \tag{3.7}$$

Regarding at this definition, it is clear to see that lower values of Ratio Cut imply less connections among users in different groups, so the lower the RC value, the better.

Similarly, the RA and RC of a complete solution $S$ are defined as:

$$RA(S) = \sum_{k=1}^{\mathcal{K}} RA(C_k) \qquad RC(S) = \sum_{k=1}^{\mathcal{K}} RC(C_k) \tag{3.8}$$

There are more metrics that are also suitable for solving the CDP. As it has been aforementioned, different authors have proposed their own metrics and objective functions, trying to approach the problem from a different point of view. However, all the metrics found in the state of the art are based on the same principles as those already mentioned: to obtain communities densely connected to each other and sparsely connected to other communities. Once the main metrics used in the CDP problem have been summarized, different variants of CDP are presented.

## 3.2 The $\alpha$-separator problem

The $\alpha$-separator problem is the first problem considered in the framework of this Doctoral Thesis. Given a network, the main objective in this problem is to find the minimum set of nodes whose removal divides the network into connected components with a size less than or equal to $\lceil \alpha \cdot n \rceil$, where $n$ is the number of elements in the network. If the network is modeled as a graph $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of edges, the objective is to divide the graph in $p$ subgraphs ($M_1, ..., M_p$, where $p \geq 2$) by removing a subset $L$ of vertices ($L \subseteq V$). For the sake of clarity, Figure 3.1 illustrates the problem with an example network and a feasible solution for the $\alpha$-separator problem.

In this figure, two feasible solutions are represented. The first one, Figure 3.1b, shows a solution with a separator conformed by nodes `B`, `C`, `F`, and `H`, whose size is 4, while the second on, Figure 3.1c, represents a better solution, given that the separator includes only two nodes: `A` and `B`. The $\alpha$-separator problem has been proved $\mathcal{NP}$-hard, since when considering $\alpha = 1/n$, it is equivalent to the minimum vertex cover problem [60]. It can be also proven that, if $\alpha = 2/3$,

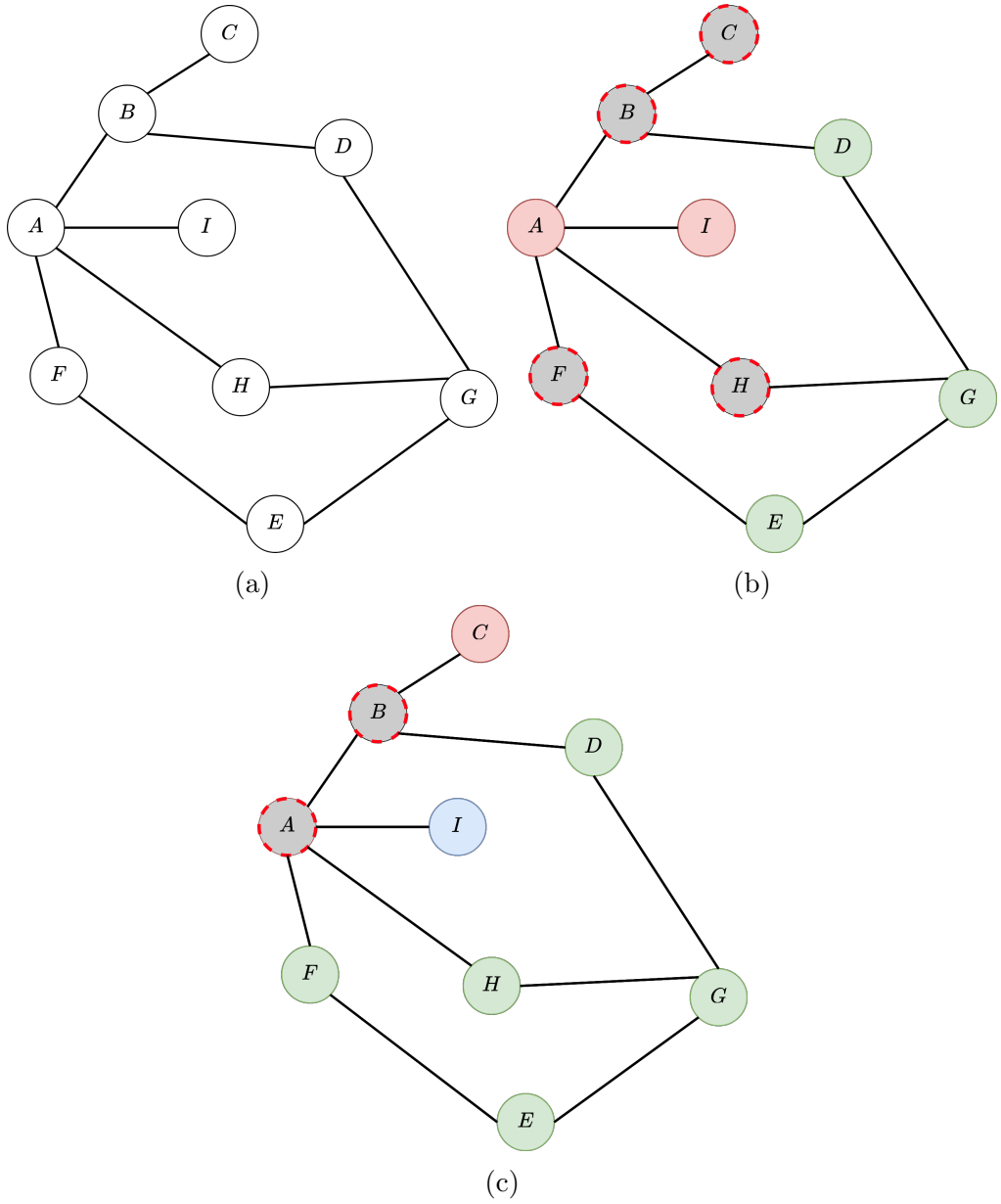



Figure 3.1: Example of a network and two feasible solutions for the $\alpha$-separator problem. The first one (3.1b) has 4 nodes in the separator (`B, C, F`, and `H`). The second one (3.1c) represents a better solution with 2 nodes in the separator (`A` and `B`).

then, the $\alpha$-separator problem is analogous to the minimum dissociation set problem (see [61] for more node-deleting problems). Therefore, it can be considered that the $\alpha$-separator is a generalization of these problems, that were also proven $\mathcal{NP}$-hard in [62].

Regarding at the solution structure, it can be seen that induced graphs can be derived from the original one. Performing the removal of the nodes included in the separator, the graph will be divided into different subgraphs, each one of them corresponding to a different connected component generated after the removal. These induced graphs are usually a good starting point for the Community Detection Problem, in which the nodes that belong to the same induced graph (or connected component) are assigned to the same community and only the nodes included into the separator need to be assigned. This idea is similar to the one applied in [46, 47, 48], where the process of finding communities is reduced to finding cliques.

Another direct application in the context of community detection consists of considering the separator nodes as the centroids for the communities. It is, they can be used to expand the communities from them, given that their presence into the separator indicates that they are relevant nodes in the network that is being analyzed. This is similar to the idea behind the $k$-Nearest Neighbors algorithm, being the main difference that the number $k$ of centroids is not previously defined and the concept of nearness must be adapted to the context of social networks.

## 3.3 The classical Community Detection Problem

In the classical CDP, the main objective is the one exposed in Section 3.1: given a certain network represented as a graph $G$, the main objective is to find a grouping of nodes such that nodes belonging to each group are highly connected to each other and lowly connected to nodes in other groups.

To illustrate the CDP, let us define a network as an undirected graph $G = (V, E)$, where $V$ represents the set of vertices (users) and $E$ represents the set of edges (relations among users). Figure 3.2 depicts an example network with $G = (V, E)$, $V = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F}, \mathsf{G}, \mathsf{H}\}$ and $E = \{(\mathsf{A}, \mathsf{B}), (\mathsf{A}, \mathsf{C}), (\mathsf{A}, \mathsf{D}), (\mathsf{B}, \mathsf{D}), (\mathsf{C}, \mathsf{D}), (\mathsf{D}, \mathsf{E}), (\mathsf{D}, \mathsf{F}), (\mathsf{F}, \mathsf{G}), (\mathsf{F}, \mathsf{H}), (\mathsf{G}, \mathsf{H})\}$. For the sake of clarity, the size of these two sets are defined as $|V| = n$ and $|E| = m$, respectively. In the figure, $n = 8$ and $m = 10$.

Many different valid solutions for this example can be reached. Depending on the metric that is being optimized and the evaluation metric used, one solution is considered better than the others. Figure 3.3 shows a feasible solution for the

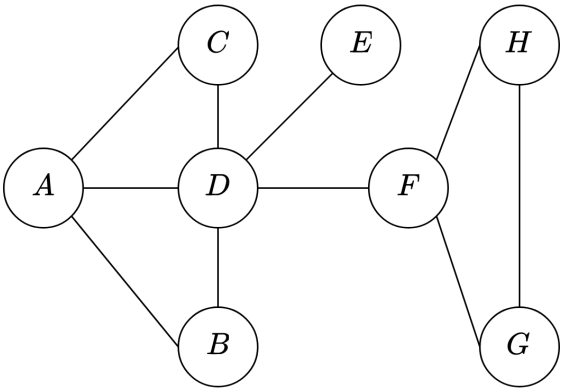



Figure 3.2: Example network for the CDP.

example network. In this example, a solution with three different communities is shown. These three communities can be numbered as $C_1, C_2$ and $C_3$, each one identified with a different colour in the figure. Therefore, this solution can be defined as $S_1 = \{C_1, C_2, C_3\}$, where $C_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}, C_2 = \{\mathtt{D}, \mathtt{E}\}$ and $C_3 = \{\mathtt{F}, \mathtt{G}, \mathtt{H}\}$.

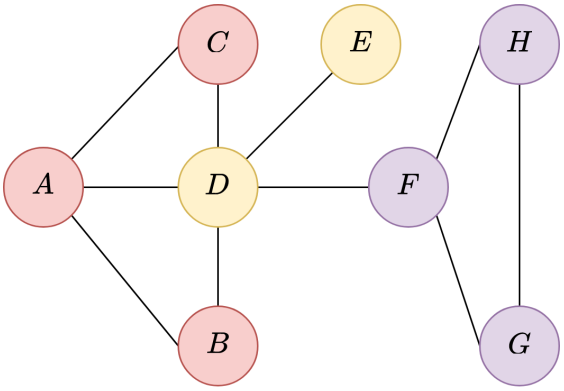



Figure 3.3: Example of a feasible solution for the CDP, with $S_1 = \{C_1, C_2, C_3\}$.

As it has been aforementioned, there is not an unique objective function in CDP. Depending on the context of the social network, the applications of the problem and others circumstances, one objective function will be more suitable than others. Without loss of generality, we consider modularity metric $Q$ (explained in Section 3.1.1) as the objective function for the CDP, which is high when a good community structure is reached in a given network $G$. Having defined this objective, the optimal solution for the CDP for a given network can be formally defined as follows:

$$S^* = \arg\max_{S \in \phi} Q(G, S) \tag{3.9}$$

The CDP is subject to only two constraints: in a given solution, all users must be assigned to one community and a user must be only associated to a single community. More formally,

$$\forall v \in V, \exists i : v \in C_i \text{ and } C_i \cap C_j = \text{ for } 1 \leq i, j \leq \mathcal{K} \tag{3.10}$$

where $\mathcal{K}$ is the number of a communities in a given solution.

Given these definitions, the modularity value for the solution $S_1$ shown in Figure 3.3 can be calculated as:

$$Q(G, S_1) = \left(\frac{2}{10} - \left(\frac{7}{2 \cdot 10}\right)^2\right) + \left(\frac{1}{10} - \left(\frac{6}{2 \cdot 10}\right)^2\right) + \left(\frac{3}{10} - \left(\frac{7}{2 \cdot 10}\right)^2\right) = 0.265 \tag{3.11}$$

As it has been defined in Section 3.1.1, the modularity value is calculated as the sum of the modularity values associated to each community. This value corresponds to the subtraction of two fractions: on the one hand, the number of intra-community edges divided by the number of total edges of the network. On the other hand, the sum of the degree of all nodes belonging to the community divided by 2 times the total number of edges of the network, all squared. Following this definition, for the first community (highlighted in red) the associated value is the number of intra-community edges, 2, divided by the number of edges in the graph, 10, minus the sum of the degree of all nodes belonging to the community, 7, divided by twice the number of edges in the whole graph. The evaluation of the remaining communities is performed in a similar way.

Another possible solution for the Figure 3.2 is the one represented in Figure 3.4. In this second example, a solution with two different communities is reached. These communities can be numbered as $C_1$ and $C_2$, each one identified with a different colour in the figure. Therefore, this solution can be defined as $S_2 = \{C_1, C_2\}$, where $C_1 = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}$ and $C_2 = \{\mathsf{F}, \mathsf{G}, \mathsf{H}\}$.

For this solution, the modularity value can be calculated as follows:

$$Q(G, S_2) = \left(\frac{6}{10} - \left(\frac{13}{2 \cdot 10}\right)^2\right) + \left(\frac{3}{10} - \left(\frac{7}{2 \cdot 10}\right)^2\right) = 0.355 \tag{3.12}$$

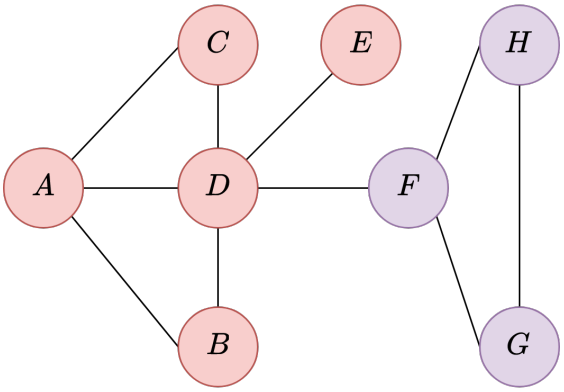



Figure 3.4: Example of a feasible solution for the CDP, with $S_2 = \{C_1, C_2\}$.

Given these both solutions, and the objective function mentioned above, it can be seen that the solution represented in Figure 3.4 is better than the solution in 3.4. But, as it has been aforementioned, in Social Network Analysis context, the metric used as objective function should not be used to compare solutions. In order to compare the two obtained solutions, the conductance metric (defined in Section 3.1.1) is used. It should be recalled that conductance for a single community is calculated as the number of inter-community edges divided by the minimum between the number of edges belonging to the community and the number of edges that does not belong to the community under evaluation. Given this definition, the conductance value for the solution S is calculated as:

$$C_o(S_1, G) = \frac{\frac{3}{min(5,5)} + \frac{4}{min(5,5)} + \frac{1}{min(4,6)}}{3} = \frac{\frac{3}{5} + \frac{4}{5} + \frac{1}{4}}{3} = 0.55 \tag{3.13}$$

Regarding to the second solution $S_2$, its conductance value can be calculated as follows:

$$C_o(S_2, G) = \frac{\frac{1}{min(7,3)} + \frac{1}{min(4,6)}}{2} = \frac{\frac{1}{3} + \frac{1}{4}}{2} = 0.29 \tag{3.14}$$

In view of these results, it can be confirmed that solution $S_2$ is better than $S_1$, as could already be intuited in view of the results obtained using modularity.

## 3.4 The multi-objective Community Detection Problem

Community Detection Problem can be also approached from a multi-objective point of view. In a multi-objective problem, the goal is to optimize two or more different objective that are in conflict, it is, one of them can not be improved without make the others worse. More formally, a multi-objective problem can be described as follows:

$$MOP = \begin{cases} Minimize\ (f_1(\sigma), f_2(\sigma), f_3(\sigma), ..., f_k(\sigma)) \\ subject\ to\ \sigma \in \phi \end{cases} \tag{3.15}$$

where $k$ is the number of objectives and $\sigma$ is a solution to the incumbent problem, $f(\sigma)$ is the objective function that will be optimized and $\phi$ is the feasible solutions set.

Solving the CDP from a multi-objective perspective allows to solve the main problem associated to the single-objective approach. Given that the optimization metrics are traditionally considered in an isolated way, this approach results in the loss of certain information related to the community structure. Tackling the problem in a multi-objective way is a possible solution for this problem, considering multiple metrics simultaneously with the aim of improving the community detection in a social network, focusing in different characteristics at the same time. The challenge faced by this type of approach is to find quality metrics that are also in conflict with each other [63]. Most of the works in the literature are focused on adapting well-known evolutionary algorithms such as NSGA-II to solve different multi-objective community detection problems. Some of them are based in decomposition, such as [64, 65]. Another evolutionary algorithm for solving CDP in a multi-objective way is presented in [66]. The authors use a continuist view with respect to what is considered a good community structure in the single-objective version of the problem, considering as objective functions the maximization of the intra-link strength of the communities and the minimization of the inter-link strength, which are very similar to those considered in [64]. Following this line, in [67] a bioinspired algorithm based on enhanced firefly methodology is proposed. In this work, authors select as objective functions the maximization of the in-degree of the nodes that belong to each community and the minimization of their out-degree. In [1] the authors present a local-based information multi-objective approach for solving the CDP, considering the Negative Ratio Association and Ratio Cut metrics. Negative Ratio Association corresponds to the negated value of Ratio Association metric defined in Section 3.1.1, while Ratio Cut is the same one defined in the aforementioned section.

Without loss of generality, these two metrics will be used to illustrate the Multi-objective Community Detection Problem (MOCDP). As it has been done to exemplify the classical CDP, an undirected graph $G = (V, E)$ representing a network is defined. For the sake of clarity, the same network shown in Figure 3.2 is used. As it occurs in the CDP, different valid solutions for the MODCP can be reached in this example. Again, the quality of a single solution depends on the metrics applied to evaluate and solve the problem. Given the multi-objective nature of the problem, two or more metrics must be used as objective functions. For the sake of clarity, NRA and RC metrics are used as minimization objectives. NRA is the negated version of Ratio Association metric, while RC is the Ratio Cut metric, both of them defined in Section 3.1.1. Having these definitions, solutions represented in Figure 3.5 and in Figure 3.6 represent feasible solutions for the MOCDP.

Different solutions can be reached in this context. Figure 3.5 shows a feasible solution for the proposed network. In particular, a solution with three different communities is presented. These three communities are numbered as $C_1, C_2$, and $C_3$, each one identified with a different colour in the figure. Therefore, this solution can be defined as $S_1 = \{C_1, C_2, C_3\}$, where $C_1 = \{\mathtt{A}, \mathtt{C}, \mathtt{E}\}, C_2 = \{\mathtt{B}, \mathtt{D}, \mathtt{H}\}$ and $C_3 = \{\mathtt{F}, \mathtt{G}\}$.

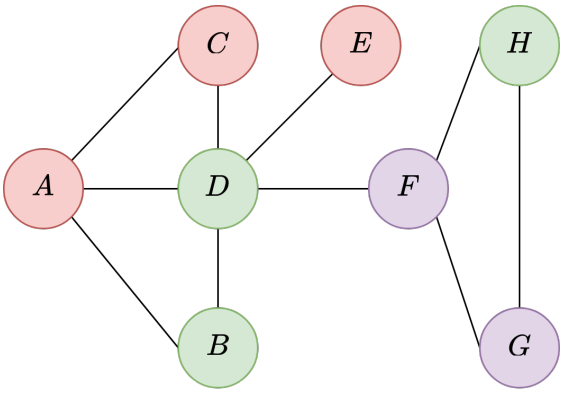



Figure 3.5: Example of a feasible solution for the MOCDP, with $S_1 = \{C_1, C_2, C_3\}$.

As it has been aforementioned, in MOCDP there is more than one objective function to be optimized, and NRA and RC are taken into account. The RC and NRA values for the solution $S_1$ (Figure 3.5) are:

$$\begin{aligned} RC(S_1) = \frac{|E_{C_1\rightarrow}|}{|C_1|} + \frac{|E_{C_2\rightarrow}|}{|C_2|} + \frac{|E_{C_3\rightarrow}|}{|C_3|} = \\ = \frac{4}{3} + \frac{7}{3} + \frac{3}{2} = 5.17 \end{aligned} \tag{3.16}$$

$$NRA(S_1) = -\left(\frac{|E_{C_1\leftarrow}|}{|C_1|} + \frac{|E_{C_2\leftarrow}|}{|C_2|} + \frac{|E_{C_3\leftarrow}|}{|C_3|}\right) = \\ = \frac{1}{3} - \frac{1}{3} - \frac{1}{2} = -1.17 \tag{3.17}$$

Another possible solution for the Figure 3.2 is the one represented in Figure 3.6. In this second example, a solution with four different communities is reached. These communities can be numbered as $C_1, C_2, C_3$ and $C_4$, each one identified with a different colour in the figure. Therefore, this solution can be defined as $S_2 = \{C_1, C_2, C_3, C_4\}$, where $C_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}, C_2 = \{\mathtt{D}, \mathtt{F}\}$, $C_3 = \{\mathtt{E}\}$, and $C_4 = \{\mathtt{G}, \mathtt{H}\}$.

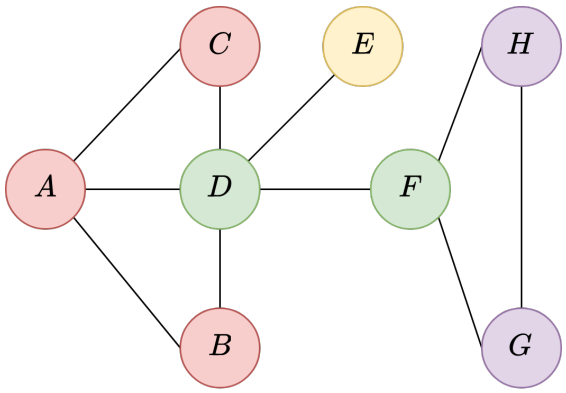



Figure 3.6: Example of a feasible solution for the MOCDP, with $S_2 = \{C_1, C_2, C_3, C_4\}$.

For this solution, the NRA and RC values can be calculated as follows:

$$RC(S_2) = \frac{|E_{C_1\rightarrow}|}{|C_1|} + \frac{|E_{C_2\rightarrow}|}{|C_2|} + \frac{|E_{C_3\rightarrow}|}{|C_3|} + \frac{|E_{C_4\rightarrow}|}{|C_4|} = \\ = \frac{3}{3} + \frac{6}{2} + \frac{1}{1} + \frac{2}{2} = 6 \tag{3.18}$$

$$NRA(S_2) = -\left(\frac{|E_{C_1\leftarrow}|}{|C_1|} + \frac{|E_{C_2\leftarrow}|}{|C_2|} + \frac{|E_{C_3\leftarrow}|}{|C_3|}\right) = \\ = \frac{2}{3} + \frac{1}{2} + \frac{0}{1} + \frac{1}{2} = -1.67 \tag{3.19}$$

As it can be seen, NRA and RC objectives are in conflict (as it has been proven in [1]). It means that one objective function can not be improved without

deteriorating the others. This situation implies that it cannot be said that one solution is better than other one. To provide a definition of optimality in the multi-objective context, a more global point of view is needed. Chapter 4 enumerates different techniques that can be applied to deal with this problem.

## 3.5 The Overlapping Community Detection Problem

In both classical and multi-objective CDP variants a common constraint must be satisfied to guarantee the feasibility of a solution: one node (user) must be assigned to one and only one group (community). Nevertheless, in real-world networks this is a constraint that is rarely met, since one user can belong to different groups simultaneously. For example, in the context of Facebook social network, an user could be part of a group that is interested in basketball and, simultaneously, belong to a musicians group, So a user does not exclusively belongs to a certain community, but more than once. To deal with this reality, the Overlapping Community Detection Problem (OCDP) [68] is tackled. It describes problems in which users have to be grouped in different communities, but each user may belong to more than one community at the same time. In the literature different authors have addressed the detection of overlapping communities. Two of the most extended algorithms to solve this problem are *Cluster-Overlap Newman Girvan Algorithm* (CONGA) [69] and *Cluster-Overlap Newman Girvan Algorithm Optimized* (CONGO) [70]. CONGA is based on the traditional Girvan-Newman algorithm [40], but splitting the vertices in a different way. CONGA has the same computational complexity that Girvan-Newman, which is $\mathcal{O}(m^3)$ (being $m = |E|$). With the aim of improving this complexity, CONGO was proposed in 2008, reducing the algorithmic complexity to $\mathcal{O}(n \cdot \log n)$ where $n$ represents the number of nodes. In the literature it can be found algorithms that have been adapted to solve the overlapping version of CDP. Some of them are the *Clique-Percolation-Method* (CPM) [71] or *Label Propagation* [72]. All these algorithms applied to Overlapping CDP suffer from the same drawbacks as the non-overlapping algorithms: the computational cost required to evaluate solutions. One of the latest research papers published that uses a metaheuristic approach for this purpose is the one authored by Xu et al. [2]. In this work the authors propose an extended adaptive version of the density peaks algorithm [73] for overlapping CDP, named EADP (Extended Adaptive Density Peaks). This algorithm selects the centre for the different communities and allocates to those centres the nodes following the density peak algorithm. To select these centres, it computes the distance to any pair of nodes. This distance depends on the number of nodes connected to them.

In the context of OCDP, new evaluation and optimization metrics must be proposed to evaluate the obtained solutions. It is because in the overlapping variant of CDP, the main constraint of the classical variant is removed. In the OCDP, a single node can be assigned to more than one group simultaneously. Another option is to adapt the existing metrics to make them fit this new situation. For example, in [74] authors propose an adaptation of the traditional modularity for the overlapping scenario, taking into account the possibility for a node to belong to several communities at once. This adaptation consists of adding a factor that represents the number of different communities to which a node belongs to. For a single solution $S$, the overlapped modularity is calculated as the sum of this value for each community and, then, averaging the results for all communities. Mathematically,

$$M_O(S) = \frac{\sum_{k=1}^{\mathcal{K}} M_O(C_k)}{|S|} \tag{3.20}$$

where $|S|$ is the number of detected communities in the incumbent solution. Given a community $C_k = (V_k, E_k)$ its overlapping modularity $M_O(C_k)$ is evaluated as:

$$M_O(C_k) = \frac{1}{|V_k|} \sum_{u \in V_k} \frac{|E_{C_k \leftarrow}(u)| - |E_{C_k \rightarrow}(u)|}{d_u \cdot s_u} \cdot \frac{|E_{C_k}|}{\frac{|V_k| \cdot (|V_k|-1)}{2}} \tag{3.21}$$

where $d_u$ indicates the degree of node $u$, and $s_u$ represents the number of groups $u$ has been associated to. Finally, given that the size of each community could be rather different, a normalization of the result is need to be applied. To do so, the difference between intra and inter-community edges is multiplied by the ratio between the number of edges that actually exists in the community ($|E_i|$), and the number of edges that a community conformed by a complete graph representing this community would have ($\frac{|V_i| \cdot (|V_i|-1)}{2}$).

Using any metric adapted to the context of overlapping community detection as the maximization objective function ($f_1$), the optimal solution for OCDP can be defined as follows:

$$\sigma^* = \arg\max_{\sigma \in \phi} f_1(G, \sigma) \tag{3.22}$$

To illustrate the OCDP, the example network $G$ from Section 3.3 is used as the network under evaluation. Nevertheless, given the overlapping nature of the problem, different solutions will be reached. With the aim of evaluating the solutions quality and compare them, the previously exposed overlapped modularity metric is used. An example of feasible solution is presented in Figure 3.7. Solution $S_1$ is defined as $S_1 = \{C_1, C_2, C_3\}$, where $C_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}\}, C_2 = \{\mathtt{D}, \mathtt{E}, \mathtt{F}\}$ and $C_3 = \{\mathtt{F\,G}, \mathtt{H}\}$.

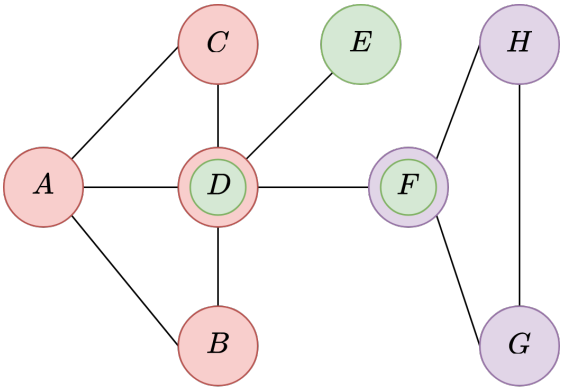



Figure 3.7: Example of a feasible solution for the OCDP, with $S_1 = \{C_1, C_2, C_3\}$.

As it can be seen, in this solution nodes `D` and `F` are overlapped nodes, it is, they belong to two communities at the same time. This is the main characteristic of OCDP. As it has been aforementioned, this value is calculated as the average of the sum of the individual modularities of each community. More specifically,

$$M_O(C_1) = \frac{1}{4} \cdot \left( \frac{3-1}{3 \cdot 1} + \frac{2-1}{2 \cdot 1} + \frac{2-1}{2 \cdot 1} + \frac{3-2}{5 \cdot 2} \right) \cdot \left( \frac{7}{\frac{4 \cdot 3}{2}} \right) = 0.52 \tag{3.23}$$

$$M_O(C_2) = \frac{1}{3} \cdot \left( \frac{2-4}{5 \cdot 2} + \frac{1-0}{1 \cdot 1} + \frac{1-3}{3 \cdot 2} \right) \cdot \left( \frac{7}{\frac{3 \cdot 2}{2}} \right) = 0.36 \tag{3.24}$$

$$M_O(C_3) = \frac{1}{3} \cdot \left( \frac{2-1}{3 \cdot 2} + \frac{2-1}{2 \cdot 1} + \frac{2-1}{2 \cdot 1} \right) \cdot \left( \frac{4}{\frac{3 \cdot 2}{2}} \right) = 0.52 \tag{3.25}$$

So, for this particular solution $S_1$, the modularity value is calculated as follows:

$$M_O(G, S_1) = \frac{0.52 + 0.36 + 0.52}{3} = 0.47 \tag{3.26}$$

Another example of feasible solution for the OCDP is shown in Figure 3.8. This time, the solution $S_2$ is defined as $S_2 = \{C_1, C_2\}$, where $C_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}\}$ and $C_2 = \{\mathtt{D}, \mathtt{F}, \mathtt{G}, \mathtt{H}\}$.

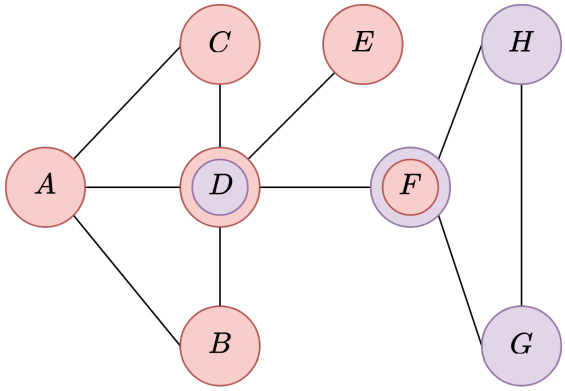



Figure 3.8: Example of a feasible solution for the OCDP, with $S_2 = \{C_1, C_2\}$.

Once again, the solution presents D and F as overlapped nodes (they are assigned to communities $C_1$ and $C_2$ simultaneously). The modularity for this second solution can be calculated as:

$$M_O(C_1) = \frac{1}{6} \cdot \left( \frac{3-1}{3 \cdot 1} + \frac{2-1}{2 \cdot 1} + \frac{2-1}{2 \cdot 1} + \frac{5-1}{5 \cdot 2} + \frac{1-0}{1 \cdot 1} + \frac{1-3}{3 \cdot 2} \right) \cdot \frac{9}{\frac{6 \cdot 5}{2}} = 0.27 \tag{3.27}$$

$$M_O(C_2) = \frac{1}{4} \cdot \left( \frac{1-5}{5 \cdot 2} + \frac{3-1}{3 \cdot 2} + \frac{2-1}{2 \cdot 1} + \frac{2-1}{2 \cdot 1} \right) \cdot \frac{8}{\frac{4 \cdot 3}{2}} = 0.31 \tag{3.28}$$

$$M_O(G, S_2) = \frac{0.27 + 0.31}{2} = 0.29 \tag{3.29}$$

In view of these results, it can be said that $S_1$ is better than $S_2$. It is important to remark that, as it happens in the classical and multi-objective variants of the problem, the objective function and the evaluation function can be adapted depending on the work context or network structure.

## 3.6 The Dynamic Community Detection Problem

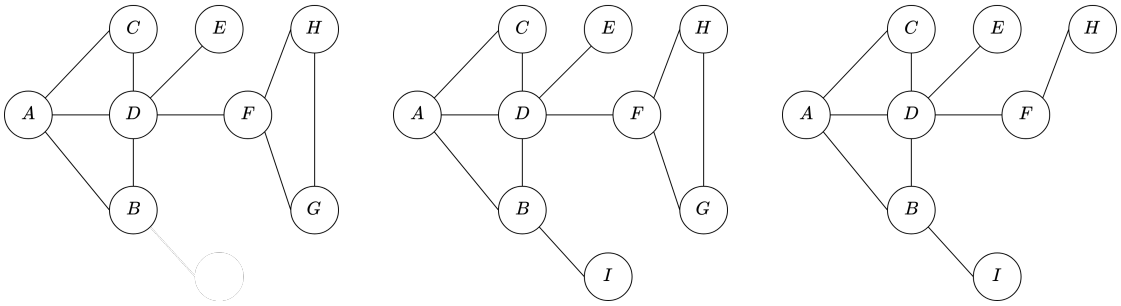
So far, the variants of the problem analyzed have focused on the study of static networks, it is, a single graph with no changes is analyzed and a single solution is provided for this network. Nevertheless, this situation does not fit very well with the real world. Every day millions of new users and relations among them appear or disappear from a social network, so it is necessary to develop algorithms that are capable to suit this reality. In this context, the Dynamic Community Detection Problem (DCDP) is proposed. In this variant of the problem, a network is analyzed in different instants of time, also known as snapshots. These snapshots are taken in intervals of time that are longer or shorter depending on the nature of the network under analysis. In each one of these snapshots the network is evolving. This evolution can occur in four different ways:

- One or more nodes appear in the graph, it is, there are new users that join to the social network.
- One or more edges appear in the graph, it is, there are new relations between network users.
- One or more nodes disappear from the graph. If it occurs, then there are users that left the social network. Typically, when an user disappears, the relations associated to him (i.e, the edges that have and endpoint in the disappeared user) are removed too.
- One or more edges disappear from the graph. In this situation, there are users who no longer interact at a given time instant.

Many authors in literature have been approached the DCDP problem. As it occurs in the context of CDP, regardless of the variant, each author propose a new methodology to reach the same objective: generate solutions with good community structure. In [75] authors use a Particle Swarm Optimization algorithm to face the Dynamic Community Detection Problem. In this work, the concept of consensus community is proposed. This concept is the one which is used to propagate information between different snapshots of the same instance. Authors in [76] propose an evolutionary algorithm that is supported in the Evolutionary Clustering Framework. In this work *sE-Autoencoder* algorithm is proposed. It is a semi-supervised algorithm that overcomes the problems that prevent the use of the original framework. The dynamic version of the Community Detection Problem can be also tackled from a multi-objective point of view. An example of this approach can be found at [77], where the authors propose an immune algorithm

to optimize simultaneously modularity and normalized mutual information [78] metrics.

With the aim of illustrating the DCDP, Figure 3.9 represents the example of a network with three snapshots ($T_1, T_2$ and $T_3$) taken in three different instants of time. As it can be seen, the four possible changes in the context of DCDP are represented in the example.




(a) Snapshot $T_1$. (b) Snapshot $T_2$. (c) Snapshot $T_3$.

Figure 3.9: Example of network evolution for three different snapshots.

To solve the DCDP, all snapshots must be solved independently. It means that a full solution for the DCDP is represented by the set of solutions for each snapshot. More formally, a solution $S$ can be defined as $S = \{S_{T_1}, S_{T_2}, .., , S_{T_y}, \}$, where $y$ is the number of snapshots, and $S_{T_y} = \{C_1, C_2, ...C_n\}$, where $n$ is the number of communities detected in a single snapshot. The process to obtain the solution of the second and subsequent snapshots can be accelerated and improved by regarding at the changes that occur in the network and the solution obtained from the previous snapshot (with respect to the current one). It is the main difference of DCDP with respect to the other variants. This situation implies that the best solution of one snapshot (in terms of the evaluation metric) can be used as starting point to the next one. As usual in CDP problems, the objective function and the evaluation metric are not the same. It means that, in each snapshot, the solution is built by optimizing the objective function, but the best solution, it is, the one that marks the starting point for the next snapshot is the evaluation metric. Therefore, it can be said that the optimal solution is the one that maximizes (or minimizes) the objetive function. In mathematical terms,

$$\sigma^* = \arg\max_{\sigma \in \phi} f_o(G, \sigma) \tag{3.30}$$

where $f_o$ is the selected objective function for the current problem. For the sake of clarity, modularity is considered as objective function to illustrate the DCDP,

and Community Score is selected as evaluation metric.

A possible solution for the example network $G$ proposed in Figure 3.9 is shown below. In the first snapshot, a solution is built from scratch, given that there is not previous information that can be exploited. However, this first solution is a good starting point to solve the second snapshot. Figure 3.10 represents a feasible solution for the first snapshot.

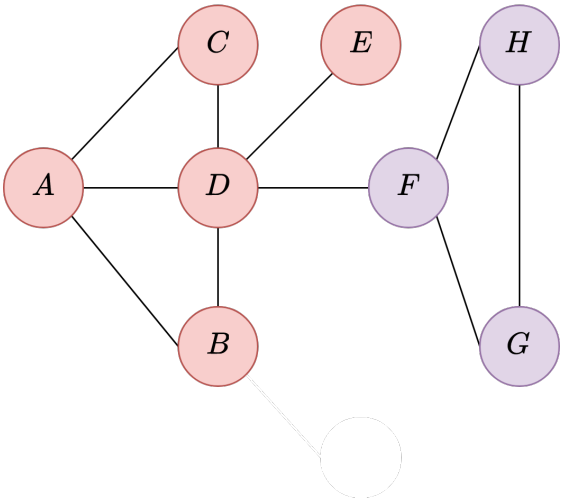
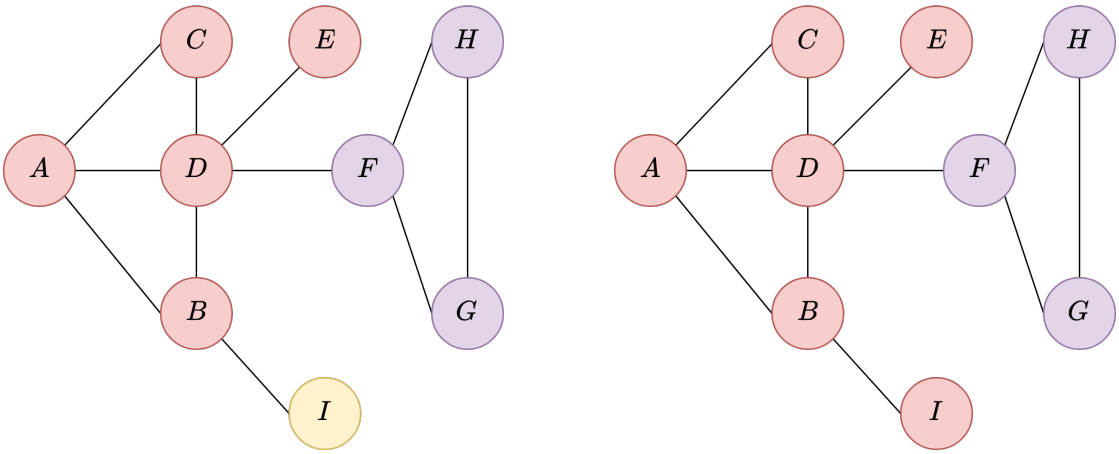



Figure 3.10: Example of a feasible solution for snapshot $T_1$ with $S_1 = \{C_1, C_2\}$.

From this point on, different solutions can be reached. Figure 3.11a and Figure 3.11b represent two feasible solutions for the second snapshot.




(a) Solution $S_{2_a}$ for snapshot $T_2$.

(b) Solution $S_{2_b}$ for snapshot $T_2$.

Figure 3.11: Example of two different feasible solutions for snapshot $T_2$.

Solution $S_{2_a}$ can be defined as $S_{2_a} = \{C_1, C_2, C_3\}$, where $C_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}\}$, $C_2 = \{\mathtt{F}, \mathtt{G}, \mathtt{H}\}$ and $C_3 = \{\mathtt{I}\}$. Its modularity value can be calculated as follows:

$$Q(G, S_{2_a}) = \left(\frac{6}{11} - \left(\frac{14}{2 \cdot 11}\right)^2\right) + \left(\frac{3}{11} - \left(\frac{7}{2 \cdot 11}\right)^2\right) + \left(\frac{0}{1} - \left(\frac{1}{2 \cdot 11}\right)^2\right) = 0.31 \tag{3.31}$$

Solution $S_{2_b}$ can be defined as $S_{2_b} = \{C_1, C_2\}$, where $C_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{I}\}$ and $C_2 = \{\mathtt{F}, \mathtt{G}, \mathtt{H}\}$. The modularity value for this solution is calculated as:

$$Q(G, S_{2_b}) = \left(\frac{7}{11} - \left(\frac{15}{2 \cdot 11}\right)^2\right) + \left(\frac{3}{11} - \left(\frac{7}{2 \cdot 11}\right)^2\right) = 0.34 \tag{3.32}$$

To select which of them will be the starting point to the third snapshot, Community Score of both solutions is calculated. For solution $S_{2_a}$, the Community Score value is:

$$CS(S_{2_a}, G) = \frac{2}{3} + \frac{1}{1} + \frac{1}{3} = 2 \tag{3.33}$$

Regarding to the second solution $S_{2_b}$, its community score is calculated as follows:

$$CS(S_{2_b}, G) = \frac{1}{3} + \frac{1}{3} = 0.67 \tag{3.34}$$

In view of these results, it seems clear that solution $S_{2_b}$ is better than solution $S_{2_a}$ in terms of both objective function and evaluation metric, so it is the selected one as starting point for the third snapshot. Finally, a feasible solution for the third snapshot could be the one represented in Figure 3.12.

With this solution, the modularity and community score values for the last snapshot are calculated as follows:

$$Q(G, S_3) = \left(\frac{7}{9} - \left(\frac{15}{2 \cdot 9}\right)^2\right) + \left(\frac{1}{9} - \left(\frac{3}{2 \cdot 9}\right)^2\right) = 0.17 \tag{3.35}$$

$$C(S_3, G) = \frac{1}{1} + \frac{1}{2} = 1.5 \tag{3.36}$$

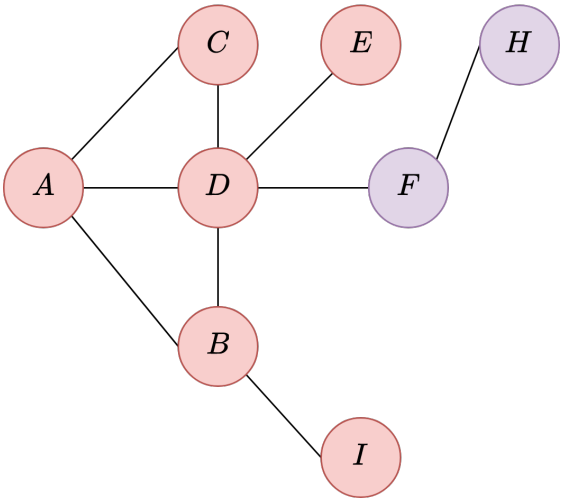



Figure 3.12: Example of a feasible solution for snapshot $T_3$ with $S_3 = \{C_1, C_2\}$.

# Chapter 4

# Methodology

*Metaheuristic algorithms have been proven to be a great instrument to provide solutions to complex combinatorial optimization problems. Real-world tasks such as to decide the best route for a vehicle float minimizing the global cost, select the best location to open a new hospital in such a way the people covered by its services is maximized or schedule the jobs of an assembly line are examples of problems where metaheuristics emerge as a well option to obtain high quality solutions in low computing times. Example of these metaheuristics are Iterated Greedy, GRASP, Tabu Search, Evolutionary Algorithms, or Scatter Search, among others.*

In recent years, the metaheuristics research area has received the attention of many researchers from different fields. Statistics, engineers, computer scientists and operational research analysts, among others, have encountered in metaheuristics a field that can be suitable and proficient to their research. A clear proof of this fact is the large amount of international conferences devoted to spread the knowledge in the metaheuristics area and its application in other research fields. Example of these conferences are Metaheuristics International Conference (MIC), the International Conference on Variable Neighborhood Search (ICVNS), or the Genetic and Evolutionary Computation Conference (GECCO).

In this Chapter, the main metaheuristic techniques applied in the context of this Thesis are described. These algorithms have been applied for solving $\mathcal{NP}$-hard problems summarized in Chapter 3. In particular, Greedy Randomized Adaptive Search Procedure (GRASP, Section 4.1), Variable Neighborhood Search (VNS, Section 4.2), Iterated Greedy (IG, Section 4.3) and Path Relinking (PR, Section 4.4) are described. Furthermore, different techniques to apply when a multi-objective problem is tackled are described in Section 4.5.

## 4.1 Greedy Randomized Search Procedure

Greedy Randomized Search Procedure (GRASP) is a metaheuristic developed by T. Feo and M. Resende as an algorithm to solve set covering problems [21]. In year 1994 it acquired a definitive terminology and form as a general purpose metaheuristic [22]. GRASP is a multi-start procedure [79, 80, 81] in which each start corresponds to an iteration of the algorithm. Each iteration has two well-differentiated phases: construction phase, that is in charge of obtaining a high-quality feasible solution, and the improvement phase, that locally optimizes the solution obtained in the previous phase. Pseudocode shown in Algorithm 1 includes a complete description of the algorithm behavior. In this algorithm, parameter $C$ represents the components for a certain solution to a given problem.

**Algorithm 1** *GRASP*$(C, \alpha)$

1: $c \leftarrow selectSeed(C)$
2: $S \leftarrow \{c\}$
3: $CL \leftarrow C \setminus \{c\}$
4: **while not** *isFeasible*$(S)$ **do**
5: $\quad g_{\min} \leftarrow \min_{c \in CL} g(c)$
6: $\quad g_{\max} \leftarrow \max_{c \in CL} g(c)$
7: $\quad \mu \leftarrow g_{\max} - \alpha \cdot (g_{\max} - g_{\min})$
8: $\quad RCL \leftarrow \{c \in CL \ : \ g(c) \geq \mu\}$
9: $\quad c \leftarrow SelectElement(RCL)$
10: $\quad S \leftarrow S \cup \{c\}$
11: $\quad CL \leftarrow CL \setminus \{c\}$
12: **end while**
13: *Improve*$(S)$
14: **return** $S$

GRASP comes from the semi-constructive heuristic proposed in [82], which is also characterized because it is a multi-start method based in a greedy randomized construction. The main difference with respect to GRASP is that this technique did not use an improvement procedure.

The construction phase is an iterative procedure in charge of building a solution $S$ element by element. Initially, it starts from a seed that is a component or components set that determine a partial solution. This seed can be randomly selected or, if it is known that certain sub-structures are part of the optimal solution, they can be the seed [83, 84] (step 1). The included components are marked as non available and the rest are included in the selectable elements set: the *Candidate List* ($CL$). Once the Candidate List is built, it is sorted using a greedy function

that assigns a value to each candidate. This algorithm phase corresponds to the *Greedy* word in the name of the metaheuristic (step 3).

Once the Candidate List is sorted, a good candidate must be selected. In the GRASP context, the best candidate is not selected since, in that case, the algorithm becomes purely greedy and, therefore, the same solution is constructed, eliminating the diversification phase. For this reason, a candidate is randomly selected from a subset of candidates. This subset is named *Restricted Candidate List* or *RCL*. To build the *RCL*, a threshold is defined using the maximum and minimum cost values assigned to the elements in the Candidate List in a certain iteration. If $c_{max}$ and $c_{min}$ are the higher and the lower cost values, respectively, the *RCL* is conformed by all the elements whose cost does not exceed (in a minimization problem) the threshold provided by the following expression:

$$\mu = c_{max} - \alpha \cdot (c_{max} - c_{min}) \tag{4.1}$$

where the $\alpha$ parameter is a value between 0 and 1 ($0 \leq \alpha \leq 1$) that determines the *RCL* size (steps 5-8). On the one hand, if $\alpha = 1$, then the the algorithm is purely greedy (only the best candidate is present in the *RCL*). On the other hand, if $\alpha = 0$, then the algorithm is totally random (all the candidates are included in the *RCL*). In the standard implementations of GRASP, the $\alpha$ parameter is randomly selected in each iteration. There are different studies about how GRASP behave depending on the $\alpha$ value. Examples of these studies can be found at [85, 86, 87]. Diverse strategies have been proposed to select the $\alpha$ value, among which the following stand out:

- Randomly select the $\alpha$ value from an uniform distribution of discrete probability (in the general case) [85].
- Automatically adjust the $\alpha$ value regarding to the recently obtained solutions quality (reactive GRASP) [88, 89].
- Select the $\alpha$ value from a non-uniform discrete descending distribution, where there are a higher probability of choose the better values. [88, 90].
- Fix the $\alpha$ value to certain number (similar to the pure greedy selection) [86, 87].

Figure 4.1 shows a representation of the original Candidate List (CL) and the Restricted Candidate List (RCL) generated with the best candidates present in the CL. The RCL size is marked by the vertical line, that depends on the $\alpha$ value.

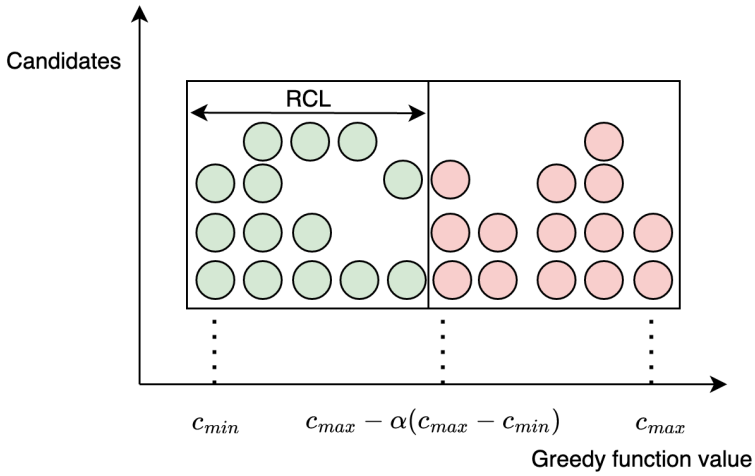



Figure 4.1: GRASP Candidate List and Restricted Candidate List composition.

Once a candidate from the RCL has been selected (step 9 in Algorithm 1), it is introduced in a partial solution $S$ (step 10) and it is marked as non-selectable element (step 11). The rest of the elements are still selectables, so the contribution to the objective function of each one of them if they were part of the solution must be re-calculated. This calculus is done through the greedy function established. Therefore, the partial solution of GRASP is adapted each time that a new candidate is added to it. The constructive phase of GRASP ends when a feasible solution is reached (step 4). This initial solution is not necessarily a local optimum, given there are many stochastic elections. It implies that constructive phase does not guarantee an optimality of the solution with respect to a certain neighborhood structure.

To deal with this problem, GRASP applies a second phase (improvement phase) that consists of a local optimization procedure based on a local search function or, even, in a complete metaheuristic (step 13). Generally, this phase improves the initial solution, but it does not guarantee the optimality of the obtained solution. Nevertheless, it is proven experimentally that the quality is considerably improved.

In the standard GRASP implementation, the improvement phase is a local search procedure. This fact implies that a neighborhood structure must be defined and examined to make a movement to the neighbor that produces some improvement in the objective function. This movement must maintain the feasibility of the solution, and the local improvement method is executed until no better solutions can be found in the explored neighborhood.

In [85] there are some that are considered determining factors influencing local search:

- The neighborhood structure, that is typically simple.
- The local optimization algorithm applied for the neighborhood. Depending on the type of movement, the improvement methods can be classified into:
  - First-Improvement, that consists of selecting the movement that leads to the first neighbor that improves the incumbent solution.
  - Best-Improvement, that consists of evaluating all the movements in the neighborhood, selecting the one that produces the highest profit.

  In practice, both alternatives may produce similar solutions in terms of quality. Therefore, given that the second method has a higher computational effort associated, the first one is more extended. In addition, it has been empirically observed that with the second option GRASP converges with higher probability to non-global optima.
- Evaluation of the greedy function of the candidates.
- The initial solution by itself. The objective is to build high-quality solutions to minimize as much as possible this fact.

The improvement phase ends when there are not movements that leads to a better solution.

There are different methods that can be introduced in the GRASP standard implementation that can result in important improvements for certain problems. The following is a brief description of those considered most relevant:

- Reactive GRASP: This method provides memory to GRASP, making the $\alpha$ value election non-random, but taking into account the past results. In this framework, those $\alpha$ values that have lead to better solutions in the past are more probably chosen. In [90] a selection rule is proposed. Generally, this implementation improves the standard GRASP results [89].
- Cost perturbation: this method consists of adding a slight noise to the costs in a similar way to noisy methods [91]. This options adds flexibility to the GRASP implementation, specially in those problems that are not so sensible to the randomization. It is useful too when there is not a simply randomizable greedy function [92, 93, 10].
- Bias functions: this technique establish an smarter criterion to select candidates from the RCL, in such a way that, instead of all candidates being

equi-probable, a distribution function is used that emphasizes some candidates over others. In [94] different probability distributions are proposed and in [90] they are applied to the job shop scheduling problem [90, 92, 10].

- Smart construction (memory and learning): this method consist of introducing a long-term memory in the GRASP framework, in such a way that the past history is taken into account when making a decision. It was originally proposed by Fleurent and Glover [95] as an useful strategy for all multi-start metaheuristics.

## 4.2 Variable Neighborhood Search

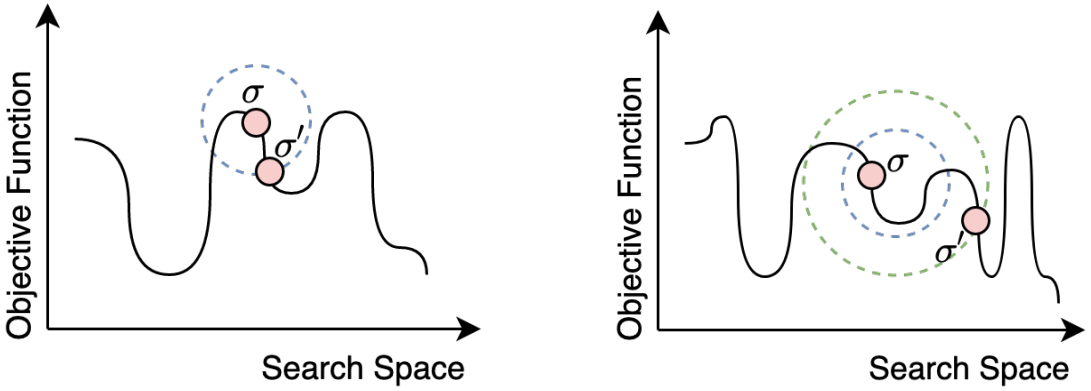
Variable Neighborhood Search (VNS) is a metaheuristic (also considered a metaheuristics framework) proposed by Hansen and Mladenović [96, 23] in the late 90's. For the sake of simplicity, it can be said that it tries to avoid to get stuck in local optima by performing systematic neighborhood changes, both when searching for a local optima and trying escape from it. In the original proposal, the three main variants of VNS were presented: Variable Neighborhood Descent, Basic VNS and Reduced VNS. Later, when the framework began to catch the attention of researchers, new schemas (like General VNS or Skewed VNS [97]) where proposed and described [98].

The basic principle of the VNS schema is based on the systematic changes in the structure of a neighborhood inside a local search procedure. The novel idea that VNS introduces is the handling of a determined set of neighborhood structures. Under this concept, for each solution $\sigma$ that belongs to the solutions space $\phi$ a set of neighborhoods $N_k(\sigma)$ is defined, where $1 \leq k \leq k_{max}$, being $k$ the neighborhood under exploration and $k_{max}$ the maximum neighborhood to be explored. These neighborhoods are built following one or more metrics that are context-based, it is, are specific of the problem that is being solved. As it has been described in [98], VNS is based in the following points:

1. A local optimum with respect to a neighborhood $N_i(\sigma)$ could not be a local optimum with respect to another neighborhood $N_j(\sigma)$.
2. A global optimum is a local optimum with respect to all possible neighborhood structures.
3. Local optima with respect to one or more neighborhood structure are relatively close among them for many problems.

It is worth mentioning that last affirmation has not been proven experimentally, it is an empirical observation. This observation implies that, sometimes, the

local optimum provides information about the global optimum. For example, a local optimum can contain several elements that are present in the global optimum, that normally are not known [99]. When an optimization problem is being solved, a solution can be defined as locally optimal only with respect to a single neighborhood structure. Stating this for different neighborhood structures is not necessarily true and, in fact, probably is not. Therefore, the solution space landscape is determined by the neighborhood structure defined [100]. It means that defining the neighborhood structure implies determining the topological properties of the search space. For the sake of clarity, Figure 4.2 shows two different profiles corresponding to two different neighborhood structures. In Figure 4.2a, the VNS framework starts from point $x_0$ and it could reach the local optimum represented in $x_1$ through some local search procedure. VNS would be stuck in this point of the solutions space. Figure 4.2b represents a situation where, by increasing the neighborhood size and, therefore, also increasing the jump size, the neighborhood structure is changed. As a consequence, some peaks of the search space could disappear, it is, the landscape of the search space is also changing. Applying these changes, the solution $x_2$ could be reached through a local search procedure.




(a) (b)

Figure 4.2: Landscape of the search space for two different neighborhood structures.

The effectiveness of VNS methodology has lead the scientific community to develop several variants, which can be classified according to the way in which the three main aforementioned ideas are combined (deterministic or stochastic way).

Reduced VNS (RVNS) [101, 102] is focused in diversification, considering stochastic changes of neighborhoods. This strategy randomly selects a solution from a certain neighborhood, without applying a local search procedure to it. This strategy is a good option when the computational time required by the local

search procedure is large or, even more, were local search procedures are really difficult to design.

On the other hand, Variable Neighborhood Descent (VND) [103, 104] is devoted to intensification by performing deterministic exploration of the search space. In this implementation, the different neighborhoods are explored in an exhaustive way, changing the neighborhood when a local optimum is reached with the aim of escaping from it. Given this behavior, the obtained solution is a local optimum with respect to all the explored neighborhoods.

Finally, Basic VNS (BVNS) [105] emerges as a compromise between diversification and intensification by combining stochastic and deterministic changes of neighborhoods. This balance comes from the way BVNS selects a solution from the neighborhood (randomly, as in RVNS) and the process that it applies to it after selecting it (a local search procedure).

As a result of the successful application of the methodology to different problems, several new variants have been proposed. The combination between BVNS and VND results in General VNS (GVNS) [106]. In this variant, a random solution is in the neighborhood is selected (as in BVNS) but the local search procedure is replaced by a complete VND algorithm. For this reason, GVNS reaches a local optimum from every randomly selected solution in a certain neighborhood, for all the explored neighborhoods. This behavior eventually leads to better solutions.

Other implementations found in the literature are Variable Neighborhood Decomposition Search (VNDS) [107], Skewed VNS (SVNS) [108], or Variable Formulation Search (VFS) [109], among others.

## 4.3 Iterated Greedy

Iterated Greedy (IG) is a metaheuristic originally proposed in [110] by Rubén Ruiz and Thomas Stützle to solve the permutation flowshop scheduling problem. It is based on the idea of repeating greedy constructions of solutions, but reducing the main problems associated to this methodology: time-consuming constructions, no information taken from one construction to another one, and, thus, repeated constructions that does not exploit the knowledge extracted from the past constructions. To do this, IG generates an initial solution and, then, it partially destroys and regenerates the initial solution resulting in a different one, repeating the process until certain termination condition is met. This condition is context-dependent and could be defined as a certain number of iterations, a number of iterations without improvement or simply be time-limited. Typically, a local search procedure is executed after the regeneration method, with the aim of finding a

local optimum in the neighborhood of the reconstructed solution. As it can be derived from this description, Iterated Greedy requires of an initial solution to operate. This initial solution can be randomly generated or be built using a greedy algorithm, even using a whole metaheuristic (like GRASP).

From an algorithmic point of view, Iterated Greedy consists of three differentiated phases:

1. Destruction phase: in this step, some of the elements belonging to the solution are removed, resulting in a partial solution.
2. Construction phase: starting from the partial solution resulting from the previous step, a construction heuristic is applied, generating a new complete solution.
3. Acceptance Test: in this step, the next solution to be destroyed is selected. This solution could be the initial solution or the solution resulting from the construction phase. In the simplest case, the best solution regarding to the objective function is accepted.

It is important to remark that the strategy used to generate the initial solution and the procedure used in the construction phase could be different and use different heuristics. The advantage of using different constructive heuristics in the context of an iterative process provides to Iterated Greedy algorithm with some advantages comparing it with the construction of different solutions from scratch. First, the required time to build a solution is significantly reduced by two main reasons: less constructive steps are needed and the required decision time per construction is reduced as there are less solution components to be chosen while the larger the partial solution is. Furthermore, a precise acceptance criterion can intensify the search process, finding the best solutions of search space region explored.

The success of the Iterated Greedy applications resides in the followed strategy when destroying and reconstructing solutions. There are different approaches that can be followed when destruction and construction phases are being developed. Some of the main important points to be taken into account are the following:

- In the destruction phase, it must be decided how many components of the solution are going to be destroyed. There are two extreme settings concerning to this point. On the one hand, only one component is destroyed. On the other hand, the solution is totally destroyed, it is, all components in it are removed. Nevertheless, these extreme situations are not suitable with the main principles of Iterated Greedy. In the first case, the behavior of the

algorithm would be similar to a randomized local search procedure. In the second one, it could result in a multi-start approach since the solution resulting from each iteration will be completely different from the original ones. Intermediate values result in a balance between diversification and intensification in the search procedure: by removing a small number of components leads to a more intense search, while removing a larger number of solution components the algorithm explores more distant solutions in the solutions space. Once this decision is taken, another choice must be made: to establish a fixed number of components to be removed or to make it variable during the algorithm execution. If it is variable, a technique to adapt this value is required. If it is fixed, it must be experimentally tuned. The last question that need to be answered when implementing Iterated Greedy destruction phase is which components should be chosen. Again, there are different possible answers. First, destruct components randomly escapes from the cycling risk that implies doing it in a deterministic way. Using a stochastic destruction, components could be uniformly chosen at random. More sophisticated processes could take into account metrics regarding the components or the remaining partial solutions, introducing a bias in the choice, prioritizing components that have larger contributions (lower costs) to the objective function.

- Regarding to the construction phase, using a constructive mechanism is crucial, given that one of the main ideas in IG is that the repairing phase is performed using one of them. Typically, the selected constructive mechanism has a greedy behavior and iterates in a deterministic way. However, it is not necessarily be the case, given that any constructive mechanism that can generate a complete solution starting from a partial one can be used in the Iterated Greedy algorithm. In the construction phase, two main groups of heuristics can be distinguished: the adaptive and the static ones. In an adaptive approach, the partial solution influences the heuristic value assigned to a particular option. This will normally result in better quality solutions than the static approach. However, this quality is typically associated to a higher computational effort. Given the flexibility of Iterated Greedy, any algorithm compatible with the idea of constructing solutions is compatible with this technique.

- Thinking about the acceptance criterion selected, it can be seen that it has a high influence on the diversification-intensification behavior of IG. Accepting any new solution without regard to its quality is as valid as accepting only solutions with higher quality than the previous one. Again, the richness of the method lies in the intermediate options, providing the whole algorithm

with a balance between intensification and diversification.

- Finally, the natural extension of any constructive heuristic is the improvement of the generated solutions using local search procedures. Iterated Greedy includes the improvement phase when the reconstruction phase is performed, trying to reach the local optimum of the neighborhood that contains the new generated solution.

Some examples of successful applications of Iterated Greedy in different problems can be found at [111, 112, 113, 114].

## 4.4 Path Relinking

Path Relinking (PR) methodology was originally proposed as an strategy that combines diversification and intensification processes in the Tabu Search [99] context. It is a relatively novel proposal and is still being developed. PR is established formally and methodologically in conjunction with Scatter Search [115], another extended metaheuristic that has been successfully applied to different optimization problems. The operation principle followed by Path Relinking is that it generates new solutions by exploring trajectories in the solutions space. To do this, it starts by selecting two high-quality solutions (origin solution and destination solution). Then, a path is generated from origin to destination, traversing the search space. Combination Method in Path Relinking is based in the trajectory generation among solutions in the search space, instead of carrying out linear combinations among them as it is done in Scatter Search. The way in which the path between the origin solution ($S_1$) and destination solution ($S_2$) is built is by performing movements that tries to reach $S_2$ from $S_1$. The trajectory $S_1 \rightarrow S_2$ is generated by gradually removing the attributes of the starting solution to introduce attributes that belong to the guiding solution. The goal is to find a better solution than $S_1$ and $S_2$ in the built trajectory between them.

To provide a more specific example of how Path Relinking works with a pair of solutions (origin solution and guiding solution), Figure 4.3 shows how the metaheuristic would behave in a context where a minimization objective function is being optimized. In this figure it can be seen how the trajectory between the origin solution and the guiding one is built, traversing the search space and, eventually, finding a solution that provides an objective function value better than the two from which it is based.

Path Relinking is a metaheuristic that allows modifications in their procedures. Some of the modifications that have improved the performance of the metaheuristic for certain problems are:

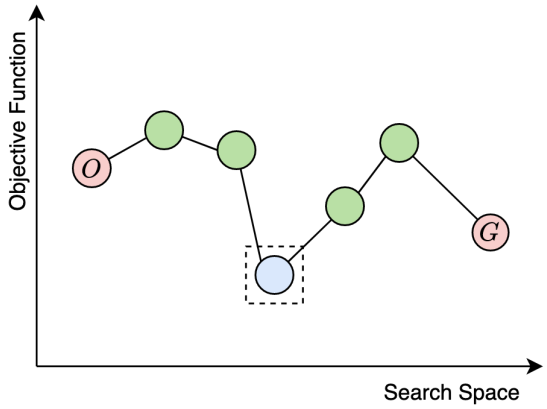



Figure 4.3: Specific example of Path Relinking behavior for a concrete pair of solutions.

- Variation: it generates simultaneously two different trajectories. One of them starts from the origin solution and tries to reach guiding solution and the other one performs the opposite movement: it tries to reach origin solution starting from the guiding one. Both trajectories stop in an intermediate point, and the final trajectory is built from these two reached intermediate points. This modification introduces more diversification to the metaheuristic, allowing to explore a larger region of the search space.

- Tunneling: this modification allows the variation in the neighborhood structure. By doing this, solutions that would not be explored in the original proposal, since the solutions are limited to a single neighborhood, are taken into account. For example, solutions that does not belong to the feasible region could be taken into account, but the search can not be get stuck in a non-feasible solution, given that guiding solution is always feasible.

- External Relinking: this method generates trajectories between two solutions by extending the path that connect them. To do this, the path is constructed by trying to move the two solutions away from each other. To do this, the procedure eliminates components common to the source solution and the guiding solution and replaces them with other components, arriving at different solutions at the end of the process.

- Multiple parents: this technique uses a guiding solutions set instead of only one. In this context, the next movement from the origin solution it is determined by the influence of all the components that are present in the parents, in such a way that those parents with a more suitable component regarding the objective function will guide the trajectory building.

- Constructive neighborhoods: it starts with an incomplete starting solutions (even an empty one) to generate a trajectory influenced by a set of solutions. The solution is built by making movements to the guiding solutions, progressively introducing their elements. In general, the inclusion of features in the solution is performed using a vote mechanism.

Examples of applications of these modifications can be found at [116, 25, 117, 118, 119].

## 4.5 Multi-objective approaches

Real-life problems are usually not single-objective, but involve more than one objective simultaneously. For example, companies engaged in freight transportation will usually want to maximize their profit, but they will also want to minimize the maintenance cost of their trucks, the miles an employee traversed reduces in a given period of time, or maximize the achievable sales area. Typically, several (if not all) of these objectives will be in conflict. In other words, improving the value of one objective will necessarily mean worsening the value of another one. This situation is known as multi-objective optimization, and it is present in multiple knowledge areas, such as logistics [120], engineering [121], or economics [122].

When two or more conflicting objectives are taken into account at the same time when solving a problem, the concept of *nondominance* appears. This term was introduced by Vilfredo Pareto (1848–1923) in the economics field, and it is also known as Pareto optimality [123]. A solution is considered nondominated (or Pareto optimal) if it is not possible to improve the value of one objective function without making one of the others worse. For the sake of clarity, the improvement of Pareto, the dominance of Pareto, and the Pareto optima terms are defined:

- Given an initial solution, a Pareto improvement is a new solution in which some objectives are improved and none gets worse.
- A solution is Pareto dominated if there exists a possible Pareto improvement
- A solution is a Pareto optimum if it does not exist any change that could drive to a better value in any objective without making another objective worse.

Given these definitions, the Pareto front is the set of all Pareto optimal solutions. Depending on the problem that is being solved, could be an infinite number of Pareto optimal solutions or could be only one. Typically, the definition of Pareto efficiency is relaxed with the aim of obtaining a more diverse set of solutions. In this context, a Pareto improvement is considered when a solution is

better than one that is present in the front under construction in, at least, one objective function value.

Then, when a multi-objective optimization [124] problem is tackled, a good approach is to slightly change the concept of solution, taking into account the whole front of non-dominated solutions as the solution to the problem instead of a single-solution. By doing it, a solution from the front could be selected depending on the specific necessities of the problem under being solved. To do this, is important to keep the front updated. It is, when a new solution is generated (using some constructive heuristic, for example), it is compared with all the solutions that are present in the non-dominated front. If the new solution dominates a solution that is already present at the front, then it will be included in the front, and all solutions that are dominated by the new solution will be removed from it. Figure 4.4 shows an example of a non-dominated front (in red colour) and some dominated solutions (in grey colour). The x-axis represents the values of each solution for a maximization objective and the y-axis represents the values of the same solution for a minimization objective that is in conflict with the first one.

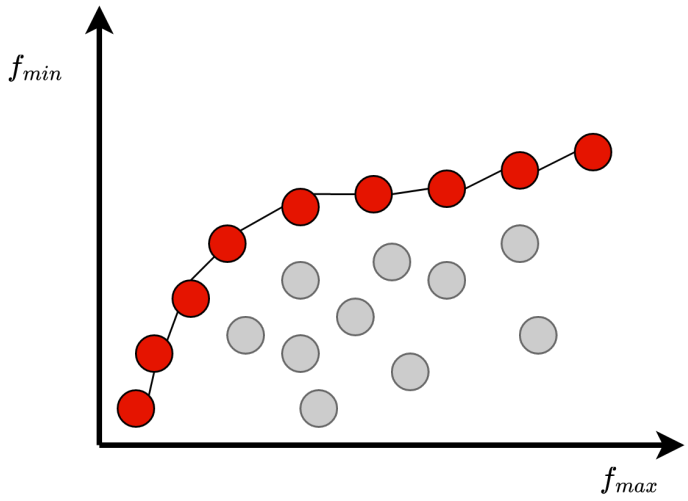



Figure 4.4: Example of a non-dominated front and some dominated solutions.

When non-dominated fronts are applied to solve a multi-objective problem, different metrics are used to evaluate the quality of the obtained fronts. These metrics provide an idea of how well-built is the front, based on their characteristics (number of solutions that are present in the front, size of the area under the curve generated by the front, etc.). Some of the most extended metrics in the multi-objective context are the following:

- Coverage metric. This metric, $C(F_1, F_2)$ evaluates the number of solutions

of the non-dominated front $F_2$ that dominates the solutions from the front $F_1$. When comparing two different fronts, a good approach is to generate a reference front $R$ which is conformed for the non-dominated solutions that are present in both fronts under evaluation (i.e., the front constructed by the union of all the non-dominated fronts under evaluation). Therefore, an smaller value of coverage metric represents a better value, given that it means that the reference front is dominating less solutions at the front under comparison.

- Hypervolume. This metric, $HV(F)$ evaluates the size of the space covered by the $F$ set of solutions. More specifically, it measures the hypervolume of the space region that is weakly dominated by a non-dominated front. Then, a greater value of $HV(F)$ implies that $F$ is better, given the more space covered by the front, the better.

- $\epsilon$-indicator. This metric, $EPS(F, R)$ sizes the smallest distance that is required to transform every solution present in the non-dominated front under evaluation ($F$) in the closest point of the reference front ($R$). It is a metric equivalent to the coverage one. Then, obtaining lower values of $\epsilon$-indicator implies that the non-dominated front under evaluation is better than others.

- Inverted Generational Distance plus. This metric, $IGD+(F, R)$, is evaluated as the inversion of the well-known generational distance metric. It sizes the distance from the non-dominated set under evaluation ($F$) to the reference set ($R$). Then, the lower value of $IGD+$, the better, given that it means that the evaluated front is closer to the reference front.

As it has been aforementioned, it could exist multiple Pareto optimal solutions to multi-objective optimization problems, which means solving such a problem is not as direct as it is for a typical single-objective optimization problem. Using Pareto front as solution is not the unique existing technique to tackle this kind of problems. Another widely used approach is to convert the original multi-objective problem into a single-objective optimization problem. This is what is known as a scalarized problem. When scalarizing a problem, it must be formulated in such a way that optimal solutions to the single-objective optimization problem are Pareto optimal solutions to the multi-objective version [125]. Scalarization techniques that are commonly used to tackle a multi-objective problem as a single-objective one are:

- Linear scalarization [126, 127]. By using this technique, all the implied objectives in the multi-objective problem are weighted and expressed as a single-objective function. By using this mechanism, the relevance of each objective must be decided and expressed by the associated weight.

- $\epsilon$-constraint method [128]. In this scalarization method, one objective is selected as the objective function of the single-objective model, while the rest are set as constraints of the problem, so that the values associated with these objectives must be strictly smaller (or larger, if we are solving a maximization problem) than a defined epsilon.

Examples of successful applications of linear scalarization can be found at [129, 130, 131]. $\epsilon$-constraint methodology has also been used to solve several multi-objective problems [132, 133].

# Chapter 5

# Joint discussion of results

*In this chapter, a discussion of the obtained results for all the optimization problems tackled in this dissertation is presented. A brief summary of the different proposals is also exposed. For each solved problem, the results summarize the average value for the objective function taken into account when a single-objective problem is tackled and widely used multi-objective metrics average when a multi-objective problem is being summarized. Also, other metrics that justify the quality of the proposal (as CPU time required by the algorithm) are presented. In all tables, the best found results are highlighted in bold font. In Section 5.1, the results for a first approach to community detection are exposed. Section 5.2 summarizes the obtained results when solving the classical CDP. The results for the multi-objective CDP are shown in Section 5.3, while results for the overlapping CDP are illustrated in Section 5.4. Finally, in Section 5.5 the obtained results for the multi-objective Dynamic Communtiy Detection Problem are presented. These results are extracted from the publications that can be found at Part II of this Thesis.*

## 5.1 Results on the Alpha separator problem

As it has been aforementioned, a good starting point to solve community detection problems is to find cliques in the network under study [46, 47, 48]. Another good approach is to detect which are the critical nodes whose removal produces the network division in different connected components, generating initial communities. This problem is an $\mathcal{NP}$-hard problem by itself, named $\alpha$-separator problem, and it is addressed in [134]. The objective function of this problem is to find the minimum separator that divides the given network in connected components with a size lower than $\lceil \alpha \cdot n \rceil$ nodes, where $n$ is the total number of nodes in the network. The results on the $\alpha$-separator problem have been published in a JCR journal;

specifically, it has been published in Expert Systems, a journal with an impact factor of 2.587 and situated in Q2 of JCR. More details about the journal can be found at Chapter 7 in Part II. In this work, an algorithm based on the Greedy Randomized Adaptive Search (GRASP) combined with Path Relinking metaheuristic is proposed. In this proposal, GRASP is used as the constructive metaheuristic following two different approaches: on the one hand, a *Greedy Random* approach, where the first element that is part of a solution is selected at random and the restricted candidate list is built following a greedy criterion. On the other hand, a *Random Greedy* approach, where the first element in the solution is selected in a greedy way, while the restricted candidate list is randomly built. The greedy criterion used is a well-known metric widely used in Social Network Analysis context: the closeness. In the *Greedy Random* version, the first node added to the separator is selected at random, with the aim of diversifying the search procedure. The candidate list is built using the closeness metric for each node, and the restricted candidate list is built with the nodes that have associated a large value of this function. Then, the next vertex to be added to the separator is randomly selected from the restricted candidate list. The *Random Greedy* approach works in a similar way, but the greedy and random phase are exchanged: the RCL is constructed with a set of elements randomly selected from the candidate list and, then, the next element included in the solution is selected in a greedy manner. Regarding at the computing time required by each algorithm, the *Random Greedy* version is faster than the *Greedy Random*, because it does not require an evaluation of the whole candidate list. Also, as it has been experimentally proven, it provides better results in average than the *Greedy Random* approach.

In the improvement phase, a local search procedure is defined to find a local optimum with respect to a predefined neighborhood. The neighborhood for this local search is the one defined by all the solutions that can be generated by applying a movement in which two vertices are removed from the solution and a new one is added. With respect to the order in which the neighborhood is explored, a *First Improvement* method is applied. There are two main reasons for this decision: the first one is that, normally, a *First Improvement* approach is so much less time-consuming than the *Best Improvement*, given that the former does not traverse the entire neighborhood, but it only performs the first improving move found. The second one is that, when a successful movement is performed, it necessarily implies an improvement in the objective function, given that a minimization problem is being solved, and the local search performs movements in which two nodes that are present in the solution are removed and substituted by only one. The local search method proposed randomly explores the neighborhood, increasing the diversification of the search. This phase stops when no improvement is found after exploring the complete neighborhood.

The GRASP procedure generates a set of diverse and high quality solutions, and an Elite Set is constructed with the most promising ones. Once the Elite Set is generated, Path Relinking is applied with the aim of exploring the trajectories between each pair of solutions included in it. This is known as static design. In a dynamic design, the Elite Set is updated each time a trajectory is explored. In this problem, the trajectory between two solutions is generated by performing a *Swap* movement. This movement removes a node that is present in the initial solution but is not included in the guiding one, and replaces it by one present in the guiding solution and not yet added to the initial separator. Through these exchanges, the initial solution approaches the guiding solution, until the two solutions end up being exactly the same. In each iteration, a new intermediate solution is generated. This intermediate solution will be infeasible with a high probability, so it must be repaired to make it feasible again. In this work, a repairing process is performed by randomly adding nodes to the separator until reaching the feasibility again, increasing the diversification of the search. Once the repair has been performed, the separator could contain redundant nodes (i.e. nodes that are not needed anymore to have a feasible solution), so the solution is traversed looking for redundant nodes that are removed.

Typically, to select the next node that is being removed from the separator, two different Path Relinking strategies exist. On the one hand, Random Path Relinking (RPR) generates the next solution in the path by randomly selecting one of the *Swap* moves in the trajectory. On the other hand, Greedy Path Relinking (GPR) selects the best *Swap* movement available. In this work, a third variant is proposed: *Greedy Randomized Path Relinking* (GRPR). In this strategy, RPR and GPR strategies are mixed, in such a way that all possible intermediate solutions are generated (as it occurs in GPR), but it selects one of the best solutions instead of the best one, in a similar manner that GRASP does. Finally, a new PR strategy, named Exterior Path Relinking (EPR) is presented. This version, originally proposed in [135], tries to reduce the generation of short trajectories generated by the classical approach by removing from the initial solution elements that are present in the guiding one. By doing this, the initial solution is getting farther to the guiding one in each iteration. This strategy ends when the initial solution does not have any common component with the guiding one. It is important to remark that solutions obtained by any of these methods are not necessarily a local optimum with respect to the neighborhood under exploration, so the aforementioned local search procedure is applied to the obtained solutions.

The computational experiments were performed over a set of 50 instances extracted from Erdös-Renyi model. These graphs are built in a way in which each new inserted node has the same probability of being connected to any existent node

in the graph. The preliminary experiment is in charge of analysing the performance of the proposed GRASP procedures and selecting the best variant, and do the same for the different local search procedures and Path Relinking proposed versions. The final experimentation analyses the contribution of the proposal by comparing it with the state-of-the-art method found in the literature, an algorithm based on Random Walks (RW). Table 5.1 shows the results comparing the average of objective function value, the required computing time, the deviation percentage when the best solution is not reached and the times that an algorithm finds the best solution.

| Algorithm | Avg. | Time (s) | Dev. (%) | #Best |
|---|---|---|---|---|
| GRASP | 63.18 | **137.41** | 5.90 | 14 |
| GRASP+PR | **62.00** | 822.29 | **3.68** | **34** |
| RW | 71.78 | 1070.36 | 18.26 | 18 |

Table 5.1: Final comparison between GRASP, GRASP+PR, and the best previous method found in the state of the art.

The main conclusion obtained from these results is that the followed approach, based on high-diversification, allows to find high-quality solutions in a low computing time. In this work, the best version of the proposed algorithm is the one that combines the Exterior Path Relinking with the *Random Greedy* version of GRASP with an $\alpha$ value of 0.25. This approach is compared with the Random Walks-based algorithm, clearly showing a superiority in terms of objective function values and required computing times. These results are supported by a non-parametric statistical test. Specifically, the pairwise Wilcoxon Signed Rank Test has been performed, with a resulting $p$-value smaller than 0.001, indicating that there exists statistically significant differences between the proposal and the state-of-the-art algorithm, with a significance level of 95%. More specifically, the proposal is able to reach 34 times the best solutions for the set of 50 instances, having a low value of deviation with respect to the best value found when it is not able to find it (3.68% of deviation in average).

## 5.2 Results on the Classical Community Detection Problem

The classical Community Detection Problem (CDP) is studied in [136] (Chapter 3.3, Part II) using a GRASP approach. The proposed algorithm exploits the diversification versatility provided by GRASP to reach high-quality solutions. To

do this, in the constructive phase of grasp, an agglomerative approach is followed. The algorithm starts by locating each node in a single community conformed by itself. Then, the candidate list is generated with all the communities present in the network. The RCL is built using the modularity as objective function, and it is ordered in a decreasing order regarding to this value. Once the candidate list and the restricted candidate list are built, a random community is selected from the RCL. Then, the algorithm looks for the best community to be joint with the first one. To decide what is the best community to be joint, a simulation of the fusion of the candidate community with all the others is performed. Then, the candidate community and the best one found are merged. If this movement results in an improvement in the modularity, then the new community becomes a new candidate and the current solution is updated. Otherwise, the candidate community is removed from the candidate list, given that it cannot be merged with other community without making worse the objective function. When there is not more possible merge movements, the construction phase is over.

In the local improvement phase of the algorithm, a local search procedure is proposed. In this procedure, the neighborhood that is explored is conformed by all solutions that can be reached by performing a movement that removes one node from its current community and inserts it in a new one. It is important to remark that this movement could lead to make a community empty if all its nodes are removed. In the same way, a new community could be generated if this movement improves the modularity of the general solution. This situation causes that, after the local search, a solution could contain a different number of communities (greater or lower). In order to decide which node is going to be removed from its corresponding solution, a heuristic criteria is applied. Concretely, the percentage of intra-community edges with respect to the total number of edges in the graph for each vertex is evaluated. Then, the algorithm selects the vertex with a lower value of this metric, and it is moved to the community that maximizes the modularity. The proposed local search follows a first improvement approach, restarting the search when the first improvement of the neighborhood under exploration is found, and stopping it when an improvement is found.

The computational experiments have been performed over a set of 100 instances extracted from the Twitter SNAP dataset and from Network repository. The experimental phase is divided in two different phases. The first one is devoted to tuning the $\alpha$ value for the GRASP procedure. This experiment is performed over a subset of 20 instances and it indicates that the selection of a random value of alpha in each iteration of the algorithm results in the best configuration for the algorithm.

The final experimentation has been carried out to compare the proposal

with a set of well-known classical algorithms in the community detection area. Concretely, it has been compared with Edge Betweenness (EB), Fast-Greedy (FG), Label Propagation (LB), Multi-level (ML), Walktrap (WT), InfoMap (IM) and Louvain (CL) algorithms. Table 5.2 shows the average modularity and conductance values for each algorithm over the whole instance set. It is important to remark that, in this work, the reported value is the opposite of conductance, evaluated as $1 - C_o(S, G)$, with the aim of having a direct comparison with modularity metric.

| | Modularity | | Conductance | |
|---|---|---|---|---|
| | Avg. | #Best | Avg. | #Best |
| EB | 0.20176 | 0 | 0.03363 | 7 |
| FG | 0.29441 | 3 | 0.44062 | 17 |
| LP | 0.15170 | 2 | 0.43734 | 6 |
| ML | 0.28843 | 2 | 0.43433 | 19 |
| WT | 0.26663 | 2 | 0.25224 | 7 |
| IM | 0.20611 | 2 | 0.37829 | 16 |
| CL | 0.31181 | 33 | 0.48002 | 9 |
| *GRASPAGG* | **0.31331** | **78** | **0.49483** | **37** |

Table 5.2: Comparison of the considered metrics over classical algorithms and the proposal.

The first conclusion to be drawn is that the proposal, despite being the combination of two simple heuristics, its powerful diversification and fast computation allows obtaining high quality and very competent solutions with respect to classical community detection algorithms. Furthermore, the balance between greediness and randomness in the constructive procedure, allows to the local search procedure perform the intensification phase in a widely region of the solutions space, obtaining good results thanks to the problem-based neighborhood definition.

The results are supported by non-parametric statistical tests. In particular, the Friedman test and Wilcoxon test were performed. The Friedman test ranked the compared algorithms from 1 (best algorithm) to $n$. The proposal and Louvain algorithms obtained the first two positions in the rank. Both statistical tests resulted in a $p$-value smaller than 0.00001, confirming that they are statistically significant differences among the algorithms. These results have been published in the Electronics JCR Journal, situated at third quartile (Q3) with an impact factor of 1.764.

## 5.3 Results on the multi-objective Community Detection Problem

The community detection problem is tackled from a multi-objective point of view in [137]. The full text can be found in Chapter 9, Part II. In this work, a Variable Neighborhood Search methodology has been used. Specifically, a Basic VNS (BVNS) algorithm has been implemented. Given that VNS framework was originally designed to solve single-objective optimization problems, it must be adapted for the multi-objective scenario [138]. In this work, this adaptation is performed by considering the non-dominated front of solutions instead of a single one as the solution that must be returned by the algorithm. The BVNS algorithm runs until the maximum neighborhood set as parameter is explored. In each iteration, the shake, improvement and neighborhood change procedures are executed, updating the non-dominated set of solutions.

Given that BVNS requires from an initial solution to work (i.e. a populated front of non-dominated solutions), it must be built in some way. For this problem, this initial set is built following a GRASP procedure. In this work, only the constructive phase of GRASP is taken into account. In this phase, the initial solution is conformed by $n$ communities, where $n$ is the number of nodes in the graph. The candidate list is built by assigning a greedy function value to each community present in the solution. Then, communities are merged in pairs, by randomly selecting an element from the RCL and merging it with the best existing community in the incumbent solution. To select the best community to join with, the selected greedy function corresponds to the ratio between intra-community edges and the community edges (intra-cluster and inter-cluster) that would be if two communities were merged. This methodology allows to generate a populated non-dominated set of solutions. It is important to remark that not all built solutions are included in the reference front, but only these that are non-dominated ones.

Regarding at the shake procedure, in the proposal it consists of a movement in which a single node is removed from its current community, inserting it in a different one selected at random. Following this procedure, obtained solutions will probably be worse than the original solution in terms of quality, but it is important to remark that the main objective in shake procedures is to escape from local optima. In this phase of the algorithm, what is sought is diversification over intensification. Given the nature of the problem, it is not necessary to check the feasibility of the resulting solution, because it is guaranteed that a single node will be assigned to at least and only one community when the shake procedure finishes.

With the aim of improving the quality of perturbed solutions, and, therefore, focusing on the intensification of the solutions (i.e. trying to reach a local optimum), a local search procedure is applied to any perturbed solution. In this work, the improvement procedure takes a set of perturbed solutions as input and returns a non-dominated front with all local optimum reached starting from the perturbed set. Two different local improvement methods are studied in this work, both of them following a first improvement approach. On the one hand, the first proposed method improves each objective function under evaluation independently. On the other hand, the second one tries to optimize both strategies simultaneously, considering alternatively one of them in each iteration of the procedure. Again, any improved solution is tried to be included into the non-dominated front. The movement that defines both local search procedures is the same: a node belonging to a community is tried to be included into a different one. Then, the new solution is tried to be added to the reference front. If it can be included, it means that it dominates at least one solution that is present in the front, so an improvement has been found. The procedure finishes when the reference front is not updated after a complete execution of the local search.

Once a high-quality non-dominated set is generated, the neighborhood change method is executed. In this work, this procedure has been adapted to suit the multi-objective scenario. The main adaptation performed is the modification of the improvement concept. In this context, an improvement is considered when the non-dominated solutions front is updated. This update indicates that a solution that dominates one present in the front has been found. Therefore, the algorithm restart the search from the initial neighborhood if an improvement has been performed. Otherwise, the next neighborhood is explored, until the maximum neighborhood, set as an input parameter, is reached.

Regarding to the computational experiments, a set of 52 synthetic networks and 12 real-world networks have been used. In this paper, two different kind of metrics have been evaluated in order to test the robustness of the algorithm. On the one hand, well-known multi-objective metrics have been studied. On the other hand, the Normalized Mutual Information (NMI) and the modularity context-based metrics have been taken into account. The experimentation have been divided into an preliminary and final phases. The former is devoted to adjust the parameters in the GRASP algorithm, the best configuration of the local improvement method and the $k_{max}$ parameter for the BVNS. The latter is performed to compare the proposal against the best algorithm found in the state of the art. Table 5.3 shows the comparison with respect to the multi-objective metrics. In this table, it can be seen that the quality and computation time of the proposal overcomes the best method found in the literature. In Table 5.4 the results regarding

to context-based metrics are shown.

| Algorithm | *CV* | *HV* | *EPS* | *IGD+* | $T(s)$ |
|---|---|---|---|---|---|
| MOBVNS | **0.07** | **0.14** | **0.86** | 0.22 | **214.64** |
| LMOEA | 0.36 | 0.02 | 0.27 | **0.21** | 1800.00 |

Table 5.3: Comparison of the reference front obtained with the best configuration for MOBVNS and the LMOEA proposed by [1]. Best results are highlighted with bold font.

| | LMOEA | | | |
|---|---|---|---|---|
| Instance | Avg. NMI | Avg. Time (s) | Best NMI | Best Time (s) |
| dolphin | 0.05 | 1800 | 0.069 | 1800 |
| footbal | 0.02 | 1800 | 0.033 | 1800 |
| karate | 0.1 | 1800 | 0.1 | 1800 |
| | MOBVNS | | | |
| Instance | Avg. NMI | Avg. Time (s) | Best NMI | Best Time (s) |
| dolphin | **0.751** | **0.41** | **0.77** | **0.11** |
| footbal | **0.864** | **1.8** | **0.877** | **0.27** |
| karate | **0.439** | **0.07** | **0.439** | **0.03** |

Table 5.4: Summary of the results of NMI metric obtained by the proposed MOBVNS and LMOEA when solving the real world instances.

It can be concluded that combining the GRASP constructive procedure with the VNS framework is a powerful mechanism to solve multi-objective problems. It is important to remark that the VNS framework requires to be adapted in order to successfully manage the multi-objective scenario. More specifically, regarding to the classical multi-objective metrics, it can be seen that MOBVNS proposal is better in all of them, except in *IGD+*, where it is slightly higher than LMOEA. Looking at the average computing time, the proposal takes approximately nine times less time than the previous algorithm. Looking at real-time performance, the proposal is able to reach higher values of NMI in four orders of magnitude less time. These results have been published in Applied Soft Computing, a Q1 journal in JCR with an impact factor of 6.725.

## 5.4 Results on the Overlapping Community Detection Problem

This work is currently under review in Journal of Heuristics. Even the results are still in process of being published, it is interesting to discuss the results obtained from the research, as well as the conclusions extracted. In this work, an Iterated Greedy metaheuristic is proposed to solve the CDP in its overlapping variant. To generate the initial solution for the framework, a GRASP metaheuristic has been applied. For the construction phase, an initial solution without communities is generated. Then, the candidate list is built including all nodes in the network, and the algorithm executes until all nodes have been assigned to, at least, one community, i.e., until the CL gets empty. To give a score to the candidates and construct the restricted candidate list, a greedy function is used. More specifically, the PageRank metric is used. This metric was originally proposed with the aim of evaluating the importance of a web page on the Internet, based on the links that are referring to it in the whole network. This idea can be applied in the CDP, given that a node that is connected to many others is a relevant node in the network and, predictably, it will be a good candidate to start building a community. Then, a candidate is randomly selected from the RCL and added to the community under construction. To decide which nodes are going to be added to the same community, a dynamic membership function algorithm is applied [139]. This method is a good approach for the OCDP, given that a node can be included in different communities if it satisfies the following membership criterion: a node is included into a community if it improves the ratio between intra-community and inter-community edges of the incumbent community. Those nodes that are included into the community under construction are removed from the CL to avoid getting them considered as starting points for a new community.

Once a solution has been built, the improvement phase of GRASP is applied. In this work, the move operator that generates new solutions (that conforms the neighborhood under exploration) consists in assign a node to a new community or remove it from its currently assigned community. A node will be moved depending on whether the subtraction between the number of edges towards a community other than its own divided by its degree and the number of edges towards its community divided by its degree is greater than a threshold defined as a parameter of the algorithm. If it occurs, then the node is removed from its current community and moved to the new one. If not, then the node is added to the new community, producing the overlapping situation. In this proposal, a first improvement strategy is followed with the aim of reducing the computing time of the algorithm. The local search procedure ends when all solutions belonging to the current neighborhood

are worse than the current one in terms of the objective function, i.e., when a local optimum is found.

Once an initial solution has been constructed through GRASP, Iterated Greedy metaheuristic framework is applied. In the destruction phase, a percentage of nodes of the solution are unassigned from the communities that they belong to. This percentage is experimentally defined. To decide which nodes must be removed from the solution, two different strategies has been followed. On the one hand, a random destruction is performed. In this version of the destruction, the nodes that will be removed from the current solution are randomly selected. On the other hand, a greedy strategy is followed. Specifically, the nodes with a larger ratio of inter-cluster edges with respect to their degree are removed from their communities. Both of them result in an infeasible solution, given that some nodes are not assigned to any community when the procedure ends.

To retrieve the feasibility of the destroyed solution, the reconstruction phase is executed. To do this, the unassigned nodes must be re-assigned to, at least, one community in the network. In this phase of the algorithm, nodes will be only assigned to a single community. The decision of add a node to more than one community is responsibility of the local search procedure, that is going to be applied after the reconstruction. To assign the nodes to a community, again, two different strategies have been followed: re-assign the nodes to a random community and the opposite strategy to that followed in the destruction phase, select the community that maximizes the ratio of intra-cluster edges with respect to its degree as the most suitable community for a node. Finally, the same local search procedure applied in the improvement phase of GRASP is applied after the reconstruction phase, with the aim of finding a local optimum in the solutions space region under exploration.

Computational experiments have been performed to evaluate the quality of the proposal. A set of 68 instances has been used. Preliminary experimentation is devoted to adjust all the parameters of the algorithm: the $\alpha$ value of GRASP and the percentage of destruction for Iterated Greedy. In addition, this experimentation is useful to select the best configuration for the construction and destruction strategies. As a result of these experiments, the best configuration of the algorithm is obtained. In the final experimentation, the algorithm is compared against the best algorithm found in the literature. To evaluate the quality of solutions an adapted version of modularity metric to the overlapping scenario has been used. Table 5.5 shows the superiority of the proposal in a set of instances with different size ($n$ represents the number of nodes present in the network).

These results show that the proposal is, in average, a better option for the

| | Iterated Greedy | | | | EADP | | | |
|---|---|---|---|---|---|---|---|---|
| | Avg. | Dev (%) | Time (s) | #Best | Avg. | Dev (%) | Time (s) | #Best |
| $0 \leq n < 2500$ | **0.319** | **3.399** | 5.501 | **13** | 0.216 | 34.855 | **0.507** | 2 |
| $2500 \leq n < 5000$ | **0.383** | **0.000** | 27.736 | **15** | 0.238 | 37.061 | **3.243** | 0 |
| $5000 \leq n < 7500$ | **0.377** | **0.000** | 48.293 | **18** | 0.236 | 37.102 | **10.781** | 0 |
| $7500 \leq n \leq 10000$ | **0.377** | **0.000** | 87.124 | **9** | 0.234 | 37.171 | **36.943** | 0 |
| *Average* | **0.364** | **0.850** | 42.164 | **55** | 0.231 | 36.547 | **12.869** | 2 |

Table 5.5: Comparison of Iterated Greedy (IG) and EADP configured as stated in [2].

evaluated networks. Although the computational time required by the EADP algorithm is less than that required by the proposal, the solutions found by Iterated Greedy outperform the previous algorithm. In addition, when Iterated Greedy is not able to find the best solutions, it has a low percentage of deviation in average (3.399% for the set of instances in which it is not able to find the best solution, 0.850% in average for the whole set of instances). The algorithm finds the best solution a total of 55 instances, while the state-of-the-art method reaches the best solution 2 times. Regarding at computing times, Iterated Greedy only takes an average of 30 seconds more than EADP, so the improvement in the solutions quality justifies the interest of the proposal.

The results derived from this research are currently under review in Journal of Heuristics, a Q2 JCR journal with an impact factor of 2.247.

## 5.5 Results on the multi-objective Dynamic Community Detection Problem

The paper that summarizes the research concerning this problem is under second review in the Expert Systems With Applications journal. Nevertheless, it is interesting to expose the strategies followed to solve it. In this work, a population-based metaheuristic is adapted to solve the multi-objective Dynamic Community Detection problem: Scatter Search. This framework has been slightly modified with the aim of facing the multi-objective nature of the problem that is being solved. The main modifications have been performed in the improvement method, the *RefSet* update method, and the Subset Generation Method. In the improvement method, a solution is improved in two different ways, each one using as objective function one of the optimization metrics of the problem: *ICS* (Inverted Community Score) and *AVG_ODF*. The *RefSet* update procedure is modified in such a way that two *RefSets* are maintained, one considering the *ICS* objective and the other one considering *AVG_ODF*. The subset generation method is adapted

to reduce the number of pairs that are combined, by dividing the *RefSet* in four different subsets: the one that contains high-quality solutions regarding to *ICS*, the one that contains high-quality solutions regarding to *AVG_ODF*, and the two containing different solutions with respect to each of the objective functions. It is important to remark that all generated solutions are tried to be included in the non-dominated solutions set.

With the aim of populating the *RefSet*, high-quality solutions and diverse solutions are generated and included in the set. In the context of MODCDP, the distance between two solutions is calculated as the sum of nodes that are located in different communities in the solutions under comparison. This is the metric used to generate the diverse *RefSet* subsets.

In the Diverse Generation Method, a greedy randomized approach has been followed. Based on the idea that finding a good community structure is a computationally demanding task, an algorithm based in the McAllister work [140] is applied. More specifically, a modified Breadth-First Search algorithm is applied. The modification takes into account the McAllister function value associated to each node in order to decide when the construction of the current community must be stopped and a new one should be built. McAllister function is used because it gives a priority to be joint to the current community to those nodes that have more labeled neighbors, it is, more neighbors added to the community under construction. This procedure ends when all nodes have been assigned to a community.

Once a diverse solutions set has been built, an improvement procedure is executed with the aim of finding a local optimum with respect to a certain neighborhood. A local search procedure is defined to reach this goal. The neighborhood that will be explored is composed for all possible solutions that can be reached by changing a node from its original community to a different one. To decide if a node is a promising candidate to perform the movement, the number of intra-community and inter-community edges is evaluated. If a node has more inter-community edges than inter-community ones, it means that it could be assigned to a better community. The destination community is selected by evaluating the number of adjacent nodes that the removed nodes has in the community under evaluation. Performing this movement, the quality of community structure will be improved, since a node will always be reassigned to a community with more intra-cluster edges than it had in its original assignment. A best improvement strategy has been followed to traverse the aforementioned neighborhood, since the efficient way in which it has been implemented allows a complete exploration of the neighborhood without requiring a high computational time.

In the combination method, a Path Relinking approach is applied to each

pair of solutions derived from the *RefSet*. More specifically, a solution belonging to the high-quality subset of *RefSet* for one objective function is combined with one that belongs to the diverse subset of *RefSet* for the same objective function. This process is applied for both objective functions. To create a path between an initial solution and a guiding solution, elements that are included in the guiding one and that are not present in the initial solution, are added to the initial one. In this work, a Random Path Relinking approach is followed, it is, the node whose assigned community will be changed is randomly selected.

It is important to remark that there exists two different approaches to solve the dynamic variant of the CDP. On the one hand, each solution can be generated from scratch for each snapshot of the network. On the other hand, the solution generated to one snapshot can be used as the starting point for the next one. In this work, both approaches have been tested, emerging the second strategy as the best one.

In the computational experimentation, both classical multi-objective metrics and context-based metrics have been studied. Specifically, Coverage, Hypervolume and Inverted Generational Distance + have been selected as multi-objective metrics and modularity has been selected as context-based metric to evaluate the quality of the solutions. A set of 69 synthetic and real-world instances has been used to test the algorithms performance. The experimental phase has been divided into two different set of experiments. The first one is devoted to select the best strategy (start from scratch in each snapshot or take advantage of the solution generated for the previous snapshot), and study the contribution of each part of the algorithm to the final solution. In final experiments, the proposal is compared against the best method found in the literature. Analyzing the results, the proposal demonstrated to be better in both synthetic and real-world networks. Table 5.6 shows the comparison between both algorithms regarding at the multi-objective metrics in synthetic networks. As it is shown, Scatter Search proposal demonstrates a superiority with respect to these metrics. More specifically, the Scatter Search algorithm is able to reach the lowest possible values for the CV and IGD+ metrics in all synthetic instances. Regarding to the HV, higher values than Immigrants are reached for all the snapshots in all instances.

The same occurs for the real-world instances evaluated (Table 5.7). As it can be seen, the Scatter Search based proposal obtains the best results in terms of the CV metric and has a significant difference in average for the other two multi-objective metrics studied.

Regarding at context-based metrics, the superiority of the proposal can be also confirmed looking at the modularity values reported in Table 5.8 and Table 5.9.

| Snapshot | Scatter Search Repairing | | | Immigrants MOGA | | |
|---|---|---|---|---|---|---|
| | AVG. CV | AVG. HV | AVG. IGD+ | AVG. CV | AVG. HV | AVG. IGD+ |
| Snapshot 0 | **0.00** | **0.62** | **0.00** | 1.00 | 0.08 | 1.60 |
| Snapshot 1 | **0.00** | **0.54** | **0.00** | 1.00 | 0.14 | 1.17 |
| Snapshot 2 | **0.00** | **0.63** | **0.00** | 1.00 | 0.11 | 0.23 |
| Snapshot 3 | **0.00** | **0.63** | **0.00** | 1.00 | 0.11 | 0.25 |
| Snapshot 4 | **0.00** | **0.61** | **0.00** | 1.00 | 0.12 | 0.28 |
| Snapshot 5 | **0.00** | **0.59** | **0.00** | 1.00 | 0.18 | 0.29 |
| Snapshot 6 | **0.00** | **0.53** | **0.00** | 1.00 | 0.14 | 1.13 |
| Snapshot 7 | **0.00** | **0.55** | **0.00** | 1.00 | 0.14 | 8.65 |
| Snapshot 8 | **0.00** | **0.49** | **0.00** | 1.00 | 0.17 | 0.39 |
| Snapshot 9 | **0.00** | **0.49** | **0.00** | 1.00 | 0.17 | 3.23 |

Table 5.6: Table comparing the obtained multi-objective metrics with Scatter Search and Immigrants MOGA algorithms in synthetic networks.

| Dataset | Scatter Search Repairing | | | Immigrants MOGA | | |
|---|---|---|---|---|---|---|
| | AVG. CV | AVG. HV | AVG. IGD+ | AVG. CV | AVG. HV | AVG. IGD+ |
| Travian Market | **0.00** | **0.70** | **0.11** | 0.84 | 0.15 | 0.61 |
| Travian Messages | **0.00** | **0.62** | **0.27** | 0.33 | 0.13 | 0.31 |

Table 5.7: Table comparing the obtained multi-objective metrics with Scatter Search and Immigrants MOGA algorithms in real-world networks.

More precisely, the Scatter Search algorithm obtains an average of modularity value one magnitude order higher than MOGA proposal in all the snapshots of synthetic instances. The same behavior can be seen in the real-world instances that have been used in the experimental phase.

| Snapshot | Scatter Search Repairing | Immigrants MOGA |
|---|---|---|
| | AVG. Mod | AVG. Mod |
| Snapshot 0 | **0.3485** | 0.0242 |
| Snapshot 1 | **0.4879** | 0.0177 |
| Snapshot 2 | **0.5313** | 0.0167 |
| Snapshot 3 | **0.5328** | 0.0191 |
| Snapshot 4 | **0.5354** | 0.0168 |
| Snapshot 5 | **0.5349** | 0.0136 |
| Snapshot 6 | **0.5223** | 0.0116 |
| Snapshot 7 | **0.5131** | 0.0045 |
| Snapshot 8 | **0.4449** | 0.0090 |
| Snapshot 9 | **0.3543** | 0.0678 |

Table 5.8: Table comparing the average modularity values obtained with Scatter Search and Immigrants MOGA algorithms in synthetic networks.

| Dataset | Scatter Search Repairing | Immigrants MOGA |
|---|---|---|
| | AVG. Mod | AVG. Mod |
| Travian Market | **0.0714** | 0.0065 |
| Travian Messages | **0.2179** | 0.0899 |

Table 5.9: Table comparing the average modularity values obtained with Scatter Search and Immigrants MOGA algorithms in real-world networks.

# Chapter 6

# Conclusions and future work

*In this Chapter the conclusions derived from the performed research are presented. The conclusions for each tackled variant of the problems are divided into sections. Additionally, a section of future works is included.*

## 6.1 Conclusions on the Alpha Separator Problem

The alpha separator problem has been studied as an starting point for the community detection problems. A GRASP based algorithm has been proposed to find critical nodes in networks, being critical nodes those whose removal implies the division of the network in $n$ connected components with a restricted size (given as parameter of the instance). In this work, the potential of GRASP coupled with a Path Relinking combination strategy has been proven. Specially, regarding to the computing time required by the algorithm to reach high-quality solutions, the proposal emerged as the best method in the state of the art.

In particular, the main contribution of this work relies in the experimental phase. Two different variants of GRASP using closeness metric as greedy function have been proposed: *RandomGreedy* and *GreedyRandom*, and the experiment shown that *RandomGreedy* strategy is more suitable for this specific problem. In addition, four different variants of Path Relinking have been proposed: *Greedy Randomized Path Relinking*, *Random Path Relinking*, *Greedy Path Relinking* and *Exterior Path Relinking*. The last one is the most novel approach, and also the best one for solving the alpha separator problem.

One paper derived from this work has been published in a Q2 JCR journal,

with an impact factor of 2.587 (JCR 2020): *Finding weaknesses in networks using Greedy Randomized Adaptive Search Procedure and Path Relinking* [134], included in Chapter 7 of Part II.

## 6.2 Conclusions on the classical Community Detection Problem

The main objective of the study of Community Detection Problem was to develop a fast algorithm that demonstrates a better performance than classical algorithms for community detection in networks. This goal was achieved with a GRASP methodology, using the well-known modularity metric as objective function. This metaheuristic has been proven to be a good starting point for solving community detection problems, given its capacity to balance the diversification in the construction phase and the intensification in the improvement phase, providing high-quality solutions in low computing times.

In this work, two heuristic strategies have been proposed: the agglomerative constructive procedure, that balances the greediness and the randomness of the search, and the improvement procedure, where a problem-based neighborhood has been successfully exploited to reach local optima.

The experimental phase allows to correctly adjust the GRASP $\alpha$ value, and avoid to overfit the algorithm by selecting a subset of the whole instances set. Additionally, the final experiment phase demonstrates that the objective of the research has been accomplished, comparing the proposal with seven well-known community detection algorithms.

The research on the CDP is presented in the paper named *On the Analysis of the Influence of the Evaluation Metric in Community Detection over Social Networks* [136], included in Chapter 8 of Part II. Specifically, this work is published in Electronics journal, a Q3 JCR with an impact factor of 1.764 (JCR 2018).

## 6.3 Conclusions on the Multi-objective Community Detection Problem

In this work a new metaheuristic method based on VNS for community detection is presented. To solve it, VNS has been adapted to suit the multi-objective scenario, considering a set of non-dominated solutions as a whole solution for the framework. To generate the initial set of non-dominated solutions, a GRASP methodology has been applied. This methodology has demonstrated to be a reliable and fast

technique to produce initial solutions in the context of CDP. Using GRASP allows the VNS framework to start the search from a promising region of the solution space.

In this work, Ratio Cut and Negative Ratio Association have been defined as objective function, trying to minimize both of them simultaneously. The evaluation of the obtained fronts has been performed using classic multi-objective metrics, as Coverage, Hypervolume, $\epsilon$-indicator and Inverted Generational Distance +.

The experimentation has demonstrated that combining GRASP with VNS results in high-quality solutions in a multi-objective context, and more specifically in the MOCDP. The efficient implementation of the algorithm and the quality of applied heuristics allows the algorithm to overcome the previous work.

The research on the MOCDP is exposed in the work named *A fast variable neighborhood search approach for multi-objective community detection* [137], included in Chapter 9 of Part II. More specifically, it is published in Applied Soft Computing, a Q1 JCR journal with an impact factor of 8.623 (JCR 2021).

## 6.4 Conclusions on the Overlapping Community Detection Problem

In this paper a new metaheuristic-method for overlapping community detection problem is proposed. Specifically, the proposal hybridizes GRASP and Iterated Greedy metaheuristics. The algorithm makes use of a modified version of the classical modularity metric adapted to the overlapping scenario. One of the main contributions of this work is the use of an intelligent move operator defined for the local search procedure. It allows to improve the quality of the solutions generated by the GRASP procedure, that makes use of the PageRank metric as greedy function. By using this local search procedure, the overlapping scenario is covered, generating solutions that allows having nodes assigned to more than one community simultaneously.

Another contribution is the development of two different strategies for the destruction and construction phases of Iterated Greedy, respectively, and their combination. A random and a greedy strategies for the construction phase have been proposed, as well as for the reconstruction phase. The experimental section allowed to select the best version of the GRASP constructive procedure and the best configuration for the local search. Also, the best combination of construction and destruction phases strategies has been set. Results show that the proposal provides a higher quality solutions than the state-of-the-art method, although

requiring a slightly longer computation time.

This work has been sent for its review to Journal of Heuristics, an impact journal indexed in the JCR (Journal Citation Reports), situated at the Q2 with an impact factor of 2.247.

## 6.5 Conclusions on the multi-objective Dynamic Community Detection Problem

In this work, a Scatter Search metaheuristic has been adapted to work in the multi-objective context. Furthermore, the greedy randomized heuristic developed allows to generate initial solutions for the Scatter Search framework in a fast way, combining the potential of the well-known Breadth-First Search algorithm and the fast calculation of the greedy McAllister function. Using this combination, and modifying the Scatter Search scheme in such a way the reference set takes into account all the objectives that are being optimized, high quality solutions are reached in low computational times. In addition, the use of Path Relinking in its random version to perform the combination method of the Scatter Search metaheuristic provides a greater diversification. This mechanism allows to populate the non-dominated set of solutions, which derives in better results regarding to the classical multi-objective metrics and the context-based metrics studied in this work.

In the experimental phase, the contribution of each part of the algorithm is demonstrated. Also, the proposal reports better solutions in terms of quality than the best previous work found in the literature, giving the opportunity to researchers and practitioners from different knowledge areas to obtain solutions with useful and accurate information, even when the network is constantly evolving.

This work is a work in progress that will be sent to a journal indexed in JCR.

## 6.6 Future work

This Thesis has been focused on solving a well-known Social Network Analysis problem: the Community Detection Problem. More specifically, it solves these problems in a heuristic way. In future works, the combination of heuristic and exact methodologies can be explored, applying math-heuristic algorithms to solve the CDP. Also, exploring new objective functions to provide the heuristic algorithms more accurate information about the problem that is being solved is a good future line of research.

From the obtained conclusions can be extracted the main idea that taking into account the edges that are present in the network is one of the bullet points to obtain high-quality results in community detection problems. In future research, it would be interesting to exploit the social networks in such a way that allows to associate information to the nodes, and not only to the edges. Using the information related to an user in the network (and not only to their relations) could give more reliable solutions, given that the knowledge about the network that is being treated could be much greater.

Also, applying machine learning and deep learning techniques to solve the community detection problems could be an interesting research line. Developing a neural network that is able to generate high-quality solutions with an unsupervised learning methodology is an interesting idea that can be studied in the future. In this sense, a new good research line can be the study of the application of the *Deep Graph Library* (DGL) [141], a very recent proposal that optimizes Graph Neural Networks (GNNs) by translating its computational patterns into generalized sparse tensor operations. Making a literature review, as far as we found, all related-works to community detection problems that make use of DGL solve them in a supervised learning based way. An interesting new research line is to solve community detection problems with an unsupervised learning approach.

Finally, the study of the community detection problem has derived in the research of problems that belong to different families, but that are interesting due to their real-world applications and their predisposition to be solved by metaheuristic algorithms due to their particular characteristics.

# Part II

# Publications: published, accepted, and submitted papers

# Contents

# Chapter 7

# Finding weaknesses in networks using Greedy Randomized Adaptive Search Procedure and Path Relinking

The journal paper associated to this part is:

- Pérez-Peló, S., Sánchez-Oro, J., & Duarte, A. (2020). "Finding weaknesses in networks using greedy randomized adaptive search procedure and path relinking". Expert Systems, 37(6), e12540 (doi:10.1111/exsy.12540).
- Status: **Published.**
- Impact Factor (JCR 2020): 2.587
- Subject Category: Computer Science, Theory & Methods. Ranking 31/110 (Q2)
- Subject Category: Computer Science, Artificial Intelligence. Ranking 73/139 (Q3)

Check for updates

Received: 18 March 2019 | Revised: 14 January 2020 | Accepted: 31 January 2020

DOI: 10.1111/exsy.12540

SPECIAL ISSUE PAPER

Expert Systems WILEY

# Finding weaknesses in networks using Greedy Randomized Adaptive Search Procedure and Path Relinking

Sergio Pérez-Peló | Jesús Sánchez-Oro | Abraham Duarte

Department Computer Science, Universidad Rey Juan Carlos, Móstoles, Spain

**Correspondence**
Jesús Sánchez-Oro, Department Computer Science, Universidad Rey Juan Carlos, C/Tulipán, S/N, Móstoles, Spain.
Email: jesus.sanchezoro@urjc.es

**Funding information**
Comunidad de Madrid, Grant/Award Number: S2018/TCS-4566; Ministerio de Ciencia e Innovación, Grant/Award Number: PGC2018-095322-B-C22

## Abstract

In recent years, the relevance of cybersecurity has been increasingly evident to companies and institutions, as well as to final users. Because of that, it is important to ensure the robustness of a network. With the aim of improving the security of the network, it is desirable to find out which are the most critical nodes in order to protect them from external attackers. This work tackles this problem, named the $\alpha$-separator problem, from a heuristic perspective, proposing an algorithm based on the Greedy Randomized Adaptive Search Procedure (GRASP). In particular, a novel approach for the constructive procedure is proposed, where centrality metrics derived from social network analysis are used as a greedy criterion. Furthermore, the quality of the provided solutions is improved by means of a combination method based on Path Relinking (PR). This work explores different variants of PR, also adapting the most recent one, Exterior PR, for the problem under consideration. The combination of GRASP + PR allows the algorithm to obtain high-quality solutions within a reasonable computing time. The proposal is supported by a set of intensive computational experiments that show the quality of the proposal, comparing it with the most competitive algorithm found in the state of art.

**KEYWORDS**

critical nodes, GRASP, metaheuristic, Path Relinking, $\alpha$-separator problem

## 1 | INTRODUCTION

In recent years, the words cyber-attack, information leakage, invasion of privacy, etc. have become increasingly common in the media. The increase in the number of attacks on networks, as well as concerns about privacy on the Internet, have created the need for more secure, reliable and robust networks. A cyber-attack on a company involving the leakage of personal information from its customers can cause significant economic and social damage (Andersson et al., 2005). Moreover, Denial of Service and Distributed Denial of Service have become two of the most widespread attacks as a successful attack can result, for example, in the deactivation of an Internet provider's service. On the other hand, if other services depend in some way on the service under attack, a cascade down may occur, affecting a considerably high number of customers. (Crucitti, Latora, & Marchiori, 2004).

It is important to identify which are the most important nodes in a network. This is a matter of concern to both actors: the attacker and the defender. On the one hand, the former is interested in disabling these important nodes to make the network more vulnerable. On the other hand, the second one will be interested in reinforcing the protection over these nodes, applying more robust security measures. Therefore, the identification of the most critical/weakest nodes in the network is a key part of the network security system.

We define a network as a graph $G = (V, E)$, where $V$ is the set of vertices ($|V| = n$), and $E$ is the set of edges ($|E| = m$). A vertex $v \in V$ represents a network node, while an edge $(u, v) \in E$, with $u, v \in V$, indicates a connection or link in the network between nodes $u$ and $v$. We define a

 © 2020 John Wiley & Sons, Ltd 

network separator as the set of vertices $S \subseteq V$ whose removal divides the network into $p$ connected components, $C_1$, $C_2$, ..., $C_p$, with $p \geq 2$. It can be trivially demonstrated that $V \setminus S = \cup_{i=1}^{p}$ and $C_i \cap C_j = \emptyset$ with $1 \leq i, j \leq p$ and $i \neq j$. Therefore, each subset $C_i$ induces a subgraph $G_i = \{V_i, E_i\}$ with $V_i = C_i$ and $E_i = \{(u, v) \in E : u, v \in C_i\}$, which is disconnected from the remaining induced subgraphs.

This work focuses on the $\alpha$-separator problem ($\alpha$-SP), which consists of finding the minimum set of vertices $S^*$, whose elimination separates the network $G$ into different connected components with fewer than $\lceil \alpha \cdot n \rceil$ vertices in each component. It is worth mentioning that the number of resultant connected components is not relevant to this problem. The actual constraint forces the number of vertices in each component to be less than or equal to $\alpha \cdot n$, with $\alpha$ being an input network parameter. This problem is $\mathcal{NP}$-hard for general topology networks when $\alpha \leq 2/3$ values are considered (Feige & Mahdian, 2006).

In Figure 1a, we show an example of a network with nine nodes and 10 links. Let us assume that $\alpha = 2/3$. Then, the size of each resulting connected component must not exceed $\lceil \alpha \cdot n \rceil = 2/3 \cdot 9 = 6$ nodes. Figure 1b depicts a feasible solution $S_1$ for the $\alpha$-SP, with a separator conformed with nodes **B**, **C**, **E** and **I**. If we remove the separator and the corresponding links, the network is divided into two connected components, $C_1 = \{\mathtt{A},\mathtt{D}\}$ and $C_2 = \{\mathtt{F},\mathtt{G},\mathtt{H}\}$, with sizes 2 and 3, respectively. Then, the objective function value for this solution is equal to 4 (i.e. the size of the separator). In Figure 1c, we represent a different solution $S_2 = \{\mathtt{A},\mathtt{B}\}$. The removal of this separator (and the associated edges) produces three connected components $C_1 = \{\mathtt{E}\}$, $C_2 = \{\mathtt{D}\}$ and $C_3 = \{\mathtt{C},\mathtt{G},\mathtt{H},\mathtt{I},\mathtt{F}\}$ that satisfy the size constraint (i.e. the size of each connected component is lower than 6). Notice that $S_2$ is better than the $S_1$ as $|S_2| < |S_1|$.

Special types of networks, such as those whose topology is a tree or a cycle, can be solved in polynomial time (Mohamed-Sidi, 2014). In that work, an approximation algorithm is also presented, based on a greedy criterion, which provides an approach ratio of $\alpha \cdot n + 1$. Nevertheless, these algorithms need to have a priori knowledge about the network topology, which is not usual in real-life problems. A different approach is described in Wachs, Grothoff, and Thurimella (2012), where the authors present a heuristic algorithm to study the separators in Autonomous Systems[1] of Internet.

The $\alpha$-SP is related with other well-known optimization problems. In particular, if $\alpha = 1/n$, then it is equivalent to the *minimum vertex cover problem* (Li, Hu, Zhang, & Yin, 2016. In addition, if $\alpha = 2/3$, then it is analogous to the *minimum dissociation set problem* (see Yannakakis (1981) for more node-deleting problems). The $\alpha$-SP can be considered a generalization of these problems, which were also proven to be $\mathcal{NP}$-hard in Garey and Johnson (1979). A deep study on the complexity and some approximation results are presented in Ben-Ameur, Mohamed-Sidi, and Neto (2015).

This problem has been mainly ignored from a heuristic point of view. In particular, the most competitive algorithm for solving the $\alpha$-SP is based on a Markov Chain Monte Carlo method hybridized with Random Walks Lee, Kwak, Lee, and Shroff (2017). Authors in that work show the superiority of their proposal, comparing it with simple heuristics such as highest-degree-first and greedy procedures.

This paper is structured as follows. Section 2 proposes an algorithmic approach to solve this problem. Section 3 presents a post-processing method based on the Path Relinking methodology. Section 4 tests the quality of the proposed algorithm, and finally, Section 5 exposes the conclusions obtained from this research.

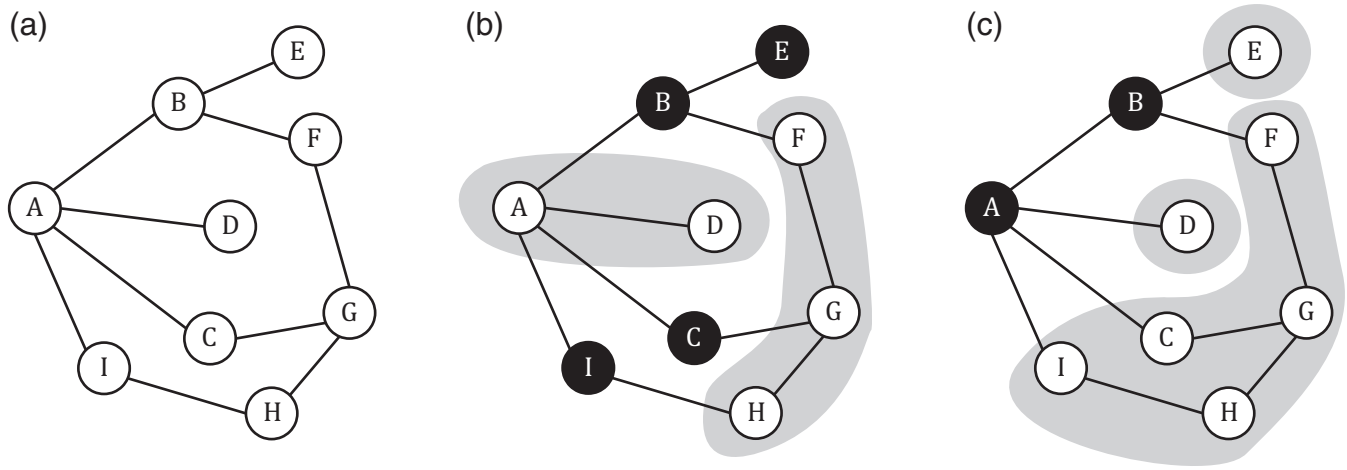



**FIGURE 1** (a) Example graph derived from a network with nine nodes and ten connections between nodes, (b) a feasible solution with four nodes in the separator (**B**, **C**, **E** and **I**) and (c) a better solution with two nodes in the separator (**A** and **B**)

## 2 | GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE

Greedy Randomized Adaptive Search Procedure (GRASP) is a multi-start strategy that was originally proposed in Feo and Resende (1989), but it was not formally defined until Feo, Resende, and Smith (1994). As a multi-start method, GRASP tries to avoid getting trapped in local optima by restarting the search from a different starting point in the search space. This behaviour allows the algorithm to increase the diversification of the search. This metaheuristic has been recently used successfully in several optimization problems (Duarte, Sánchez-Oro, Resende, Glover, & Martí, 2015; Quintana, Sánchez-Oro, & Duarte, 2016; Sánchez-Oro, Laguna, Martí, & Duarte, 2016).

Each GRASP iteration has two distinct phases: construction and improvement. The former is designed for constructing an initial high-quality solution, while the objective of the latter is to improve the initial solution by means of a local optimization method (F. W. Glover & Kochenberger, 2006). On the one hand, constructive methods start from an empty solution and then iteratively adding elements to it. On the other hand, the local optimizer considered for the second phase is a local search method, but GRASP is commonly hybridized with more complex search methods, such as Tabu Search (F. Glover & Laguna, 1998) or Variable Neighbourhood Search (Hansen & Mladenović, 2014), among others, which highlights the versatility of GRASP methodology.

### 2.1 | Construction phase

The constructive procedure, named *GreedyRandom*, starts from an empty solution, and it iteratively adds nodes to the solution under construction until it becomes feasible. With the aim of diversifying the search, the first node to be inserted into the separator (i.e. the solution in the context of the $\alpha$-SP) is randomly selected from *V*. Then, it is necessary to add vertices to it until the problem constraint is satisfied: the size of every connected component in the network should be smaller than $\lceil \alpha \cdot n \rceil$. Algorithm 1 presents the pseudocode of the proposed method.

**Algorithm 1** *GreedyRandom*(G, $\beta$)

1: v ← *Random*(V)
2: S ← {v}
3: CL ← V \ {v}
4: **while not** *isFeasible*(S) **do**
5: $g_{min} \leftarrow \min_{v \in CL} g(v)$
6: $g_{max} \leftarrow \max_{v \in CL} g(v)$
7: $\mu \leftarrow g_{max} - \beta \cdot (g_{max} - g_{min})$
8: RCL ← {v ∈ CL : g(v) ≥ $\mu$}
9: v ← *Random*(RCL)
10: S ← S ∪ {v}
11: CL ← CL \ {v}
12: **end while.**
13: **return** S

The procedure starts by randomly choosing the first node that will be included in the solution (steps 1–2). The random selection of this first node allows the procedure to increase the diversity of the solutions generated at the construction phase. Next, the candidate list *CL* is created with all the nodes of the network except the initially chosen one (step 3). Then, the constructive method iteratively adds a new node to the solution until it satisfies the aforementioned problem constraints (steps 4–12). The *is Feasible* function is intended to check that the size of all the connected components of the network is smaller than $\lceil \alpha \cdot n \rceil$.

The selection of the next element to be incorporated in the solution under construction is a key aspect in the design of effective GRASP constructive methods. Considering the complexity of the proposed greedy function (based on the closeness centrality metric, Newman (2008)), it will be thoroughly described in Section 2.2. For the sake of clarity, we now refer to this function as *g*. Steps 5 and 6 evaluate the minimum and maximum value of the greedy function among all the candidates, respectively. These two values are used to calculate the $\mu$ threshold (step 7) that is responsible for limiting the nodes that will be part of the restricted candidate list *RCL* (step 8). The threshold value depends on a $\beta$ parameter (with $0 \leq \beta \leq 1$) that controls the randomness/greediness of the method. On the one hand, if $\beta = 0$, then $\mu = g_{max}$, and, therefore, only those vertices with the maximum value of the greedy function will be considered in the *RCL*, resulting in a totally greedy method. On the other hand, if $\beta = 1$, then $\mu = g_{min}$, allowing the selection of every node in the *CL*, becoming a totally random method. Through this way, the higher the value of the $\beta$ parameter, the more random the method is and vice versa. Section 4 will discuss how this parameter affects the quality of the obtained solutions.

This work additionally proposes an alternative constructive procedure, named *RandomGreedy*, where the greedy and random phases are swapped. In the *GreedyRandom* variant, the *RCL* is created with the most promising elements of the *CL* (i.e. those whose greedy function value exceeds a certain threshold), and then, it randomly selects the next element from the *RCL*. On the contrary, the *RandomGreedy* variant first constructs the *RCL* with a set of $\beta \cdot |CL|$ elements randomly selected from the *CL*, and then, it chooses the element with the largest greedy function value among those included in the *RCL*. The main difference between the *GreedyRandom* and *RandomGreedy* constructive procedure is the interchange between the greedy and random phases of the constructive procedure. In addition, it is worth mentioning that *RandomGreedy* is presumably faster than *GreedyRandom* as it does not require the evaluation of the complete *CL* but it requires the evaluation of the elements included in the *RCL* (this hypothesis will be confirmed in Section 4).

## 2.2 | Greedy criterion

The selection of the next element to be incorporated in a partial solution in GRASP methodology depends on a greedy criterion. Specifically, for each node $v \in V$, it is necessary to define a greedy function *g(v)*, which evaluates the relevance of inserting it into the partial solution. In the context of the $\alpha$-SP problem, the greedy criterion should indicate how likely a node is to be critical.

This work presents a novel approach to identify whether a node is critical or not. In particular, it consists of using criteria derived from social network analysis (SNA). Following this idea, the greedy criterion will consider the network a social network, identifying which nodes are relevant in the network as if it were a SNA problem. In SNA, a node is considered relevant if it is responsible for maintaining a large number of users connected through it Scott and Carrington (2011). Therefore, the analogy with the $\alpha$-SP problem is straightforward: a node is important in the network if its removal disconnects several devices from the network.

The literature of SNA offers a vast amount of metrics to evaluate the relevance of a user in a social network. However, it must be kept in mind that the evaluation of most of them is rather computationally demanding. This is mainly because most of them, such as *betweenness* (Brandes, 2001) or *pagerank* (Page, Brin, Motwani, and Winograd (1999)), need to know all the shortest paths between each pair of users in the network. It is worth mentioning that the construction of all shortest paths between each pair of vertices is unapproachable for real problems. Then, it is usually tackled by considering a fixed number of the shortest paths (Yen (1971)).

In this work, we will use the metric known as *closeness* (Newman (2008)), which considers that a vertex is important if it is close to several nodes in the network. More formally, given a connected network *G*, the *closeness* value for a vertex is calculated as the inverse of the sum of the lengths of the shortest paths between the analysed node and the remaining nodes in the graph. It should be noted that the evaluation of this metric only requires the evaluation of the shortest path between each pair of vertices, so it is enough to make a breadth-first search (if the network is not weighted) or to apply the Dijkstra algorithm (if the network is weighted). The idea that underlies this metric is that the more central a node is, the closer to all the other nodes it is. Indeed, the larger the closeness value, the more relevant a node.

We need to adapt the general definition of closeness to the $\alpha$-SP as this optimization problem deals with different connected components. It becomes a problem because the distance between nodes in different connected components is evaluated as infinite. Then, we only evaluate distances from each node *v* in a specific connected component $C_i$ (with $1 \le i \le p$, with *p* being the number of connected components) to the remaining nodes in $C_i$. Specifically, the *closeness* of a node $v \in C_i$, defined as *C(v)*, is then evaluated as the inverse of the sum of the distances from that node *v* to the other nodes in the same connected component. In mathematical terms,

$$C(v) = \frac{1}{\sum_{u \in C_i} d(v,u)}$$

where *d(v, u)* is the distance between nodes *v* and *u*, and $C_i$ is the connected component to which both nodes belong. The distance between nodes *v* and *u* considered in this work is evaluated as the length of the shortest path that connects *u* with *v* in the network.

## 2.3 | Improvement phase

Solutions generated in the construction phase described in Section 2.1 are constructed by a randomized procedure that controls its greediness through a parameter $\beta$. For this reason, the solution constructed may not be a local optimum, and therefore, it can be further improved through any local optimization method. This work proposes a local search procedure to improve those solutions.

Local search procedures are designed with the objective of finding the local optimum of a solution with respect to a predefined neighbourhood. A neighbourhood can be defined as all the solutions that can be obtained from a single one by means of a single movement. Then, prior to defining the neighbourhood, we must define the considered movement. We define the movement *Move*2x1(*S*, *v*, *u*, *w*) as the removal of nodes *v* and *u* from the current solution (i.e. separator) and the insertion of node *w* into it. More formally,

$$Move2x1(S,v,u,w) = (S \setminus \{u,v\}) \cup \{w\}$$

Notice that this movement may lead to unfeasible solutions as the resulting one might not be a separator for the problem under consideration. The proposed local search method will only consider those moves that result in a feasible solution, discarding those moves that produce unfeasible solutions. In addition, it is worth mentioning that any feasible movement will result in an improvement as the size of the separator would have been decreased in one unit.

Then, given a feasible solution $S$, its associated neighbourhood $N_{2\times1}(S)$ is defined as all the solutions that can be generated by applying the *Move2x1* operator to solution $S$, with any of the available nodes. More formally,

$$N_{2\times1}(S) \leftarrow \{S' \leftarrow Move2x1(S,v,u,w)\ \forall v,u \in S \wedge \forall w \in V \setminus S\}$$

The last step for defining a local search method consists of establishing the order in which the neighbourhood will be explored. Traditionally, two methods for exploring the neighbourhood are considered: *First Improvement* and *Best Improvement*. The former performs the first improving move found when the scanning the neighbourhood, while the latter requires the complete neighbourhood to be analysed, performing the movement that leads to the best solution. This work only considers the *First Improvement* approach, mainly due to the following reasons. First, the *First Improvement* strategy is usually less computationally demanding than *Best Improvement* because it does not explore the complete set of reachable solutions in each step but performs the first improving move found. Furthermore, note that, given the defined neighbourhood, any movement that leads to a feasible solution will produce a better solution in terms of the objective function value. Specifically, the new solution will outperform the original one in one unit as the new separator will contain one node less than the original solution. Then, the *Best Improvement* approach is discarded for this local search.

Finally, there is a need to indicate the order in which the neighbour solutions will be explored as, in the context of *First Improvement*, the order of exploration may affect the quality of the resulting solutions. The local search proposed in this work randomly explores the neighbourhood to increase the diversification of the search.

The local search method stops when no improvement is found after exploring the complete neighbourhood, returning the best solution found during the search.

## 3 | PATH RELINKING

*Path Relinking* (PR) is a solution combination method originally proposed as a methodology to integrate intensification and diversification strategies in the Tabu Search context (F. Glover, 1996). The PR behaviour is based on exploring the trajectories that connect high-quality solutions, with the objective of generating intermediate solutions that can eventually be better than the original solutions used to build the trajectory. Laguna and Martí (1999) adapted PR to the GRASP context with the aim of increasing the intensification of the search. The PR algorithm operates over a set of solutions, called *Elite Set* (ES), that is usually ordered following a predefined descendent quality criterion of each solution. In this work, we consider the value of the objective function of the solution as a quality criterion. In particular, the ES consists of the highest-quality solutions generated by the GRASP algorithm. This design is generally denominated static (F. Glover, Laguna, & Martí, 2000) because GRASP is first applied to build the ES, and then, PR is applied to explore the trajectories between all pairs of solutions in the ES. On the contrary, the dynamic design considers updating the ES each time a trajectory is explored.

A trajectory between an initial solution $S_i$ and a guiding one $S_g$ is generated by iteratively including attributes or properties of the guiding solution into the initial one. In the context of $\alpha$-SP, we will include in the separator of the initial solution those nodes that are present in the separator of the guiding one by interchanging them with those nodes that are in the initial solution but not in the guiding one. For this purpose, we first need to define a new movement that removes from a separator $S$ the node $v$ and inserts node $u$ into it, producing a new solution $S' \leftarrow (S \setminus \{v\}) \cup \{u\}$. For the sake of clarity, we named this move $S' \leftarrow Swap(S, v, u)$.

Then, a trajectory between two solutions is created by iteratively performing $Swap(S_i, v, u)$ moves, where $v \in S_i \setminus S_g$ and $u \in S_g \setminus S_i$ until $S_i$ becomes $S_g$. In each iteration, PR performs a *swap* movement that generates a new intermediate solution in the trajectory. It should be noted that this movement produces, with high probability, an unfeasible solution (i.e. the intermediate solution is not a valid separator), so a repair operator must be applied to the intermediate solution to make it feasible. The proposed repair operator consists of adding random nodes to the separator until the solution becomes feasible, thus increasing the diversification of the search.

Figure 2 shows a trajectory created between two solutions, $S_i = \{B, I, G, H\}$ and $S_g = \{A, B, C, F\}$, for the graph depicted in Figure 1a, but this time considering $\alpha = 1/3$. Then, the network must be disconnected in components of $\lceil \alpha \cdot n \rceil = 1/3 \cdot 9 = 3$ nodes at most. The illustration is divided into three parts: path, which represents the actual path created between both solutions; repair, which considers those solutions of the path that have been repaired in order to make them feasible; and clean, which includes solutions of the path that originally contained redundant nodes (i.e. those that are not needed any more in the separator) where these redundant nodes have been removed. In this example, the initial solution is

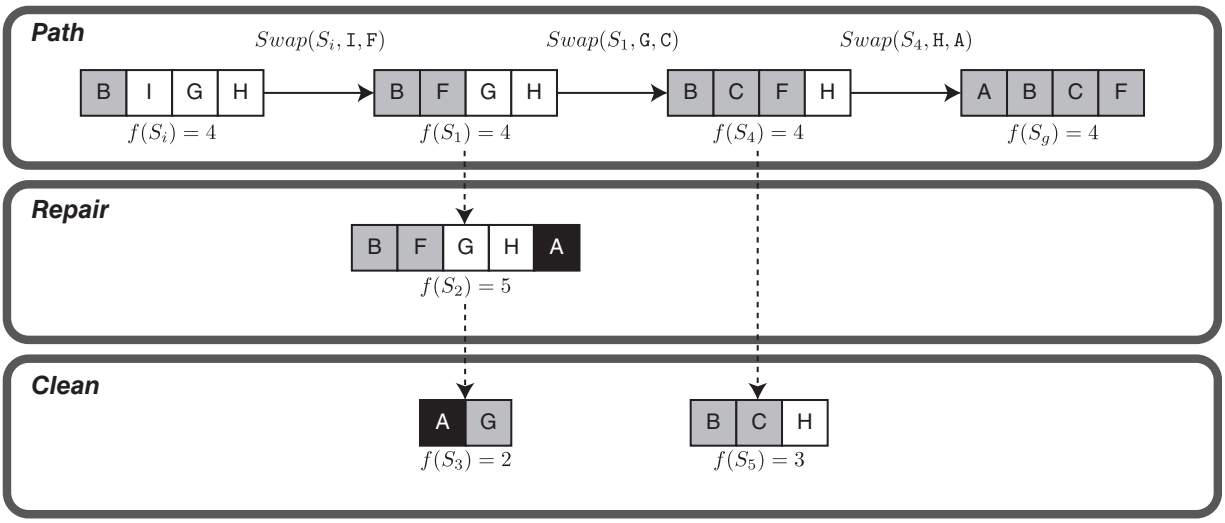



**FIGURE 2** Example of a trajectory created between solutions $S_i = \{\mathtt{B},\mathtt{I},\mathtt{G},\mathtt{H}\}$ and $S_g = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{F}\}$, considering $\alpha = 1/3$

$S_i$, while the guiding one is $S_g$, so the trajectory consists of creating a path of solutions from $S_i$ to $S_g$. In each step of the path, a new element belonging to the separator of $S_g$ will be included in $S_i$, swapping it with an element included in $S_i$ that does not belong to $S_g$. Figure 2 shows the nodes included in the separators of each solution, and the nodes in common with $S_g$ have been highlighted in grey.

The first solution in the path, $S_1$, is generated by performing the move $Swap(S_i, \mathtt{I}, \mathtt{G})$, thus removing node **I** from the separator and replacing it with **G**. Notice that the resulting solution $S_1$ is not feasible as the graph is divided into components $C_1^1 = \{\mathtt{E}\}$ and $C_2^1 = \{\mathtt{A},\mathtt{D},\mathtt{C},\mathtt{I}\}$, so $|C_2^1| > 3$, violating the problem constraint. Then, $S_1$ must be repaired by inserting new nodes into the solution until it becomes feasible. Inserting node *A* into the separator generates solution $S_2$, which is a feasible solution. After repairing a solution, we may have introduced redundant nodes in the separator that are not required anymore. Therefore, we scan the separator in order to remove those nodes that are not required to maintain the feasibility of the solution. In this case, nodes **B**, **F**, **G** and **H** are removed from the solution, generating solution $S_3$, which is feasible. In addition, the number of nodes in $S_3$ is 2, being the best solution found in the path so far.

Once reaching a feasible solution, the path continues with the original unfeasible one, which is $S_1$. The next solution, $S_4$, is generated from the move $Swap(S_1, \mathtt{G}, \mathtt{C})$, which is already a feasible solution, disconnecting the network into $C_1^4 = \{\mathtt{E}\}$, $C_2^4 = \{\mathtt{G}\}$ and $C_3^4 = \{\mathtt{A},\mathtt{D},\mathtt{I}\}$, all of them with a number of nodes lesser than or equal to 3. This move may produce a solution with redundant nodes again. In this case, node **F** is not necessary anymore in the solution, so we remove it to generate $S_5$, obtaining a better solution than $S_4$.

The last movement, $Swap(S_4, \mathtt{H}, \mathtt{A})$, leads us to reaching $S_g$, concluding the trajectory between $S_i$ and $S_g$. At this point, the algorithm returns the best solution found in the trajectory, which is $S_3$, as it only contains two nodes in the separator. This new solution achieves an improvement of two nodes with respect to the solutions considered to be endpoints of the trajectory.

The choice of the next *Swap* movement to be performed in each iteration results in two different PR strategies. On the one hand, *Greedy PR* (GPR) selects, in each iteration, the best *Swap* movement available. In particular, it generates all the possible solutions that can be generated in the current trajectory, also applying the repair operator. Then, the trajectory continues through the best intermediate solution. On the other hand, *Random PR* (RPR) generates the next solution of the trajectory by randomly selecting one of the available *Swap* moves in the path. Notice that GPR is focused on intensification, sacrificing computational time, while RPR aims for diversification, resulting in the fastest strategy as it does not require a complete analysis of the available *Swap* moves to be performed.

In this work, we propose a third variant that mixes the GPR and RPR, named *Greedy Randomized PR* (GRPR). This new strategy generates all the possible intermediate solutions as in GPR, but instead of selecting the best one, it selects one of the most promising ones, following the same strategy than the GRASP constructive stage. We consider a parameter $\sigma \in [0, 1]$ (equivalent to $\beta$ in GRASP), which is able to establish a trade-off between greediness and randomness of the strategy. This strategy allows us to create a path focused on intensification but escaping from local optima derived from choosing a totally greedy variant, balancing the diversification and intensification of the search. Notice that evaluating all possible paths between solutions can eventually lead to the evaluation of a large number of intermediate solutions. The number of intermediate solutions is limited by the number of different critical nodes in the solutions considered to construct the path. However, high-quality solutions share a reasonable percentage of critical nodes, so the number of different critical nodes in two high-quality solutions is small, thus leading to a small number of intermediate solutions in the path between them. This behaviour allows the algorithm to evaluate all possible paths between two solutions.

The intermediate solutions generated during trajectory exploration are not necessarily locally optimal with respect to any neighbourhood, even more considering the repair operator. Therefore, the local search method described in Section 2.3 is applied to the best solution found in the path to further improve its quality.

This work considers that all the constructed solutions are included in the ES (avoiding repeated solutions). Hence, PR is applied to every pair of different solutions generated in the constructive and improvement phases of the algorithm.

Finally, a new PR strategy, named *Exterior PR*, is presented. The aforementioned variants (GPR, RPR and GRPR) are focused on constructing a path that connects two solutions. However, if the solutions considered for the path are very similar, then the trajectory will be short, which is not interesting in the PR methodology. The *Exterior PR* (EPR) variant was originally proposed by Duarte et al. (2015) with the aim of exploring trajectories that go beyond the considered solutions. In particular, starting from the initial ($S_i$) and guiding ($S_g$) solutions, EPR tries to generate, in each iteration, a solution obtained by removing from the initial solution those elements that are already present in the guiding one. Therefore, the main idea of EPR is not getting closer to the guiding solution but getting further from it. Hence, this strategy is totally focused on the diversification of the search, trying to generate intermediate solutions that are rather different from the two endpoints of the path. EPR ends when the initial solution becomes completely different to the guiding one (i.e. $S_i \cap S_g = \emptyset$), returning the best solution found in the path. Again, the local search method is applied to the best intermediate solution in order to obtain a local optimum.

## 4 | COMPUTATIONAL RESULTS

This section has two main objectives: 1) find the best values for the parameters of the proposed algorithm and 2) compare the selected variant with the best previous method found in the state of the art. All the algorithms have been developed in Java 9, and the experiments have been conducted in an Intel Core 2 Duo 2.66 GHz with 4GB of RAM computer.

It is important to remark that the instance set used in the best previous method is not publicly available, and we have not received a response from the previous authors regarding this issue. Notwithstanding, the instance set used in this experimentation has been generated using the same graph generator proposed by the best previous work (Lee et al., 2017). Specifically, graphs are generated using the Erdös Rényi model (Erdös & Rényi, 1960), in which each new inserted node has the same probability of being connected to any existent node in the graph. A set of 50 instances (25 with 100 vertices and 25 with 200 vertices) with different density (from 200 to 500 edges) has been generated. Each instance has been tested considering $\alpha$ = {0.2, 0.4, 0.6}. In order to facilitate future comparisons, the instances can be requested by email to the authors.

Experimentation is divided into two different parts: preliminary experimentation and final experimentation. The former is designed to adjust the parameters for the algorithms and select the best version of them, while the latter has the aim of confirming the quality of the proposal by comparing it with the best previous method found in the state of the art. It is worth mentioning that the preliminary experimentation will consider a representative subset of 20 of 50 instances to avoid overfitting.

All the experiments report the following metrics: *Avg*., the average objective function value; *Time (s)*, the average computing time measured in seconds; *Dev*. (%), the average deviation with respect to the best solution found in the experiment; and *#Best*, the number of times the algorithm reaches that best solution.

The first preliminary experiment is designed for analysing the performance of the proposed constructive procedures and selecting the best variant. In particular, this experiment analyses the effect of the $\beta$ parameter in both constructive procedures presented in Section 2.1. The behaviour of the constructive procedure isolated is not a faithful representation of its adequacy to the problem. This is mainly because certain variants may produce worse quality solutions that are a better starting point for the local improvement method. Therefore, the experiments designed to select the best constructive procedure consists of generating 10 different solutions, improving them with the local search presented in Section 2.3. Considering the value for the $\beta$ parameter, we have tested $\beta$ = {0.25, 0.50, 0.75, *RND*}, where RND indicates that a random value is selected in each independent construction. Table 1 shows the results obtained by the *GreedyRandom* constructive procedure.

As it can be derived from these results, increasing the value of $\beta$ worsens the quality of the solutions obtained, which is reflected in all the considered metrics. This result suggests that the best solutions are reached when reducing the randomness in the constructive procedure.

**TABLE 1** Results for the constructive procedure *GreedyRandom* when coupled with the local search method

| Algorithm | Avg. | Time (s) | Dev. (%) | #Best |
|---|---|---|---|---|
| *GreedyRandom* (0.25) | 68.35 | 16.69 | 1.16 | 11 |
| *GreedyRandom* (0.50) | 68.95 | 16.92 | 2.33 | 7 |
| *GreedyRandom* (0.75) | 71.35 | 18.30 | 6.69 | 1 |
| *GreedyRandom* (RND) | 68.70 | 17.61 | 2.10 | 10 |

*Note:* The number between parenthesis indicates the value of $\beta$.

Therefore, we select $\beta$ = 0.25 for the *GreedyRandom* procedure. Analysing the computing time, we can clearly see that the differences among the considered variants are negligible.

We perform an equivalent experiment for the second proposed constructive method. Specifically, Table 2 shows the corresponding results when considering the *RandomGreedy* procedure.

These results confirm the hypothesis, suggesting that the lower the randomness of the constructive procedure, the better the results. Again, selecting $\beta$ = 0.25 leads the constructive procedure to obtain the best solutions in all the considered metrics: a larger number of best solutions found while maintaining the smallest deviation with respect to the best solution when it is not reached. Therefore, we again select $\beta$ = 0.25 for the *RandomGreedy* variant. Once again, the computing time required by each algorithm is rather similar.

To conclude with the constructive procedure selection, we directly compare the results obtained with the best *GreedyRandom* variant with those obtained by the best *RandomGreedy* variant. Table 3 shows the comparison between *GreedyRandom* and *RandomGreedy* when considering $\alpha$ = 0.25.

Analysing the results obtained, the *RandomGreedy* variant is clearly better than the *GreedyRandom* procedure. In particular, it is able to reach a larger number of best solutions (13 versus 15), with a deviation of just 0.77%, while *GreedyRandom* obtains 3.27%. These results suggest that *RandomGreedy* is able to reach most of the best solutions, and in those cases in which it does not reach the best solution, it remains close to the best solution in terms of quality. In this case, the computing time required by the *RandomGreedy* variant is also smaller as it does the evaluation of all the candidate nodes to be included in the solution under construction is not precise.

Furthermore, we can confirm the hypothesis stated in Section 2.1, which indicates that *RandomGreedy* should be faster than *GreedyRandom* as it does not require the complete evaluation of the *CL*. Analysing the results presented in Tables 1, 2 and 3, we can clearly see that *RandomGreedy* is consistently faster than *GreedyRandom* in all their variants. As a result, we select *RandomGreedy* with $\beta$ = 0.25 as the constructive procedure for the remainder of the paper.

The next experiment aims to analyse the behaviour of the local search when varying the portion of the search space explored. It is well known that an exhaustive search is rather time consuming, thus affecting the efficiency of the complete algorithm. In order to avoid this, we will now analyse two factors, objective function value and computing time, when considering a reduction of the search space explored by the local search method. In order to do so, we test the efficacy and efficiency when exploring a percentage $\delta$ of the available movements inside local search. Specifically, we test $\delta$ = {0.10, 0.20, ..., 0.90}, with delta being the percentage of nodes considered to be evaluated with the *Move2x1* operator, defined in Section 2.3. It is expected that small values of $\delta$ will result in shorter computing times, but it might deteriorate the quality of the solutions found. Figure 3 shows the comparison between quality and computing time when considering different values for the $\delta$ parameter.

As can be derived from the results, the search is consistently improving the results without considerably increasing the computing time until reaching $\delta$ = 0.6. At that point, the deviation decreases rather slowly (from approximately 4% until reaching 0.5%), while the computing time is rather larger (from about 150 seconds to more than 500 seconds). Hence, considering values larger than $\delta$ = 0.6 is not recommended for the local search method. The remaining experiment will then consider $\delta$ = 0.6.

Once the best constructive and local search methods have been identified, the next step consists of selecting the best PR variant to be considered. In particular, we propose four different variants: GPR, RPR, GRPR and EPR. Among all the variants, GRPR is the only one with a parameter, $\sigma$, that must be adjusted, controlling the balance between intensification and diversification in the combination. Therefore, the next preliminary experiment is devoted to selecting the best value for the $\sigma$ parameter. In particular, we have tested $\sigma$ = {0.25, 0.50, 0.75, *RND*} as in the previous experiments. We have generated and improved 10 different solutions with the selected constructive and local search methods and then combined each pair of solutions with the corresponding combination method. Table 4 shows the results derived from this experiment.

Although there are no significant differences in the obtained results, selecting a random value of $\sigma$ in each iteration reaches slightly better solutions, and the computing time is not drastically affected. Therefore, we select $\sigma$ = RND for the remaining experiments. Having selected the

**TABLE 2** Results for the constructive procedure *RandomGreedy* when coupled with the local search method

| Algorithm | Avg. | Time (s) | Dev. (%) | #Best |
|---|---|---|---|---|
| *RandomGreedy* (0.25) | 67.65 | 14.75 | 1.24 | 11 |
| *RandomGreedy* (0.50) | 67.90 | 15.46 | 2.10 | 8 |
| *RandomGreedy* (0.75) | 68.15 | 14.82 | 2.54 | 3 |
| *RandomGreedy* (RND) | 67.85 | 14.84 | 2.25 | 6 |

*Note:* The number between parenthesis indicates the value of $\beta$.

**TABLE 3** Comparison between best *GreedyRandom* and *RandomGreedy* constructive procedures

| Algorithm | Avg. | Time (s) | Dev. (%) | #Best |
|---|---|---|---|---|
| *GreedyRandom* ($\alpha$ = 0.25) | 68.35 | 16.69 | 3.27 | 13 |
| *RandomGreedy* ($\alpha$ = 0.25) | 67.65 | 14.75 | 0.77 | 15 |

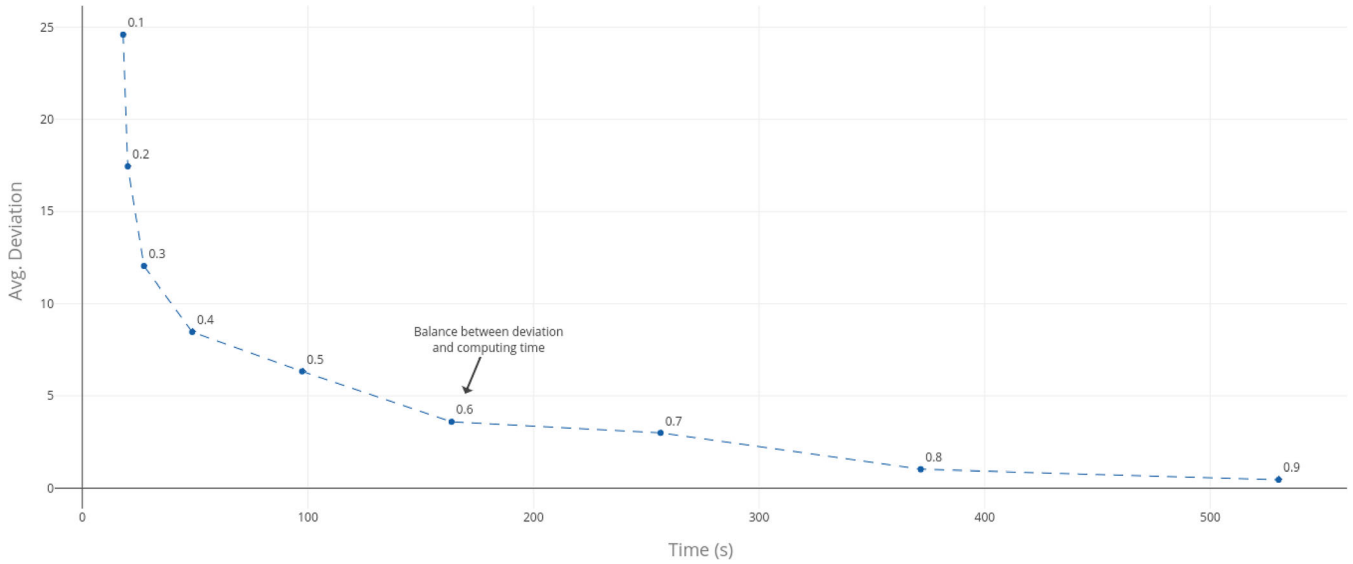



**FIGURE 3** Analysis of the average deviation and computing time of each variant of the local search procedure

**TABLE 4** Adjustment of the $\sigma$ parameter in the Greedy Randomized Path Relinking algorithm

| Algorithm | Avg. | Time (s) | Dev. (%) | #Best |
|---|---|---|---|---|
| GRPR (0.25) | 61.40 | 825.10 | 0.53 | 16 |
| GRPR (0.50) | 61.45 | 837.83 | 0.59 | 15 |
| GRPR (0.75) | 61.45 | 833.52 | 0.59 | 15 |
| GRPR (RND) | 61.25 | 835.42 | 0.49 | 16 |

*Note:* The number between parenthesis indicates the value of $\sigma$.
Abbreviations: GRPR, Greedy Randomized Path Relinking.

**TABLE 5** Comparison among the proposed Path Relinking variants

| Algorithm | Avg. | Time (s) | Dev. (%) | #best |
|---|---|---|---|---|
| GRPR (RND) | 61.25 | 835.42 | 0.89 | 13 |
| RPR | 61.25 | 838.58 | 0.91 | 13 |
| GPR | 61.25 | 839.07 | 0.96 | 13 |
| EPR | 61.20 | 845.52 | 0.85 | 13 |

Abbreviations: EPR, Exterior Path Relinking; GPR, Greedy Path Relinking; GRPR, Greedy Randomized Path Relinking; RPR, Random Path Relinking.

best GRPR variant, we now perform a comparison among all the PR variants with the aim of selecting the best one for the final experiment. Table 5 shows the comparison among RPR, GPR, GRPR and EPR.

As can be derived from the presented results, the values obtained by the different variants are rather similar. In particular, each variant is able to obtain 13 of 20 best solutions in similar computing times. We then select EPR as the best variant as it presents the smallest deviation (0.85%), although it is closely followed by GRPR (0.89%). The superiority of EPR can be partially justified as it is totally focused on the diversification of the search. If we analyse the constructive and local search methods, we can clearly see that they are mainly focused on the intensification part of the search, so the combination with EPR generates an adequate balance between diversification and intensification. Regarding the computing time, although EPR presents a slightly larger computing time, the quality of the solutions obtained makes up for it. Then, we select EPR as the best PR variant for the $\alpha$-SP.

The final experiment selects the best variant and compares it with the best previous method found in the state of the art. In particular, the proposed GRASP with PR (GRASP+PR) algorithm considers the following components:

- Constructive Procedure: *RandomGreedy*(0.25)
- Local Search: $\delta$ = 0.6
- PR: EPR

As far as we know, the best previous algorithm for the $\alpha$-SP is a Random Walk algorithm (RW) based on a Metropolis chain (Lee et al., 2017). Unfortunately, neither the original source code nor the original instances are available, so we have carefully re-implemented the original algorithm following the detailed descriptions given in the original work. It is important to remark that this experiment considers the complete set of 50 instances. In addition to GRASP+PR, we have also included in this final comparison the results obtained by GRASP (without applying the PR algorithm) as, in some scenarios, it is more important to provide a fast solution of average quality than to spend several seconds trying to refine the best solution found. Table 6 shows the results obtained in the comparison.

Regarding these results, the proposed algorithm, GRASP+PR, is able to reach a larger number of best solutions than RW (34 vs 18). Furthermore, the average deviation with respect to the best solution found is considerably small in GRASP+PR, indicating that, in those cases in which it is not able to reach the best solution, it remains rather close to it. On the contrary, the deviation of RW is 18.26%, which means that it is not close to the best solution found by GRASP+PR. Analysing the computing time, we can see that GRASP+PR is 1.30 times faster than RW, being more adequate for real scenarios. Finally, RW is able to disconnect the network by removing, on average, 71.78 nodes, while GRASP+PR can disconnect it by removing just 62.00 nodes on average, which also confirms the quality of the proposal.

In order to confirm that there are statistically significant differences between the compared algorithms, thus confirming the superiority of the proposal, we perform the pairwise Wilcoxon Signed Rank Test. The resulting $p$-value of .001 indicates that there are statistically significant differences between GRASP+PR and RW with a significance level of 95%. Therefore, GRASP+PR emerges as the best algorithm in the state of the art for the $\alpha$-SP.

As has been mentioned throughout the manuscript, computing time is a key factor in the context of $\alpha$-SP. In order to analyse this relevant issue, we conducted an additional experiment to compare the computing time required by both, the proposed algorithm, GRASP+PR, and the best method identified in the literature, RW. Figure 4 shows the computing time required by each algorithm in every single instance considered in the experimentation. As we can see, GRASP+PR is clearly faster than RW. However, an average computing time of approximately 800 seconds may not be enough in some real-case scenarios.

In view of the results obtained, we can conclude that GRASP+PR and RW require equivalent computing times when facing large instances ($n \geq 200$), although GRASP+PR is considerably faster in small and medium instances ($n < 200$). Therefore, if a fast solution is needed in special applications, we do recommend considering GRASP without applying PR, which is able to obtain a slightly larger deviation than GRASP+PR (5.90

**TABLE 6** Final comparison between GRASP, GRASP + PR and the best previous method found in the state of the art

| Algorithm | Avg. | Time (s) | Dev. (%) | #best |
|---|---|---|---|---|
| GRASP | 63.18 | 137.41 | 5.90 | 14 |
| GRASP + PR | 62.00 | 822.29 | 3.68 | 34 |
| RW | 71.78 | 1070.36 | 18.26 | 18 |

Abbreviations: GRASP, Greedy Randomized Adaptive Search Procedure; PR, Path Relinking; RW, Random Walk algorithm.

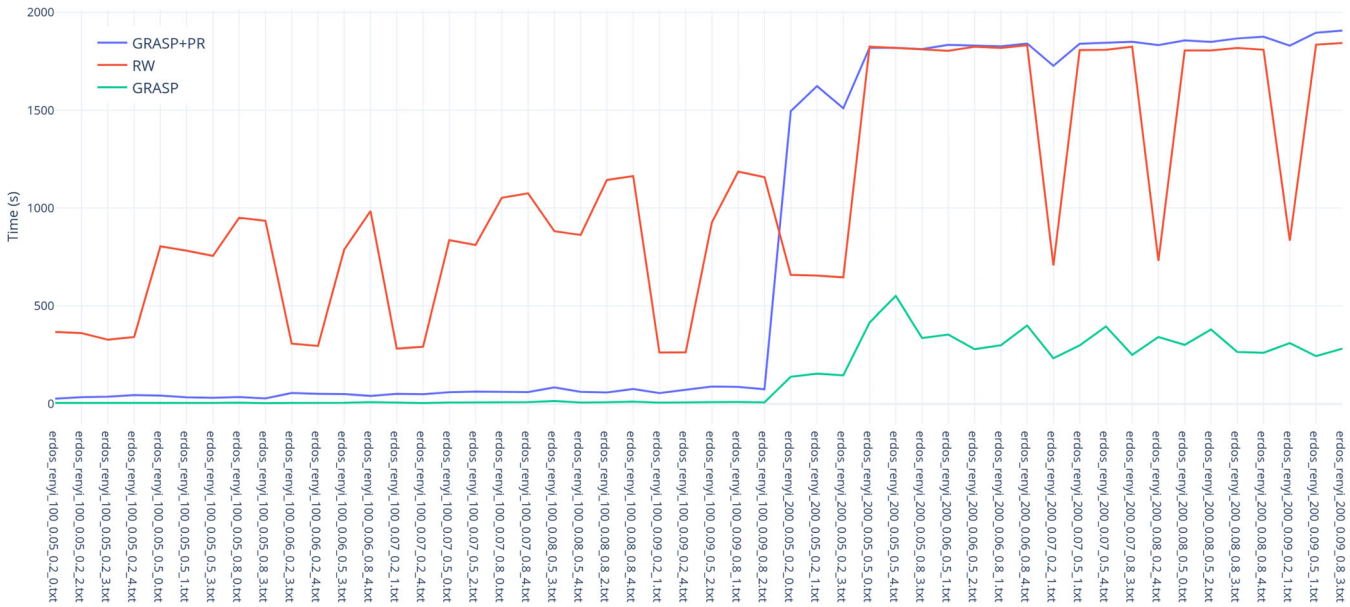



**FIGURE 4** Comparison of computing time for each one of the considered instances

vs 3.68%) but requires an eighth of the time (137.41 s) needed by GRASP+PR or RW (822.29 s and 1070.36 s, respectively). Furthermore, analysing the graph, we can see that GRASP scales better with the size of the instance.

## 5 | CONCLUSIONS

In this work, an algorithm based on GRASP has been presented for the detection of critical points in networks, particularly a constructive procedure that leverages features derived from SNA, coupled with an improvement strategy for reaching local optima. In addition, the generated solutions are combined by using PR to improve their quality. We perform a thorough experimentation for adjusting the parameters of the algorithm and show the merit of each proposed strategy. Finally, the best variant is compared with the best algorithm found in the state of the art. As can be derived from the experiments, the proposed algorithm is able to solve the $\alpha$-SP, being 1.30 faster than the previous method, obtaining clearly better results, which are supported by non-parametric statistical tests. The proposed algorithm emerges as the best method in the state of the art, being able to obtain high-quality solutions in short computing times with GRASP and improve them later with PR in those situations in which time is not critical.

### ACKNOWLEDGEMENTS

This work has been partially funded by 'Ministerio de Ciencia, Innovación y Universidades' under grant ref. PGC2018-095322-B-C22 and by 'Comunidad de Madrid' and 'Fondos Estructurales' of European Union with the project 'Cybersecurity, Network Analysis and Monitoring for the Next Generation Internet' (CYNAMON), with grant ref. S2018/TCS-4566.

### CONFLICT OF INTEREST

The authors confirm that there are no conflicts of interest.

### ORCID

*Sergio Pérez-Peló* https://orcid.org/0000-0002-1915-4160
*Jesús Sánchez-Oro* https://orcid.org/0000-0003-1702-4941
*Abraham Duarte* https://orcid.org/0000-0002-4532-3124

### ENDNOTE

[1] An Autonomous System conforms to one or more networks managed and supervised by a single entity or organization.

### REFERENCES

Andersson, G., Donalek, P., Farmer, R., Hatziargyriou, N., Kamwa, I., Kundur, P., … Vittal, V. (2005). Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance. *IEEE Transactions on Power Systems*, *20*(4), 1922–1928.

Ben-Ameur, W., Mohamed-Sidi, M.-A., & Neto, J. (2015). The k-separator problem: Polyhedra, complexity and approximation results. *Journal of Combinatorial Optimization*, *29*(1), 276–307.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, *25*(2), 163–177.

Crucitti, P., Latora, V., & Marchiori, M. (2004). Model for cascading failures in complex networks. *Physical Review E*, *69*, 045104.

Duarte, A., Sánchez-Oro, J., Resende, M. G. C., Glover, F., & Martí, R. (2015). Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. *Information Sciences*, *296*, 46–60.

Erdös, P., & Rényi, A. (1960). On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, *5*(1), 17–60.

Feige, U., & Mahdian, M. (2006). Finding small balanced separators. In Proceedings of the thirty-eighth annual ACM symposium.

Feo, T. A., & Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, *8*(2), 67–71.

Feo, T. A., Resende, M. G. C., & Smith, S. H. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, *42*(5), 860–878.

Garey, M., & Johnson, D. (1979). *Computers and intractability - A guide to the theory of NP-completeness*. San Francisco, CA: Freeman.

Glover, F. (1996). *Tabu search and adaptive memory programming – Advances, applications, and challenges*. Dordrecht, MA: Kluwer Academic Publishers.

Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093-2229). Boston, MA: Springer.

Glover, F., Laguna, M., & Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, *29*(3), 653–684.

Glover, F. W., & Kochenberger, G. A. (Eds.). (2006). Handbook of metaheuristics (Vol. 57). Springer Science & Business Media.

Hansen, P., & Mladenović, N. (2014). Variable neighborhood search. In *Search methodologies* (pp. 313–337). Boston, MA: Springer.

Laguna, M., & Martí, R. (1999). Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, *11*(1), 44–52.

Lee, J., Kwak, J., Lee, H. W., & Shroff, N. B. (2017, March). *Finding minimum node separators: A markov chain monte carlo method*. Paper presented at the Drcn 2017 - Design of Reliable Communication Networks; 13th International Conference (pp. 1–8).

Li, R., Hu, S., Zhang, H., & Yin, M. (2016). An efficient local search framework for the minimum weighted vertex cover problem. *Information Sciences*, *372*, 428–445.

Mohamed-Sidi, M. (2014). *K-separator problem (Problème de k-Sèparateur)*. (Unpublished doctoral dissertation). Telecom & Management SudParis, Èvry, Essonne, France.

Newman M. E. J. (2008) Mathematics of Networks. In P. Macmillan, (Ed) *The New Palgrave Dictionary of Economics*. Palgrave Macmillan, London.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The pagerank citation ranking: Bringing order to the web* (1999-66). Stanford InfoLab.

Quintana, J. D., Sánchez-Oro, J., & Duarte, A. (2016). Efficient greedy randomized adaptive search procedure for the generalized regenerator location problem. *International Journal of Computational Intelligence Systems*, *9*(6), 1016–1027.

Sánchez-Oro, J., Laguna, M., Martí, R., & Duarte, A. (2016). Scatter search for the bandpass problem. *Journal of Global Optimization*, *66*(4), 769–790.

Scott, J., & Carrington, P. J. (2011). The *SAGE handbook of social network analysis*. SAGE publications.

Wachs, M., Grothoff, C., & Thurimella, R. (2012). Partitioning the internet. In 2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS) (pp. 1-8). IEEE.

Yannakakis, M. (1981). Node-deletion problems on bipartite graphs. *SIAM Journal on Computing*, *10*(2), 310–327.

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Management Science*, *17*(11), 712–716.

## AUTHOR BIOGRAPHIES

**Sergio Pérez-Peló** was born in Madrid (Spain) on June 21, 1995. He has achieved his double degree in Computer Sciences and Software engineering from the Universidad Rey Juan Carlos (URJC) in 2017 and his M.S in Cibersecurity in 2018 from the Universidad Oberta de Cataluña (UOC). He is actually working in his PhD in URJC as a member of the Group for Research on Algorithms For Optimization (GRAFO). His main research interests focus on Artificial Intelligence and Operations Research, focusing in Social Network Analysis and real-world optimization problems.

**Jesús Sánchez-Oro** was born in Madrid (Spain) on December 31, 1987. He holds a degree in Computer Science from the Universidad Rey Juan Carlos (2010), his Master's degree in Computer Vision from the Universidad Rey Juan Carlos in 2011, and his Ph.D. in Computer Science in 2016 from the Universidad Rey Juan Carlos. He is visiting professor at the Computer Science Department, and he is a member of the Group for Research on Algorithms For Optimization (GRAFO). His main research interests focus on Artificial Intelligence and Operations Research, specially in heuristics and metaheuristics for solving hard optimization problems.

**Abraham Duarte** was born in Hervás (Cáceres, Spain) on October 24, 1975. He received his M.S. degree in Physics Sciences (Electronic Speciality) from the Universidad Complutense de Madrid (UCM) in 1998 and his Ph.D. in Computer Science in 2004 from the Universidad Rey Juan Carlos (URJC). He is Full Professor at the Computer Science Department where is the leader of the Group for Research on Algorithms For Optimization (GRAFO). His main research interests focus on the interface among Computer Science, Artificial Intelligence and Operations Research. Most of his publications deal with the development of metaheuristics procedures for optimization problems.

**How to cite this article:** Pérez-Peló S, Sánchez-Oro J, Duarte A. Finding weaknesses in networks using Greedy Randomized Adaptive Search Procedure and Path Relinking. *Expert Systems*. 2020;37:e12540. https://doi.org/10.1111/exsy.12540

# Chapter 8

# On the Analysis of the Influence of the Evaluation Metric in Community Detection over Social Networks

The journal paper associated to this part is:

- Pérez-Peló, S., Sánchez-Oro, J., Martín-Santamaría, R., & Duarte, A. (2019). "On the Analysis of the Influence of the Evaluation Metric in Community Detection over Social Networks". Electronics, 8(1), 23. (doi:10.3390/electronics8010023).
- Status: **Published.**
- Impact Factor (JCR 2018): 1.764
- Subject Category: Engineering, Electrical & Electronic. Ranking 154/266 (Q3)

*electronics*

MDPI

*Article*

# On the Analysis of the Influence of the Evaluation Metric in Community Detection over Social Networks

**Sergio Pérez-Peló, Jesús Sánchez-Oro *, Raúl Martín-Santamaría and Abraham Duarte**

Department Computer Sciences, Universidad Rey Juan Carlos, 28933 Móstoles, Spain;
sergio.perez.pelo@urjc.es (S.P.-P.); raul.martin@urjc.es (R.M.-S.); abraham.duarte@urjc.es (A.D.)
* Correspondence: jesus.sanchezoro@urjc.es

Received: 18 October 2018; Accepted: 20 December 2018; Published: 24 December 2018


**Abstract:** Community detection in social networks is becoming one of the key tasks in social network analysis, since it helps with analyzing groups of users with similar interests. As a consequence, it is possible to detect radicalism or even reduce the size of the data to be analyzed, among other applications. This paper presents a metaheuristic approach based on Greedy Randomized Adaptive Search Procedure (GRASP) methodology for detecting communities in social networks. The community detection problem is modeled as an optimization problem, where the objective function to be optimized is the modularity of the network, a well-known metric in this scientific field. The results obtained outperform classical methods of community detection over a set of real-life instances with respect to the quality of the communities detected.

**Keywords:** social network; community detection; metaheuristic; optimization; GRASP

## 1. Introduction

The evolution of social networks in the last few decades has aroused the interest of scientists from different and diverse areas, from psychology to computer sciences. Millions of people constantly share all their personal and professional information in several social networks [1]. Furthermore, social networks have become one of the most used information sources, mainly due to their ability to provide the user with real-time content. Social networks are not only a new way of communication, but also a powerful tool that can be used to gather information related to relevant questions. For instance, which is the favourite political party for the next elections?, what are the most commented on movies in the last year?, which is the best rated restaurant in a certain area?, etc.

Extracting relevant information from social networks is a matter of interest mainly due to the huge amount of potential data available. However, traditional network analysis techniques are becoming obsolete because of the exponential growth of the social networks, in terms of the number of active users.

The analysis of social networks has become one of the most popular and challenging tasks in data science [2]. One of the most tackled problems in social networks is the analysis of the relevance of the users in a given social network [3]. The relevance of a user is commonly related to the number of followers or friends that the user has in a certain social network. However, this concept can be extended since a user may be relevant not only if he/she is connected with a large number of users, but also with users that are relevant too. Several metrics have been proposed for analyzing the relevance of a user in a social network, with PageRank emerging as one of the most used [4]. Furthermore, it is interesting to know in advance which users will be the most relevant ones before they become influential [5]. Finally, in the field of marketing analysis, there is special interest in generating the profile of a user given a set of tweets written by that user [6].

Evaluating the relevance of a user has evolved into a more complex problem that consists of detecting specific users (often named influencers) with certain personal attributes that can be personal (credibility or enthusiasm) or related to their social networks (connectivity or centrality). These attributes allow them to influence a large number of users either directly or indirectly [7].

Another important problem regarding the influence of people in other users is the analysis of sentiments in social networks. It is focused on finding out what people think about a certain topic by analyzing the information they post in social networks. We refer the reader to [8] to find a complete survey on sentiment analysis techniques.

The previously described problems deal with only individual users. Nevertheless, some problems also exist related to the structure of the network, devoted to finding specific attributes and properties that can help to infer additional information of the whole social network. In this context, community detection emerges as one of the most studied problems.

Most of the social networks present a common feature named community structure [9]. Networks with this property have the capacity to be divided into groups in such a way that the connections among users in the same group are dense, while connections among users in different groups are sparse. Connections among users can represent different features depending on the social network and the user profile, i.e., from professional relationships to friendships or hobbies in common. Community detection tasks are devoted to finding and analyzing these groups in order to better understand and visualize the structure of network and the relationships among their users.

Performing community detection algorithms over current social networks requires a huge computational effort mainly due to their continuous growth. Furthermore, since these networks are constantly changing (new friendships, mentions to users, viral information, etc.), it is interesting to perform the community detection in the shortest possible computing time, producing real-time information. These features make traditional exact methods not suitable for the current size of social networks, requiring heuristic algorithms in order to accelerate the process without losing quality. Recent works have tackled the community detection problem from a non-exact perspective in order to generate high quality solutions in short computing time [10]. Several studies are devoted to reducing the computational effort for detecting communities in social networks [11,12]. When comparing traditional algorithms over modern large social networks, it can be seen that some of the algorithms require more than 40,000 s for networks with approximately 10,000 nodes, and they are not able to provide a solution after 24 h computing for networks with more than 50,000 nodes.

The growth of social networks complicates their representation and understanding. The communities of a social network usually summarize the whole network but reduce its size and, therefore, makes it easier to analyze. In addition, detecting communities in social networks has several practical applications. Recommendation systems leverage the data of similar users in order to suggest content that can be interesting for them. In order to find similar users in a network, we can simply perform a community detection over the network [13], improving the results of the recommendation system. Communities in social networks also identify people with similar interests, allowing us to evaluate the popularity of a political party [14], or even to detect radicalism in social networks [15].

Although several community detection algorithms exist that have been proposed with the aim of identifying similar users in networks, most of the available algorithms have been designed for optimizing a specific objective function, it being hard to be adapted to a different one (see Section 3 for a detailed description of the considered algorithms). However, the continuous evolution of this area results in a continuous proposal of new metrics that better evaluates the community structure of a given network. This work presents an efficient and versatile algorithm that can be easily adapted to different optimization metrics. To the best of our knowledge, this is the first algorithm based on classical metaheuristics for detecting communities in social networks. The success of this proposal opens a new research line for modeling social network problems as optimization problems, with the aim of applying metaheuristics for solving them.

The main contributions of this work are itemized as follows:

- A new solution representation for the community detection problem is presented.
- A new constructive procedure for generating partitions based on the Greedy Randomized Adaptive Search Procedure (GRASP) is introduced.
- The local search proposed is able to handle not only the change of community of certain nodes, but also the creation and elimination of communities, increasing the portion of search space explored.
- A classical metaheuristic algorithm, GRASP, is adapted to be competitive in social network analysis.
- The proposed algorithm is highly scalable, being easily adapted to be executed in distributed systems.
- A thorough comparison with the most used methods in community detection is provided, analyzing the advantages and disadvantages of each one of them.
- A comparison between two of the most extended metrics is presented, one of them for optimization and the other for evaluation. Furthermore, the evaluation of the metric used for optimization is also performed, with the aim of testing its suitability for the networks considered.

The remainder of the paper is structured as follows: Section 2 formally defines the problem considered as well as the metrics proposed for the evaluation of solutions; Section 3 presents a thorough description of the classical algorithms proposed for detecting communities in social networks; Section 4 presents the new procedure proposed for detecting communities; Section 5 introduces the computational experiments performed to test the quality of the proposal; and finally Section 6 draws some conclusions on the research.

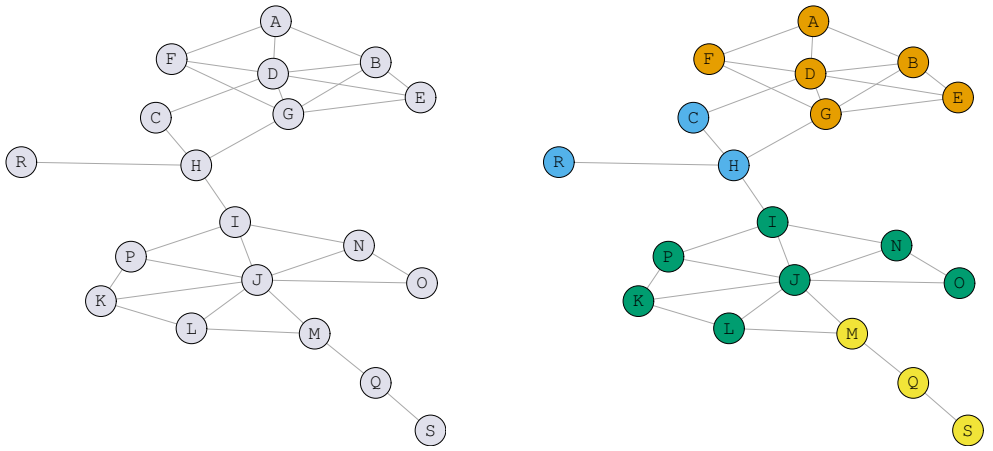
## 2. Problem Statement

A social network is represented as a graph $G = V, E$, where the set of vertices $V$, with $|V| = n$, represents the users of the network and the set of edges $E$, with $|E| = m$, represents relations between users belonging to the network. An edge $(v_1, v_2) \in E$, with $v_1, v_2 \in V$ can represent different types of relations depending on the social network under consideration. For example, on Twitter, a relation represents that a user follows/is followed by another user, or if there has been some interaction (comment, like, share, etc.) between users, while, on LinkedIn, it represents a professional relationship.

This work is focused on the Community Detection Problem (CDP), which involves grouping users of a social network into communities. A desirable community in a social network is densely connected to the nodes in the same community and sparsely connected (or even unconnected) to nodes in other communities. Therefore, the main objective is to obtain groups or communities of users that are similar and, at the same time, different to the users in other communities with respect to a certain criterion.

A solution for the CDP is represented by a set of decision variables $S$, with $|S| = n$, where $S_v = j$ indicates that vertex $v$ is assigned to community $j$ in solution $S$. Therefore, the community for each vertex $v \in V$ is represented by the corresponding decision variable $S_v$. It is worth mentioning that the number of communities is not predefined in advance. Therefore, a solution where all users are assigned to the same community is feasible. Similarly, a solution where each user is assigned to a different community is also valid.

Figure 1a shows an example graph with 19 vertices and 31 edges derived from a social network. In this example, an edge represents a friendship relationship between two users; for instance, users `A` and `B` are friends, while users `A` and `C` are not friends, but they have a friend in common, which is vertex `D`.




(**a**) (**b**)

**Figure 1.** (**a**) example of a graph derived from a social network and (**b**) a possible solution for the community detection (each community is represented with a different color).

Figure 1b shows a possible solution $S$ for the community detection problem, where each community is represented with a different color. Notice that, in this example, the number of communities is 4. For the sake of completeness, we report in Table 1 the community to which each vertex has been assigned. For example, vertex $A$ belongs to community 1 ($S_A = 1$), vertex $C$ to community 2 ($S_C = 2$), and so on).

**Table 1.** Community assigned to each vertex in the solution depicted in Figure 1b.

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 4 | 2 | 4 |

The CDP then consists in finding a solution $S^\star$ that maximizes a certain objective function value, denoted as $f$. In mathematical terms,

$$S^\star \leftarrow \arg\max_{S \in \mathbb{S}} f(S),$$

where $\mathbb{S}$ is the set of all possible solutions for a given social network.

There exists a large variety of quality metrics that can be used as objective functions for finding high quality solutions. Most of the metrics are focused on maximizing the density of intra-community edges (those connecting vertices of the same community) while minimizing inter-community edges (those connecting vertices in different communities).

Notice that the metric considered for optimization is not required from the ground truth since we are dealing with an unsupervised clustering problem [16]. However, some metrics used for evaluating the quality of an algorithm assume that the optimal partition (ground truth) is known beforehand, considering that an algorithm is better if it minimizes the distance from the generated partition to the optimal one. One example of such a metric is the Omega-Index (see [10] for further details). In this work, we consider an alternative approach where the optimal partition is not known.

In this research, we evaluate two metrics that have traditionally been used for optimizing the quality of a solution for the CDP: conductance and modularity [17]. The conductance metric is normalized in the range 0–1, while modularity can take on negative values, ranging from $-1/2$ to 1. For both metrics, the largest value indicates the value of the optimal partition, while random assignment of users to communities is expected to produce values close to the smallest value.

Notice that, in some cases, it is not possible to reach the optimal score due to the internal structure of the network.

The first metric considered in this paper is known as conductance [18]. Given a network $G$, a solution $S = \{S_1, S_2, \ldots, S_n\}$, and a specific community $k$, its conductance, $Cn(k, S, G)$, is defined as the number of edges that connect vertices of different communities divided by the minimum between the number of edges with, at least, an endpoint in the community and the number of edges without an endpoint in the community. More formally,

$$Cn(k, S, G) = \frac{|(v, u) \in E : S_v = k \wedge S_u \neq k|}{\min\{|(v, u) \in E : S_v = k \vee S_u = k|, |(v, u) \in E : S_v \neq k \wedge S_u \neq k|\}}.$$

Then, the conductance of a complete solution $Cn(S, G)$ is evaluated as the average conductance for all the communities in the graph. In mathematical terms,

$$Cn(S, G) = \sum_{k=1}^{k_{max}} Cn(k, S, G),$$

where $k_{max}$ is the number of communities in the incumbent solution $S$, i.e., $k_{max} = \max\{S_1, S_2, \ldots, S_n\}$.

Let us illustrate the computation of this metric with the example depicted in Figure 1b. The conductance of community 2 is evaluated as follows:

$$\begin{aligned} Cn(k, S, G) &= \frac{|(\mathtt{C},\mathtt{D}), (\mathtt{H},\mathtt{G}), (\mathtt{H},\mathtt{I})|}{\min\{|(\mathtt{C},\mathtt{D}), (\mathtt{C},\mathtt{H}), (\mathtt{H},\mathtt{G}), (\mathtt{H},\mathtt{I}), (\mathtt{R},\mathtt{H})|, |E \backslash (\mathtt{C},\mathtt{D}), (\mathtt{C},\mathtt{H}), (\mathtt{H},\mathtt{G}), (\mathtt{H},\mathtt{I}), (\mathtt{R},\mathtt{H})|\}} \\ &= \frac{3}{\min\{5, 26\}} = \frac{3}{5} = 0.6. \end{aligned}$$

Similarly, the conductance of the remaining communities (1, 3, and 4) are $Cn(1, S, G) = 0.15$, $Cn(3, S, G) = 0.21$, and $Cn(4, S, G) = 0.5$, respectively. Therefore, $Cn(S, G) = 0.22$.

In order to have a direct comparison with other metrics, it is usually reported the opposite of the conductance evaluated as $\overline{Cn}(G) = 1 - Cn(G)$. In the aforementioned example, this value is then $\overline{Cn}(G) = 0.78$. Then, the objective is to maximize this value to produce high quality solutions.

The second metric is the modularity [19] that evaluates, for each edge connecting vertices in the same community, the probability of the existence of that edge in a random graph. The modularity of a community $k$ over a solution $S$ for a network $G$ is defined as follows:

$$\begin{aligned} Md(k, S, G) &= (e_{jj} - a_j^2), \\ e_{kk} &= \frac{|\{(v, u) \in E : S_v = S_u = k\}|}{|E|}, \\ a_k &= \frac{|\{(v, u) \in E : S_v = k\}|}{|E|}, \end{aligned}$$

where $k_{max}$ is the number of communities in the solution, $e_{kk}$ is the percentage of intra-community edges (with respect to the whole set of edges) in the community $k$, and $a_k$ is the percentage of edges with at least one endpoint in $k$. Then, the modularity of the complete solution $S$ is formally defined as:

$$Md(S, G) = \sum_{k=1}^{k_{max}} Md(k, S, G).$$

Let us illustrate how we can compute this metric for the example depicted in Figure 1b. Specifically, the modularity of community 2 is evaluated as:

$$e_{22} = \frac{|(\mathtt{C},\mathtt{H}),(\mathtt{H},\mathtt{R})|}{|E|} = \frac{2}{31},$$
$$a_2 = \frac{|(\mathtt{C},\mathtt{D}),(\mathtt{H},\mathtt{G}),(\mathtt{H},\mathtt{I})|}{|E|} = \frac{3}{31},$$
$$Md(2,S,G) = e_{22} - a_2^2 = \frac{2}{31} - \left(\frac{3}{31}\right)^2 = 0.06.$$

Similarly, the modularity of the remaining communities (1, 3, and 4) are $Md(1,S,G) = 0.35$, $Md(3,S,G) = 0.34$, and $Md(4,S,G) = 0.06$, respectively. Then, the modularity of the complete solution depicted in Figure 1b is $Md(S,G) = 0.75$.

The main disadvantage of modularity metric is its resolution metric. As stated in [20], optimizing modularity may lead the algorithm to miss substructures of the network, thus ignoring the detection of some sub-communities. This behavior does not depend on a particular network structure, but on the ratio between intra-community relations versus the total number of relations in the network.

The majority of the traditional algorithms for community detection considers this metric as the one to be optimized in order to find high-quality partitions in communities, since it does not fall in trivial solutions (i.e., those with a single community for all the network, or those with a different community for each node of the network), being a robust metric to be considered for optimization.

## 3. Algorithms for Community Detection

Several algorithms have been proposed for detecting communities in social networks (see, for instance, [9,21,22]). Community detection algorithms can be classified into two different classes: agglomerative or divisive. On the one hand, agglomerative methods start from a solution where each vertex is located in a different community and try to optimize a given objective function by joining two or more communities at each step. On the other hand, divisive methods start from a solution with all the vertices located in a single community, and the objective function is optimized by dividing one or more communities in each step.

Most of the algorithms are not exact procedures, since in most of the networks it is not feasible to find the optimal solution in a reasonable time, mainly due to the number of users in the network [19,23]. This section is devoted to describing the most used algorithms in the state of the art for the CDP, in order to have a framework of comparison for the algorithm presented in this work.

### *3.1. Edge-Betwenness (EB)*

The idea of the Edge-Betweenness algorithm [9] relies on identifying those vertices that appear in the majority of the paths in the graph. Specifically, authors define the betweenness of an edge as the number of shortest paths between pairs of vertices that contains the edge under evaluation. Therefore, groups or communities are generated by removing the edge with the largest edge betweenness value in each step. This algorithm has a computational complexity of $O(m^2 n)$.

### *3.2. Fast-Greedy (FG)*

The Fast-Greedy algorithm [21] is focused on optimizing the modularity of the solutions generated. This agglomerative method starts from a solution where each vertex is located in a different community and iteratively joins the two communities that produce the solution with maximum modularity value. The optimization and data structures presented in the original work reduces the computational complexity of the algorithm to $O(n \cdot m \cdot \log n)$.

### *3.3. Infomap (IM)*

The Infomap algorithm [22] proposed a fast stochastic and recursive search method which is based on joining neighbor vertices into the same community. The method starts with each vertex located in a different community. Then, it randomly selects a vertex and assigns it to the community that minimizes the map equation, presented in the original work [22], which is an efficient estimation of the optimality of a certain partition. Then, the method creates a new network where the new vertices are the communities detected until now. The algorithm stops when no changes are produced in the communities.

### *3.4. Label Propagation (LP)*

The Label Propagation algorithm [24] initially assigns a different label to each vertex of the graph. Then, in each iteration, the algorithm modifies the label of each vertex depending on the label assigned to its adjacent vertices. Specifically, a vertex receives the most common label in all its neighbors, stopping when no changes are produced in the labels of the graph. At the end of the process, the label of a vertex identifies the community to which that vertex belongs to. It is worth mentioning that this algorithm does not consider any quality metric, since the optimization is performed with respect to the labels of the neighbors of each vertex. The main advantage of this algorithm is its computational complexity of $O(n + m)$.

### *3.5. Multi-Level (ML)*

The Multi-Level algorithm [11] is designed for detecting communities in large networks focused on optimizing the modularity of the solution. The algorithm consists of two well-differenced phases. The first phase starts by assigning each vertex to a different community. In each step, the method evaluates, for each vertex $v$, the profit of merging it in the community of each adjacent vertex in terms of modularity. Then, vertex $v$ is inserted in the community that produces the maximum profit, only if the profit is positive, stopping when no improvement can be found. The second phase is based on creating a new network where each node represents a community, where the weights of the edges identifies the sum of the weights of the edges between nodes in the corresponding communities. Then, the first phase is applied again to this new network. The two phases are iteratively applied until no changes are performed in the communities detected. The main advantage of this algorithm relies on the efficient evaluation of the profit in terms of modularity, resulting in a linear complexity when the number of vertices is similar to the number of edges. However, if the graph is fully connected, the algorithm presents a complexity of $O(n^2)$.

### *3.6. Spinglass (SG)*

The Spinglass algorithm [25] is inspired by statistical physics, in particular in the Potts model. The algorithm simulates that each vertex of the graph is a particle that can have any of the spin states. Additionally, the edges represent the interactions between particles, which influence the vertices in changing their spin state or not. Then, the method simulates the model a predefined number of iterations. Finally, the spin state of each particle identifies the community of each vertex. It is a computationally demanding algorithm mainly due to the simulation and it is not deterministic.

### *3.7. Walktrap (WT)*

The Walktrap algorithm [12] relies on the idea that random walks over a graph usually get caught in the most densely connected parts, which are often the communities of the graph. Then, authors define a distance that can be evaluated efficiently. This distance is then used in a hierarchical clustering algorithm that iteratively merges vertices into communities, creating a dendogram of the community structure of the graph. The complexity of the method is $O(mnH)$, $H$ being the height of the dendogram. The worst case corresponds to a dendogram of height $n$, resulting in a complexity of $O(mn^2)$, which

corresponds to very dense graphs. However, real-life social networks are usually sparse, and when $H$ is small and the dendogram is balanced, the complexity is reduced to $O(n^2 \log n)$.

### *3.8. Louvain (CL)*

The Louvain algorithm [11] proposed a method divided into two phases: the first phase considers that each node is initially a community in a network. Then, it joins communities looking for the modularity metric optimization, and stops when a local maximum of modularity is reached. In the second phase, the algorithm builds a new graph with the communities obtained from the first phase, restarting with the first phase. According to [11], this method has a complexity of $O(n)$.

### *3.9. Evaluation of the Previous Methods*

This section is devoted to evaluating the results obtained by the previously described methods over an example graph that presents community structure [26] to provide an illustrative example. Figure 2 depicts the graphical results over the community detection in the graph, where each community is represented with a different color.

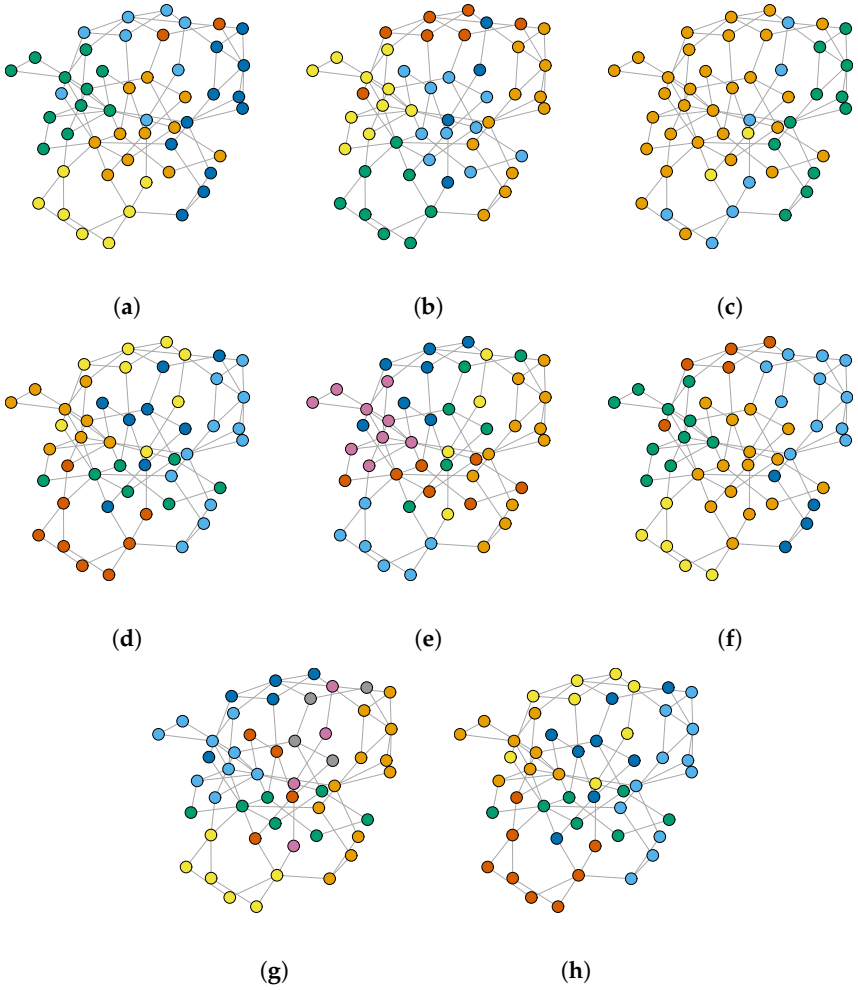



**Figure 2.** Comparison of the community detection of the described algorithms over a example graph with 50 nodes that presents community structure. (**a**) edge-betweenness; (**b**) fast-greedy; (**c**) label propagation; (**d**) Multi-Level; (**e**) Spinglass; (**f**) Walktrap; (**g**) Infomap; (**h**) Louvain.

As can be seen, the results are different for each algorithm. Additionally, Table 2 presents the results obtained by each considered algorithm over the example graph depicted in Figure 2, considering the two metrics described in Section 2 and the number of communities found.

**Table 2.** Evaluation of the solution generated by each algorithm over the example graph using the three considered metrics.

| Algorithm | Modularity | Conductance | Number of Communities |
|---|---|---|---|
| EB | 0.5245 | 0.5248 | 6 |
| FG | 0.5284 | 0.5306 | 6 |
| LP | 0.3596 | 0.4571 | 4 |
| ML | 0.5158 | 0.5191 | 6 |
| SG | 0.5257 | 0.5009 | 7 |
| WT | 0.4927 | 0.5260 | 6 |
| IM | 0.5231 | 0.4732 | 8 |
| CL | 0.5158 | 0.5191 | 6 |

First of all, we will analyze the modularity metric, since it is the most used metric in community detection optimization and, furthermore, it is the metric to be optimized in the current research. The best modularity value is obtained with the Fast-Greedy algorithm (0.5284), closely followed by Spinglass (0.5257), Edge Betweenness (0.5245) and Infomap (0.5231).

Analyzing the conductance value, the three considered algorithms present the same behavior as with modularity: Fast-Greedy is the best approach (0.5306), but now followed by Walktrap (0.5260) and then Edge Betweenness (0.5248). Notice that the differences among algorithms considering conductance are larger.

Finally, analyzing the number of communities detected, five out of the eight algorithms detect six communities, which seem to be the actual number of communities in the social network. The largest number of communities, 8, is found with the Infomap algorithm. These results suggests that artificially increasing the number of communities does not lead to better results.

## 4. Greedy Randomized Adaptive Search Procedure

Metaheuristics comprehend a set of approximate algorithms designed for solving hard combinatorial optimization problems for which traditional heuristic methods are not effective. These algorithms provide a general framework for creating new hybrid algorithms with concepts derived from artificial intelligence, biological evolution and statistical mechanisms [27].

Greedy Randomized Adaptive Search Procedure (GRASP) is a metaheuristic originally presented in [28] and formally defined in [29]. We refer the reader to [30] for a recent survey on this methodology. This metaheuristic can be divided into two main phases: solution construction and local improvement.

The first phase iteratively adds elements to an initially empty solution until it becomes feasible. The first element is usually selected at random, acting as a seed for the procedure. The algorithm then constructs a candidate list (*CL*) with all the elements that must be included in the solution. After that, a Restricted Candidate List (*RCL*) is created with the most promising elements of the *CL* according to a predefined greedy function. Then, in each iteration, an element is selected at random from the *RCL* and added to the solution under construction, updating the *CL* and *RCL* in each step until reaching a feasible solution.

The construction phase of the GRASP algorithm presents a random part devoted to increasing the diversity of the solutions generated. In particular, in the previous description, the random part relies on the random selection of the next element from the *RCL*. Therefore, most of the obtained solutions are not local optimum and can be improved by means of a local optimizer. The second phase of the GRASP algorithm is intended to find a local optimum of the solution generated, usually applying a local search method, although it can be replaced with a more complex optimizer, like Tabu Search or Variable Neighborhood Search, for instance [31–33].

The algorithm presented in this section is able to optimize any of the metrics defined in Section 1. However, heavily optimizing conductance usually leads to the trivial partition where all the vertices are in the same community. Therefore, the proposed algorithm is focused on optimizing the modularity, which has been traditionally considered as a good optimization metric.

Analyzing the related literature, most of the algorithms are designed for optimizing a specific objective function value. However, the versatility of the proposed algorithm allows it to be easily adapted to optimize either a new or a traditional metric, which converts it into a generic algorithm for finding community structures for any optimization metric.

Furthermore, the algorithm is proposed as a framework for detecting communities. It is easy to replace the constructive method or local search procedure proposed with a different one, or even embed a more complex local optimizer, such as Tabu Search [34], or Variable Neighborhood Search [35], among others.

### *4.1. Constructive Procedure*

The constructive procedure designed for the community detection problem, named *GRASPAGG* follows an agglomerative approach, where each element is initially located in a different community. Then, *GRASPAGG* iteratively joins two of the most promising communities with the objective of maximizing the modularity. Algorithm 1 shows the pseudocode of the *GRASPAGG* constructive method.

**Algorithm 1** *GRASPAGG*$(G, \alpha)$.

1: $S_v \leftarrow v \; \forall v \in V$
2: $CL \leftarrow \{1, 2, \ldots n\}$
3: **while** $CL \neq \emptyset$ **do**
4: $\quad g_{\min} \leftarrow \min_{j \in CL} Md(S, G, j)$
5: $\quad g_{\max} \leftarrow \max_{j \in CL} Md(S, G, j)$
6: $\quad \mu \leftarrow g_{\min} + \alpha \cdot (g_{\max} - g_{\min})$
7: $\quad RCL \leftarrow \{j \in CL \; : \; Md(S, G, j) \geq \mu\}$
8: $\quad j_1 \leftarrow Random(RCL)$
9: $\quad j_2 \leftarrow IdentifyBestJoin(S, j_1)$
10: $\quad S' \leftarrow Join(S, j_1, j_2)$
11: $\quad$ **if** $f(S') > f(S)$ **then**
12: $\quad\quad S \leftarrow S'$
13: $\quad\quad CL \leftarrow CL \setminus \{j_2\}$
14: $\quad$ **else**
15: $\quad\quad CL \leftarrow CL \setminus \{j_1\}$
16: $\quad$ **end if**
17: **end while**
18: **return** $S$

The method starts by assigning a different community to each node in the graph $G$ (step 1), creating the $CL$ with every community in the solution $S$ under construction (step 2). In other words, at the beginning of the construction, there will be $n$ nodes assigned to $n$ different communities. Then, the minimum ($g_{\min}$) and maximum ($g_{\max}$) values for the greedy function under evaluation are calculated (steps 4 and 5). We propose as a greedy function the modularity value of each community $j$. A threshold $\mu$ is evaluated (step 6) to construct the $RCL$ with the most promising candidates in $CL$ (step 7). The next steps select the two communities that will be joined in the current iteration. The first one, $j_1$, is selected at random from the $RCL$ (step 8). The second community $j_2$ is the one that maximizes the modularity of the resulting solution after joining communities $j_1$ and $j_2$ (step 9). If the method has found an improvement in the modularity after joining both communities, a new iteration is performed, updating the incumbent solution (step 12) and the candidate list (step 13); otherwise, the community $j_1$ is removed from the candidate list since any join involving $j_1$ produces a worse

solution. Finally, *GRASPAGG* stops when it is not possible to join two communities improving the modularity, returning the best solution found.

### *4.2. Local Optimization*

This section presents a local search procedure designed to find a local optimum for every solution constructed in the previous phase. In order to define a local search method, we firstly need to define the neighborhood in which the local optimum will be found. For this problem, we consider all the solutions that can be reached from a given solution $S$ by removing one node from its current community and inserting it in a different one. Specifically, after performing the move $Move(S, v, j)$, the vertex $v$ will be located at community $j$ (i.e., $S_v = j$).The neighborhood $N(S)$ is defined as:

$$N(S) \leftarrow \{Move(S, v, j) \; \forall v \in S \; : \; j \neq S_v\}.$$

It is worth mentioning that the number of communities is not a priori defined for the problem. Therefore, if $v$ is the last vertex in the community $j'$, then community $j'$ will disappear after performing the corresponding move. In the same line, the method also considers creating a new community for vertex $v$ if it improves the modularity. Thus, after performing the local search method, the number of communities may have varied either increasing or decreasing.

The next step for defining the local search method is the selection of the vertex to be moved to another community. For this purpose, we define a heuristic criteria based on the number of intra-community edges of the vertex under evaluation with respect to the total number of edges in the graph. Specifically, the local search method selects the vertex $v$ with the smallest ratio $r$ between number of edges in the same community and the total number of incident edges to $v$. More formally,

$$r(v, S) \leftarrow \frac{|(v, u) \in E \; : \; S_v = S_u|}{|E|} \quad \forall u, w \in V.$$

The local search method traverses all the nodes in the solution following an ascending order with respect to the previously defined criterion. Each node is moved from its current community to the one that maximizes the modularity among all the existing communities in the incumbent solution.

The proposed local search procedure follows a first improvement approach. In particular, given a solution $S$, this strategy scans its neighborhood $N(S)$ in search for the first solution $S' \in N(S)$ such that $f(S') > f(S)$. The method stops when no improvement is found after exploring the whole neighborhood.

### *4.3. Complexity Analysis*

The complexity analysis of the proposed algorithm can be split into two different stages: constructive and local improvement. Firstly, the complexity of the constructive procedure is analyzed.

The constructive procedure iterates until the candidate list is empty. One candidate is removed in each iteration, either due to the joining of two communities or because the candidate cannot be joined. Following a straightforward implementation, this method requires traversing all the nodes and all the edges, resulting in a complexity of $O(n \cdot m)$. However, we cache the degree of each node and its adjacents in efficient data structures when reading the social network (since the degree of a node will not change during the execution). Therefore, we reduce the complexity of generating the candidate list to $O(n)$. The method then iterates until no improvement in the modularity is found when joining two clusters. In the worst case, it requires performing $n - 1$ iterations, resulting in a total complexity of $O(n^2)$. However, we select in each iteration the community with the smallest modularity value, finding improvements in the first iterations. Therefore, the complexity of this stage is $O(\log n)$, mainly due to the cost of maintaining the communities sorted by modularity. Finally, the complexity of the complete constructive procedure is $O(n \log n)$.

The second stage corresponds to the local search procedure, where each node is considered to be inserted in each community. Therefore, it presents a complexity of $O(n \cdot k)$, $k$ being the number of communities. However, it is worth mentioning that notation $O()$ refers to the worst case, and it is possible to optimize the search with the aim of avoiding the worst case. In particular, the local search method evaluates each node $v \in S$ following an ascending order (worst nodes first) with respect to the ratio $r(v, S)$ defined in Section 4.2. This ordering is a heuristic that minimizes the number of movements performed before finding an improvement, since it is presumed that nodes with a small ratio value are not located in the best community. Therefore, on average, the algorithm complexity linearly grows with the problem size.

Analyzing the complexity of both stages, the resulting GRASP algorithm presents a global complexity of $O(n \log n)$ plus the complexity of the local search, which is linear (in the average case) with respect to the problem size, resulting in a final complexity of $O(n \log n)$ per iteration.

## 5. Computational Results

This section is devoted to analyzing the quality of the proposed algorithm when compared with the most popular community detection algorithms presented in Section 3. Since most of the algorithms are focused on optimizing the modularity, the evaluation of the quality must be performed over a different metric. In this work, we consider the conductance with the aim of testing the robustness of the methods when including one additional metric. We also consider the modularity value obtained with each algorithm, although it should not be taken into account in the evaluation of the quality of the community detection. However, we consider that it is interesting to analyze how far an algorithm is able to optimize the detection considering the modularity value. The proposed algorithm has been implemented in Java 8 and the experiments have been conducted in an Intel Core 2 Duo E7300 2.66 GHz with 4 GB RAM.

The instances used for the experiment have been extracted from the Twitter SNAP dataset [36] and from Network repository [37]. Specifically, we have selected 100 instances with vertices ranging from 50 to 400 that represent the ego-network of several Twitter users (data is anonymized in the dataset, and the ego user is not included in it) and other interactions between users from other networks.

The first experiment is devoted to tuning the $\alpha$ parameter of the *GRASPAGG* procedure. This parameter controls the degree of randomness of the method: on the one hand, $\alpha = 0$ results in a totally random method, while $\alpha = 1$ considers a completely greedy method. Therefore, it is interesting to test values distributed in the range 0–1 to analyze whether the best results for the CDP are obtained with a small or large percentage of randomness/greediness in the construction. In this experiment, we have considered $\alpha = \{0.25, 0.50, 0.75, RND\}$, where *RND* indicates that a random value of $\alpha$ is selected for each construction. This experiment has been conducted over a subset of 20 representative instances in order to avoid overfitting.

Table 3 reports the results obtained with the different values of $\alpha$. Specifically, two statistics are considered: Avg., the average of the best modularity value obtained for each instance, and #Best, the number of times that an algorithm matches that best solution. Notice that conductance is not included in this preliminary experiment since it is performed for tuning the algorithm, and conductance should be used only for evaluating its quality.

**Table 3.** Results obtained by the *GRASPAGG* algorithm considering different values for $\alpha$ parameter.

| $\alpha$ | Avg. Modularity | #Best |
|---|---|---|
| 0.25 | 0.31961 | 6 |
| 0.50 | 0.32019 | 5 |
| 0.75 | 0.32063 | 4 |
| RND | 0.32080 | 9 |

Analyzing the results presented in Table 3, we can clearly see that $\alpha = RND$ is able to obtain the largest number of best solutions (9 out of 20). However, the quality of the solutions provided when it does not match the best solution is considerably worse than the other $\alpha$ values. If we now analyze the average modularity value and the number of times that the algorithms reach the best solution, we can conclude that a random value for $\alpha$ in each iteration obtains the best results, closely followed by $\alpha = 0.75$. Therefore, the final version of the algorithm is configured with $\alpha = RND$.

Once the best $\alpha$ parameter for the proposed algorithm has been adjusted, it is necessary to compare its performance with the most used community detection algorithms found in the literature. Specifically, we have included in the comparison the algorithms described in Section 3: Edge Betweenness (EB), Fast-Greedy (FG), Label Propagation (LP), Multi-level (ML), Walktrap(WT), InfoMap (IM) and Louvain (CL). Table 4 shows the aforementioned comparison.

**Table 4.** Comparison of the considered metrics over all the algorithms presented in Section 3 and the proposed *GRASPAGG* method.

| Algorithm | Modularity | | Conductance | |
|---|---|---|---|---|
| | **Avg.** | **#Best** | **Avg.** | **#Best** |
| EB | 0.20176 | 0 | 0.03363 | 7 |
| FG | 0.29441 | 3 | 0.44062 | 17 |
| LP | 0.15170 | 2 | 0.43734 | 6 |
| ML | 0.28843 | 2 | 0.43433 | 19 |
| WT | 0.26663 | 2 | 0.25224 | 7 |
| IM | 0.20611 | 2 | 0.37829 | 16 |
| CL | 0.31181 | 33 | 0.48002 | 9 |
| *GRASPAGG* | 0.31331 | 78 | 0.49483 | 37 |

Firstly, the analysis will be focused on the modularity value. In particular, *GRASPAGG* is able to obtain a slightly better result than Louvain (0.31331 vs. 0.31181), which is the second best approach. However, regarding the number of best solutions found, we can clearly see the superiority of our proposal, doubling the number of best solutions found with the Louvain method. Then, the next best algorithm is Fast-Greedy, achieving a total of three best solutions found. It is worth mentioning that the remaining algorithms are far from the results with respect to modularity. This can be partially explained since not all the algorithms are focused on optimizing modularity. Therefore, we need to consider an additional metric to have a fair comparison.

The conductance metric should be the one considered for evaluating the performance of each algorithm, since it is an objective metric that has not been used for any of the compared methods. The results show that the trend continues but with more significant differences among methods. Again, the best average conductance value is obtained by *GRASPAGG* (0.49483), and the second best value is reached with the Louvain method (0.48002). However, the differences between both results are larger, confirming the superiority of our proposal. Additionally, the *GRASPAGG* method is able to reach 37 out of 100 instances, while Louvain only obtains nine. The third best algorithm is again Fast-Greedy, with a conductance of 0.44062. Notice that, although the conductance of Louvain is better than the one of Fast-Greedy, the latter is able to achieve a larger number of best solutions found (17 vs. 9). The same conclusions can be derived when analyzing Multi-level and Infomap algorithms.

Finally, we have conducted different statistical tests to validate the results reported in Table 4. In particular, we have performed the Friedman non-parametric statistical test with all the individual values obtained in the previous experiment to confirm whether there exist statistically significant differences among the compared algorithms or not. The Friedman test ranks each algorithm according to the conductance value obtained, giving rank 1 to the best algorithm, 2 to the second one, and so on. The larger the differences in the average, the smaller the $p$-value will be. The average ranks values obtained with this test are *GRASPAGG* (1.70), CL (2.47), LP (2.73), FG (2.88), ML (3.30), IM (4.12), WT (5.04), and EB (5.74), resulting in a $p$-value smaller than 0.00001. Additionally, we have performed

the Wilcoxon signed rank test for the best two algorithms (i.e., *GRASPAGG* and CL). The resulting $p$-value smaller than 0.00001 confirms that there are statistically significant differences between both algorithms, then supporting the quality of the proposed GRASP method.

## 6. Conclusions

This paper has proposed a new metaheuristic method for community detection in a social network based on Greedy Randomized Adaptive Search Procedure methodology. The problem is addressed by optimizing the modularity metric, which is a robust metric to evaluate the quality of a partition in a social network.

The proposed algorithm is composed of two heuristic strategies. On the one hand, a constructive procedure based on an agglomerative scheme is proposed, which tries to balance the randomness and greediness of the search. On the other hand, an improvement procedure based on a problem-dependent neighborhood definition is presented. The main advantage of this local search is not only to find the best community for each node, but also to create and destroy communities in the incumbent solution.

The experiments firstly select the best value for the algorithm parameters and then compare its results with the most used algorithms for community detection found in the literature. The computational results show how *GRASPAGG* is able to obtain better results in both metrics than the previous methods. Additionally, the statistical tests performed over the evaluation metric support the quality of the proposed algorithm, emerging as a competitive algorithm for detecting communities in social networks.

**Author Contributions:** R.M.-S. and S.P.-P. have implemented the algorithm and revised the paper, J.S.-O. and A.D. have designed the algorithms and written and revised the manuscript.

**Funding:** This work has been partially founded by the Ministerio de Economía y Competitividad with Grant No. TIN2015-65460-C2-2-P.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Borgatti, S.P.; Everett, M.G.; Johnson, J.C. *Analyzing Social Networks*; Sage: Thousand Oaks, CA, USA, 2018.
2. Dorogovtsev, S.N.; Mendes, J.F.F. *Evolution of Networks: From Biological Nets to the Internet and WWW*; Oxford University Press: Oxford, UK, 2003.
3. Tang, J.; Sun, J.; Wang, C.; Yang, Z. Social influence analysis in large-scale networks. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 807–816.
4. Page, L.; Brin, S.; Motwani, R.; Winograd, T. *The PageRank Citation Ranking: Bringing Order to the Web*; Technical Report; Stanford InfoLab: Stanford, CA, USA, 1999.
5. Almgren, K.; Lee, J. An empirical comparison of influence measurements for social network analysis. *Soc. Netw. Anal. Min.* **2016**, *6*, 52. [CrossRef]
6. Ikeda, K.; Hattori, G.; Ono, C.; Asoh, H.; Higashino, T. Twitter user profiling based on text and community mining for market analysis. *Knowl.-Based Syst.* **2013**, *51*, 35–47. [CrossRef]
7. Gladwell, M. *The Tipping Point—How Little Things Can Make a Big Difference*; Little Brown and Company: Boston, MA, USA, 2000.
8. Pang, B.; Lee, L. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.* **2008**, *2*, 1–135. [CrossRef]
9. Girvan, M.; Newman, M.E.J. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [CrossRef]
10. González-Pardo, A.; Jung, J.J.; Camacho, D. ACO-based clustering for Ego Network analysis. *Future Gen. Comput. Syst.* **2017**, *66*, 160–170. [CrossRef]
11. Blondel, V.D.; Guillaume, J.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Sta. Mech. Theory Exp.* **2008**, *2008*, P10008. [CrossRef]
12. Pons, P.; Latapy, M. Computing Communities in Large Networks Using Random Walks. *J. Graph Algorithms Appl.* **2006**, *10*, 191–218. [CrossRef]

13. Cao, C.; Ni, Q.; Zhai, Y. An Improved Collaborative Filtering Recommendation Algorithm Based on Community Detection in Social Networks. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 11–15 July 2015; Silva, S., Esparcia-Alcázar, A.I., Eds.; ACM: New York, NY, USA, 2015; pp. 1–8.
14. Zalmout, N.; Ghanem, M. Multidimensional community detection in Twitter. In Proceedings of the 8th International Conference for Internet Technology and Secured Transactions (ICITST-2013), London, UK, 9–12 December 2013; pp. 83–88.
15. Camacho, D.; González-Pardo, A.; Ortigosa, A.; Gilpérez-López, I.; Urruela, C. RiskTrack: A New Approach for Risk Assessment on Radicalisation Based on Social Media Data. In Proceedings of the AfCAI 2016: Workshop on Affective Computing and Context Awareness in Ambient Intelligence, Murcia, Spain, 24–25 November 2016; Ezquerro, M.T.H., Nalepa, G.J., Mendez, J.T.P., Eds.; CEUR-WS: Aachen, Germany, 2016; Volume 1794.
16. Conde-Céspedes, P.; Marcotorchino, J.F.; Viennet, E. Comparison of linear modularization criteria using the relational formalism, an approach to easily identify resolution limit. In *Advances in Knowledge Discovery and Management*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 101–120.
17. Emmons, S.; Kobourov, S.; Gallant, M.; Börner, K. Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale. *PLoS ONE* **2016**, *11*, e0159161. [CrossRef] [PubMed]
18. Almeida, H.; Guedes, D.; Meira, W., Jr.; Zaki, M.J. Is There a Best Quality Metric for Graph Clusters? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; Lecture Notes in Computer Science; Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6911, pp. 44–59.
19. Newman, M.E.J.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **2004**, *69*, 026113. [CrossRef]
20. Fortunato, S.; Barthélemy, M. Resolution limit in community detection. *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 36–41. [CrossRef]
21. Clauset, A.; Newman, M.E.J.; Moore, C. Finding community structure in very large networks. *Phys. Rev. E* **2004**, *70*, 066111. [CrossRef] [PubMed]
22. Rosvall, M.; Axelsson, D.; Bergstrom, C.T. The map equation. *Eur. Phys. J. Spec. Top.* **2009**, *178*, 13–23. [CrossRef]
23. Fortunato, S. Community detection in graphs. *Phys. Rep.* **2010**, *486*, 75–174. [CrossRef]
24. Raghavan, U.N.; Albert, R.; Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **2007**, *76*, 036106. [CrossRef] [PubMed]
25. Reichardt, J.; Bornholdt, S. Statistical Mechanics of Community Detection. *Phys. Rev. E* **2006**, *74*, 016110. [CrossRef] [PubMed]
26. Lancichinetti, A.; Fortunato, S.; Radicchi, F. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **2008**, *78*, 046110. [CrossRef] [PubMed]
27. Kelly, J.P. *Meta-Heuristics: Theory and Applications*; Kluwer Academic Publishers: Norwell, MA, USA, 1996.
28. Feo, T.A.; Resende, M.G.C. A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.* **1989**, *8*, 67–71. [CrossRef]
29. Feo, T.A.; Resende, M.G.C.; Smith, S.H. A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Oper. Res.* **1994**, *42*, 860–878. [CrossRef]
30. Resende, M.G.C.; Ribeiro, C.C. GRASP: Greedy Randomized Adaptive Search Procedures. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*; Burke, E.K., Kendall, G., Eds.; Springer: Boston, MA, USA, 2014; pp. 287–312.
31. Sánchez-Oro, J.; López-Sánchez, A.D.; Colmenar, J.M. A general variable neighborhood search for solving the multi-objective open vehicle routing problem. *J. Heuristics* **2017**, 1–30. [CrossRef]
32. Martí, R.; Martínez-Gavara, A.; Sánchez-Oro, J.; Duarte, A. Tabu search for the dynamic Bipartite Drawing Problem. *Comput. Oper. Res.* **2018**, *91*, 1–12. [CrossRef]
33. Sánchez-Oro, J.; Mladenović, N.; Duarte, A. General Variable Neighborhood Search for computing graph separators. *Optim. Lett.* **2017**, *11*, 1069–1089. [CrossRef]
34. Glover, F.; Laguna, M. *Tabu Search*; Kluwer Academic Publishers: Norwell, MA, USA, 1997.
35. Hansen, P.; Mladenović, N.; Todosijević, R.; Hanafi, S. Variable neighborhood search: Basics and variants. *EURO J. Comput. Optim.* **2017**, *5*, 423–454. [CrossRef]

36. Yang, J.; Leskovec, J. Patterns of temporal variation in online media. In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, Hong Kong, China, 9–12 February 2011; pp. 177–186.
37. Rossi R.A.; Ahmed, N.K. An Interactive Data Repository with Visual Analytics. *SIGKDD Explor.* **2016**, *17*, 37–41.

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

# Chapter 9

# A fast variable neighborhood search approach for multi-objective community detection

The journal paper associated to this part is:

- Pérez-Peló, S., Sánchez-Oro, J., Gonzalez-Pardo, A., & Duarte, A. (2021). "A fast variable neighborhood search approach for multi-objective community detection". Applied Soft Computing, 112, 107838. (doi:10.1016/j.asoc.2021.107838).
- Status: **Published.**
- Impact Factor (JCR 2021): 8.263
- Subject Category: Computer Science, Interdisciplinary Applications. Ranking 11/111 (Q1)
- Subject Category: Computer Science, Artificial Intelligence. Ranking 24/139 (Q1)

Applied Soft Computing 112 (2021) 107838

Contents lists available at ScienceDirect

Applied Soft Computing

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

# A fast variable neighborhood search approach for multi-objective community detection

Sergio Pérez-Peló, Jesús Sánchez-Oro, Antonio Gonzalez-Pardo, Abraham Duarte *

*Dept. Computer Science, Universidad Rey Juan Carlos, C/Tulipán, S/N, Móstoles, Spain*

ARTICLE INFO

*Article history:*
Received 16 February 2021
Received in revised form 14 August 2021
Accepted 15 August 2021
Available online 27 August 2021

*Keywords:*
Community detection
Variable Neighborhood Search
Greedy Randomized Adaptive Search Procedure
Metaheuristic
Multi-objective

ABSTRACT

Community detection in social networks is becoming one of the key tasks in social network analysis, since it helps analyzing groups of users with similar interests. This task is also useful in different areas, such as biology (interactions of genes and proteins), psychology (diagnostic criteria), or criminology (fraud detection). This paper presents a metaheuristic approach based on Variable Neighborhood Search (VNS) which leverages the combination of quality and diversity of a constructive procedure inspired in Greedy Randomized Adaptative Search Procedure (GRASP) for detecting communities in social networks. In this work, the community detection problem is modeled as a bi-objective optimization problem, where the two objective functions to be optimized are the Negative Ratio Association (NRA) and Ratio Cut (RC), two objectives that have already been proven to be in conflict. To evaluate the quality of the obtained solutions, we use the Normalized Mutual Information (NMI) metric for the instances under evaluation whose optimal solution is known, and modularity for those in which the optimal solution is unknown. Furthermore, we use metrics widely used in multi-objective optimization community to evaluate solutions, such as coverage, $\epsilon$-indicator, hypervolume, and inverted generational distance. The obtained results outperform the state-of-the-art method for community detection over a set of real-life instances in both, quality and computing time.

© 2021 Elsevier B.V. All rights reserved.

## Code metadata

Permanent link to reproducible Capsule: https://doi.org/10.24433/CO.9934279.v1.

## 1. Introduction

In recent years, the growth and development of social networks has caused scientists from different areas of knowledge to be interested in the study of their structure and its implications. Users of social networks are increasing every day, which has made them a very common source of data. Analyzing how the users are related between them, or how the information that they are sharing is intertwined, we can potentially obtain additional information that can be useful for other interests. For example, we can estimate the potential impact of a marketing campaign, what the general opinion about a certain topic is, what the users think about a company, a person or a service, etc.

In addition, the growth of big data techniques and concepts such as smart cities have led to the need for real-time analysis of large amounts of information quickly and efficiently. However, most of the traditional techniques are not adapted to deal with vast amounts of data, becoming unsuitable for most of the current challenges in social network analysis [1]. An efficient and effective analysis of social networks can report a high amount of benefits, so it is interesting to have a set of powerful algorithms that allow us to perform that analysis.

Most of the networks represent complex models with a large amount of data and interactions. The analysis of social networks are able to evaluate properties such as small world [2] or scale free [3], easing the understanding of those real-world networks. There is a special property that attracts the gaze of the scientific community, which is the community structure [4]. This property is able to shed light over several problems of different research areas: from social science to biological science, among others. See [5] for a thorough survey on community detection.

This work proposes a Variable Neighborhood Search (VNS) algorithm [6] for solving the Community Detection Problem (CDP) from a multi-objective perspective, resulting in the Multi-Objective Community Detection Problem (MOCDP). Every traditional objective function considered for the single-objective CDP presents one or more drawbacks for modeling the community structure of a given solution. In order to deal with this problem, the model of CDP as a multi-objective optimization problem is

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
*E-mail addresses:* sergio.perez.pelo@urjc.es (S. Pérez-Peló), jesus.sanchezoro@urjc.es (J. Sánchez-Oro), antonio.gpardo@urjc.es (A. Gonzalez-Pardo), abraham.duarte@urjc.es (A. Duarte).

https://doi.org/10.1016/j.asoc.2021.107838
1568-4946/© 2021 Elsevier B.V. All rights reserved.

becoming more relevant for the scientific community. Notice that considering more than one objective at the same time, which may complement among them, allows us to model the community structure of a network more precisely.

Although VNS has been traditionally considered for single-objective optimization, recent works have adapted the original VNS framework for tackling multi-objective optimization problems, resulting in robust algorithms which are competitive with the best methods in the literature [7,8].

The main contributions of this work are the following:

- An adaptation of the well-known VNS metaheuristic is proposed for dealing with multi-objective optimization problems. It considers that a solution in the VNS framework is the complete set of non-dominated solutions.
- The initial set of non-dominated solutions for the VNS is generated using the constructive phase of Greedy Randomized Adaptive Search Procedure (GRASP). This feature allows VNS to start the search from a set of diverse and high-quality solutions.
- A new greedy function that can be efficiently computed is proposed for the constructive procedure and for the local search method, with the aim of guiding the search for promising regions of the search space without requiring large computing times.
- Two local search methods are presented: the first one, Combined Search Procedure, follows the traditional multi-objective local optimization methods. However, the Independent Search Procedure is a new proposal which tries to further improve the set of non-dominated solutions by independently improving each considered objective.
- The quality of the proposal is analyzed through multi-objective optimization perspective. As far as we know, previous works only consider the metrics related with social network analysis, but it is interesting to evaluate the set of non-dominated solutions under multi-objective optimization metrics which are specifically designed to that end.

The paper is structured as follows: Section 2 formally defines the considered problem, as well as the metrics proposed for the evaluation of solutions; Section 3 briefly reviews the most relevant papers related with our research; Section 4 presents the new Variable Neighborhood Search-based procedure proposed for detecting communities and exposes how the initial solution set is generated with a constructive procedure, as well as the neighborhood structures considered within Variable Neighborhood Search; Section 5 introduces the computational experiments performed to test the quality of the proposal; and finally Section 6 draws some conclusions on the research.

## 2. Problem definition

A social network, conformed with a set of users and a set of relations among them, can be modeled as a graph $G = (V, E)$. Users are represented by the set of nodes $V$, with $|V| = n$, while relations among users are represented by the set of edges $E$, with $|E| = m$. Notice that an edge $(u, v) \in E$ indicates that users $u, v \in V$ are related in the social network. The kind of relation between the users strictly depends on the purpose of the social network (friendship, work, etc.). This paper considers bidirectional relationships. Thus, if there is a relation $(u, v)$, then the relation $(v, u)$ is also contemplated (i.e., $G$ is an undirected graph).

The aim of this work is to deal with the Community Detection Problem (CDP) following a multi-objective approach. A community $C_i \subseteq V$ inside a network $G$ is defined as a set of users and the relations that connect those users. In other words, the community $C_i$ is represented by the induced subgraph $G_i = (C_i, E_i)$, where $E_i = \{(u, v) \in E : u, v \in C_i\}$. The CDP then consists in separating the complete social network into communities or groups. Although there is not a formal definition for community in the literature, the most widely accepted definition considers that a community is a group of users that are closely related to each other (i.e., share some properties/interests in the social network).

In terms of graphs, a well-detected community is the one whose nodes are densely connected among them and sparsely connected to nodes which do not belong to the community. Given a community $C_i$, the edges that connect nodes in the same community are usually known as intra-community edges, $\mathcal{E}_{\blacktriangleleft}(C_i)$, while those connecting nodes in different communities are named as inter-community edges, $\mathcal{E}_{\blacktriangleright}(C_i)$. In mathematical terms,

$$\mathcal{E}_{\blacktriangleleft}(C_i) = \{(u, v) \in E : u, v \in C_i\}$$
$$\mathcal{E}_{\blacktriangleright}(C_i) = \{(u, v) \in E : u \in C_i \wedge v \notin C_i\}$$

Following this definition, a community $C_i$ in a social network is well defined if it presents a large number of intra-community edges $\mathcal{E}_{\blacktriangleleft}(C_i)$ and, at the same time, a small number of inter-community ones $\mathcal{E}_{\blacktriangleright}(C_i)$.

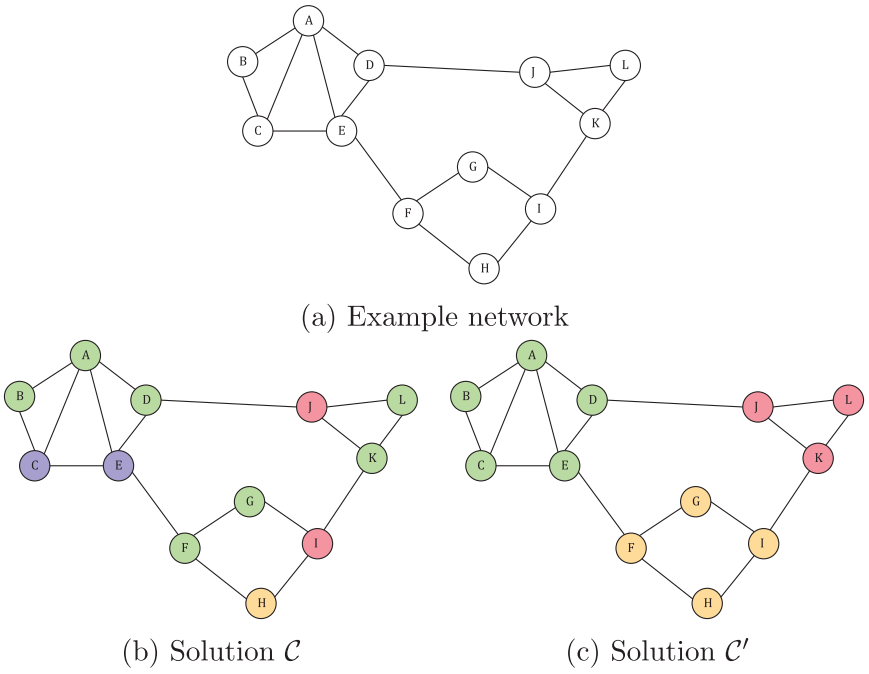
Given a social network, the CDP consists in assigning each user to a single community. Each community is labeled with an integer number $i$, with $1 \leq i \leq c \leq n$, being $c$ the number of communities detected. Depending on the problem under consideration, the number of communities may be fixed or not [9]. In the CDP variant tackled in this paper the number of communities is not fixed *a priori*.

A solution $\mathcal{C}$ for the CDP is modeled as the set of communities $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$ of the network. Then, a solution for the CDP is feasible when all the nodes have been assigned to a single community, i.e., $\sum_{i=1}^{c} |C_i| = n$ and $C_i \cap C_j = \emptyset$ for $1 \leq i, j \leq j$ with $i \neq j$.

Fig. 1(a) shows an example of a network with 12 nodes and 17 edges. Figs. 1(b) and 1(c) shows two feasible solutions $\mathcal{C}$ and $\mathcal{C}'$, respectively, for the CDP, where each node is colored with a different color that corresponds to its community (1-green, 2-red, 3-yellow, 4-blue). The first solution is represented as $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$, where $C_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{D}, \mathtt{F}, \mathtt{G}, \mathtt{K}, \mathtt{L}\}$, $C_2 = \{\mathtt{J}, \mathtt{I}\}$, $C_3 = \{\mathtt{H}\}$, $C_4 = \{\mathtt{C}, \mathtt{E}\}$. Similarly, the second solution is defined as $\mathcal{C}' = \{C'_1, C'_2, C'_3\}$, where $C'_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}\}$, $C'_2 = \{\mathtt{J}, \mathtt{K}, \mathtt{L}\}$, $C'_3 = \{\mathtt{F}, \mathtt{G}, \mathtt{H}, \mathtt{I}\}$.

Although solution depicted in Fig. 1(c) is clearly more visually appearing than the one presented in Fig. 1(b), there is not a common criterion to decide whether a solution presents a good community detection or not. There are several widely accepted metrics to evaluate the community structure of a solution. In particular, modularity [10] is one of the most extended metrics, and it has been used by several bioinspired algorithms [5,11] to find high quality solutions. However, it has some disadvantages. On the one hand, maximizing modularity is an $\mathcal{NP}$-hard problem [12]. On the other hand, a large value of modularity does not necessarily indicates that the communities detected are realistic since, in some cases, random networks without community structure can present large modularity values [13]. Last but not least, modularity has the well-known problem of resolution limitation [14]. This problem refers to the fact that the maximization of modularity is not able to reveal communities which are smaller than a certain scale, depending on the network size and on the degree of connections among real communities.

Most of the previous works are focused on the single-objective variant of the CDP (see for instance [15,16]). However, it may be interesting to consider more than one objective at the same time since it could lead us to find new and more reliable communities




(a) Example network

(b) Solution $\mathcal{C}$ (c) Solution $\mathcal{C}'$

**Fig. 1.** Example of two different solutions for the CDP over an example network.

in the networks. In a multi-objective optimization problem, two or more objectives are in conflict with each other, which means that improving one of the objectives usually leads to deteriorate the other objectives. Therefore, there is not a single solution with the optimal value in all the considered objectives. The main goal in multi-objective programming is to find a set of non-dominated solutions.

This work tackles the CDP from a multi-objective perspective, resulting in the Multi-Objective Community Detection Problem (MOCDP). In particular, two conflicting objectives are considered: Negative Ratio Association (NRA) and Ratio Cut (RC). The former measures the percentage of intra-community edges that exists with respect to the size of the community, while the latter evaluates the percentage of inter-community edges in a community with respect to its size. In mathematical terms, NRA and RC of a cluster $i$ are defined as:

$$NRA(C_i) = -\frac{\mathcal{E}_{\blacktriangleleft}(C_i)}{|C_i|} \qquad RC(C_i) = \frac{\mathcal{E}_{\blacktriangleright}(C_i)}{|C_i|}$$

Similarly, the NRA and RC of a complete solution $\mathcal{C}$ are defined as:

$$NRA(C) = \sum_{C_i \in C} NRA(C_i) \qquad RC(C) = \sum_{C_i \in C} RC(C_i)$$

being $c$ the number of communities in the corresponding solution.

Following these definitions, a solution with small NRA and RC values presents a good community structure. These two metrics were proven to be in conflict in [17]. Analyzing the objective functions independently, optimizing only NRA usually results in solutions with small communities which are densely connected, while focusing only in RC leads us to obtain solutions with large communities. Notice that, dealing with both metrics simultaneously, allows us to overcome the drawbacks of each metric when considered independently. The MOCDP is focused on minimizing both objectives, NRA and RC.

Let us illustrate how we can evaluate these two metrics by considering the example introduced in Fig. 1. Specifically, for solution $\mathcal{C}$, we have:

$$NRA(\mathcal{C}) = -\frac{4}{7} - \frac{0}{2} - \frac{0}{1} - \frac{1}{2} = -1.07$$

$$RC(\mathcal{C}) = \frac{11}{7} + \frac{6}{2} + \frac{2}{1} + \frac{5}{2} = 9.07$$

Similarly, for solution $\mathcal{C}'$:

$$NRA(\mathcal{C}') = -\frac{7}{5} - \frac{3}{3} - \frac{4}{4} = -3.4$$

$$RC(\mathcal{C}') = \frac{2}{5} + \frac{2}{3} + \frac{2}{4} = 1.57$$

As it can be derived from the equations, the best solution with respect to both NRA and RC is $\mathcal{C}'$, since it presents the minimum values in both objective functions. Therefore, we can also conclude that $\mathcal{C}'$ dominates $\mathcal{C}$.

## 3. Literature review

Community detection problems (CDP) have attracted the interest of the scientific community in the last years, mainly due to the relevance of the results derived from this research. It is possible to find relevant research works in the literature that describe how community detection can be applied in real-world environments, thus finding an interesting utility for this area of research. For example, some works use Community Detection techniques in the area of cybersecurity, where the goal is to reduce the threats over a certain cluster of actives using these techniques. As an example, in [18] authors propose a modularity-based adaptive algorithm applied to social-aware message forwarding strategy in MANETs (Mobile Ad Hoc Networks) and worm propagation containment in Online Social Networks. A different field in which these algorithms could be useful is Business Intelligence and Business Science, where certain topics can be modeled as a network. For instance, in [19], we can find an application in business science in terms of topological features and nodal attributes.

Other research field that focus the attention of the research community is politics. In the last years there has been a huge increase in the use of social networks for political purposes. In this domain, there are two main problems that can be solved: topic opinion and political polarization. The former refers to those works whose goal is to understand what users think about a specific topic. The latter contains the works that try to align SN users

with the different political parties. An example of topic opinion is [20] where authors tried to classify citizen's voting intention based on the tweets published during the Scottish Independence Referendum in 2014.

An example about political polarization, is the work published by Borge-Holthoefer et al. [21]. In this work, authors analyzed the social structure and the content of the tweets published by the users to understand the opinion evolution in Egypt during the summer of 2013. In this summer, there was a military takeover that resulted in an increase of polarized tweets but authors did not observe an ideological shift in the users.

In addition to the application domains, it is necessary to know which algorithms, or techniques, are the most popular for detecting the communities. In this sense, it is important to highlight that community detection it is a complex problem that it is difficult to solve by using classical algorithms. For this reason, it is really common to find researchers that use heuristics algorithms, and more precisely, bioinspired techniques [22].

In the literature, exact methods devoted to solve the community detection problem can be found, although they are not very efficient in solving the problem when the networks to be analyzed are too large. However, recent works have been focused on proposing new exact algorithms for dealing with large networks. Srinivas and Rajendran [23] propose a mathematical model for finding community structure on influential nodes, testing it in large scale networks. Although the performance is close to the state of the art, it is not able to reach a solution for networks with 115 nodes in a time limit of six hours, highlighting the relevance of using heuristic approaches. In the same way, Alinezhad et al. [24] propose a mathematical formulation for solving the community detection problem in attributed networks, considering both topological and node attributes. They limit the computing time to 7200 s for the exact procedure, since it is not able to converge in a reasonable computing time. The computational requirements of the exact procedures confirms the necessity of considering heuristic approaches for obtaining high quality solutions in small computing times.

In the area of bioinspired computation, evolutionary approaches are the most popular. It is important to highlight the review performed by Pizzuti in [25] about Evolutionary Computation (EC) techniques to detect communities in networks. An interesting work about EC is the work published by Said et al. [26], where authors designed a clustering coefficient-based genetic algorithm able to detect cohesive groups from dense graphs and also, communities in sparse networks. Other relevant work is [27] that presents a genetic algorithm that uses a multi-individual ensemble learning-based crossover function. The algorithm is improved with a local search strategy to speed up the convergence.

Other well-known bioinspired algorithms are the ones belonging to swarm intelligence. In this new group, the most popular algorithms are Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). These two algorithms are inspired by the social behavior of birds within a flock, and the behavior of ants seeking a path from the nest to the source of food, respectively. PSO has been successfully used for CDP in [28], where a discrete PSO algorithm is used to extract the communities in large-scale social networks by optimizing the modularity. Regarding ACO algorithm, this algorithm has been used to extract high-quality communities in Ego Networks [16].

All these works face the CDP from a single-objective perspective, usually considering modularity as optimization criterion (see [29] for a recent and complete survey on CDP). As far as we know, the most recent approach for solving the single objective CDP considering a metaheuristic framework is presented in [15]. In particular, the authors propose a Greedy Randomized Adaptive Search Procedure devoted to maximize the modularity of the communities detected.

Nevertheless, there are other works that try to solve the CDP by optimizing a multi-objective function (MOCDP). Multi-objective optimization has evolved in the last years with novel approaches for generating robust approximations of the Pareto front. For instance, [30] proposes a novel interactive preference-based multi-objective evolutionary algorithm for designing a bolt supporting network, while [31] presents a dynamic robust multi-objective optimization method for solving problems where the time is a key factor. The rationale behind following a multi-objective approach in CDP is that the optimization metrics, traditionally considered isolatedly, always have one or more handicaps, resulting in the loss of information related to the community structure. The MOCDP emerges as a possible solution for this problem, considering two or more metrics simultaneously for improving the community detection in a social network. The goal in this case is to find the different communities of a network by considering different conflicting objectives to be optimized [32]. Most of the works are focused on adapting well-known evolutionary algorithms such as NSGA-II to solve different multi-objective community detection problems. An evolutionary algorithm based on decomposition [33] is designed for maximizing the density of internal degrees while minimizing the density of external degrees. Another evolutionary algorithm for solving MOCDP is presented in [34], considering as objective functions the maximization of the intra-link strength of the communities and the minimization of the inter-link strength, which are very similar to those considered in [33]. Finally, another bioinspired algorithm, based on enhanced firefly methodology is presented in [35], which maximizes the in-degree of the nodes in each community while minimizing their out-degree. Notice that, although each previous work considers different objective functions, they are very similar among them, focusing on locating in the same community the most connected nodes. As far as we know, [17] presents the most recent multi-objective approach for solving the MOCDP, considering the NRA and RC metrics previously defined.

## 4. Algorithmic approach

Heuristic algorithms are designed for reaching a local optimum in short computing times. However, they usually stagnate in those local optima, reducing the portion of the search space explored. Metaheuristic algorithms emerge as a solution to overcome this situation by guiding the search of the heuristic method, thus reaching further regions of the search space [36].

This paper presents a metaheuristic algorithm based on the Variable Neighborhood Search (VNS) [6] framework. VNS methodology was originally designed to escape from local optima by performing systematic changes of neighborhood. It is worth mentioning that, as a metaheuristic approach, it cannot guarantee the optimality of the obtained solutions.

The effectiveness of VNS methodology has lead the scientific community to develop several variants, which can be classified according to the balance between diversification and intensification. On the one hand, Reduced VNS (RVNS) [37,38] is focused in diversification, considering stochastic changes of neighborhoods. On the other hand, Variable Neighborhood Descent (VND) [39,40] is devoted to intensification by performing deterministic changes of neighborhoods. Finally, Basic VNS (BVNS) [41] arises as a compromise between intensification and diversification by combining stochastic and deterministic changes of neighborhoods. As a result of the success of the methodology, several new variants have been proposed: General VNS (GVNS) [42], Variable Neighborhood Decomposition Search (VNDS) [43], Skewed VNS (SVNS) [44], or Variable Formulation Search (VFS) [45], among others.

VNS methodology was originally designed for tackling single objective optimization problems. Recently, the VNS framework has been also adapted for dealing with multiobjective problems [7]. Multi-objective VNS has lead to several recent successful research, emerging as one of the most robust methodologies in the area [8,46]. In this work, we adapt the multi-objective VNS presented in [7] for solving the multi-objective community detection problem, focusing in the BVNS variant (MOBVNS). Algorithm 1 presents the general framework of MOBVNS.

**Algorithm 1** *MOBVNS* $(S, k_{\max})$

1: $k \leftarrow 1$
2: **while** $k \leq k_{\max}$ **do**
3: $S' \leftarrow Shake(S, k)$
4: $S'' \leftarrow Improve(S')$
5: $k \leftarrow NeighborhoodChange(S, S'', k)$
6: **end while**
7: **return** $S$

The algorithm starts from an initial set of non-dominated solutions denoted with $S$. In the context of VNS, the initial front can be generated either at random, or using a more elaborated constructive procedure. In our case, this initial set is generated with the Greedy Randomized Adaptive Search Procedure (GRASP) described in Section 4.1. The second input parameter of the MOBVNS algorithm is the maximum neighborhood to be explored during the search, $k_{\max}$. As stated in previous works [47], the maximum neighborhood to be considered in the VNS algorithm is usually small, to avoid exploring completely different solutions in each iteration, which will eventually lead to a multi-start approach.

The algorithm starts by considering the neighborhood $k = 1$ (step 1). Then, MOBVNS iterates until reaching the maximum neighborhood $k_{\max}$ (steps 2–6). In each iteration, a perturbed set of solutions $S'$ is generated with the shake method presented in Section 4.2. Then, $S''$ is created as the set of non-dominated solutions derived from applying the local search method introduced in Section 4.3 to each solution contained in $S'$ to reach a local optimum of each perturbed solution. Finally, the neighborhood change procedure (Section 4.4) is responsible for selecting the next neighborhood to be explored.

The traditional neighborhood change method inside single-objective VNS restarts the search from the first neighborhood ($k = 1$) every time an improvement is found. Otherwise, the search continues in the next neighborhood ($k = k + 1$). In the context of multi-objective optimization, the definition of improvement is slightly modified. Specifically, neighborhood change method considers that an improvement is found if a solution has been able to enter in the set of non-dominated ones.

The algorithm ends when no improvement for the set of non-dominated solutions is found in any of the neighborhoods (i.e., the algorithm has not been able to insert a new solution in it), returning the resulting set of non-dominated solutions.

### 4.1. Generation of the initial set of non-dominated solutions

The main objective of a good constructive method in a multi-objective problem is to generate a front with high quality solutions (i.e, those that are non-dominated) while maintaining the diversity among them. With this aim, we propose a constructive procedure based on the Greedy Randomized Adaptive Search Procedure (GRASP). This metaheuristic is originally presented in [48] and formally defined in [49] which consists of two different phases: construction and local search. We refer the reader to [50] for a recent survey on this methodology and some extensions recently studied.

In this paper, we only consider the first stage (i.e., the construction phase) of the GRASP to populate an initial set of non-dominated solutions. Algorithm 2 shows the associated pseudocode. This procedure starts by creating one community for each node in the network (step 1) and initializing the set of non-dominated solutions $S$ with it (step 2). The next step corresponds to compute all the possible new communities that can be created by merging two of the existing ones, creating a Candidate List ($CL$) with them (step 3).

**Algorithm 2** *Construction* $(G = (V, E), \alpha)$.

1: $\mathcal{C} \leftarrow \{C_1, C_2, \ldots, C_n : C_i = \{v_i\}, v_i \in V \wedge \ 1 \leq i \leq n\}$
2: $S \leftarrow \{\mathcal{C}\}$
3: $CL \leftarrow \{\langle C_i, C_j\rangle, \forall C_i, C_j \in \mathcal{C}, 1 \leq i < j \leq n\}$
4: **while** $|CL| > 1$ **do**
5: $g_{\min} \leftarrow \min_{\langle C_i, C_j\rangle \in CL} g(C_i, C_j)$
6: $g_{\max} \leftarrow \max_{\langle C_i, C_j\rangle \in CL} g(C_i, C_j)$
7: $\mu \leftarrow g_{\max} - \alpha \cdot (g_{\max} - g_{\min})$
8: $RCL \leftarrow \{\langle C_i, C_j\rangle \in CL : g(\langle C_i, C_j\rangle) \geq \mu\}$
9: $\langle C_i, C_j\rangle \leftarrow Random(RCL)$
10: $\mathcal{C} \leftarrow \left(\mathcal{C} \setminus \{C_i, C_j\}\right) \cup (C_i \cup C_j)$
11: $CL \leftarrow \{\langle C_i, C_j\rangle, \forall C_i, C_j \in S, 1 \leq i < j \leq n\}$
12: $updateNDS(S, \mathcal{C})$
13: **end while**
14: **return** $S$

Next steps are repeated until the $CL$ contains a single candidate (i.e., there are only two communities in the solution). In each iteration, all the candidates are evaluated under a certain greedy criterion $g$ (steps 5 and 6). Then, these two values are used to define the threshold $\mu$ (Line 7) that depends on the parameter $\alpha \in [0, 1]$, which controls the randomness/greediness of the method. On the one hand, when $\alpha = 0$, $\mu = g_{\max}$ and only those communities with maximum value of the greedy function are included in the Restricted Candidate List ($RCL$). On the other hand, if $\alpha = 1$, $\mu = g_{\min}$, all the communities are included in the $RCL$ and then the algorithm becomes totally random. Therefore, the threshold defines the size of the $RCL$ because only the most promising candidates of $CL$ will belong to $RCL$ (step 8). Once the $RCL$ is constructed, an entry is randomly selected (step 9), specifying the two communities that will be merged (step 10).

Then, the algorithm updates the $CL$ (step 11), by removing all the pairs in which the two communities that have been merged, $C_i$ and $C_j$, were involved, and including a new candidate to merge the new community created $C_i \cup C_j$ with every other community in the solution (step 11). Finally, the resulting community is evaluated to be considered in the set of non-dominated solutions (step 12).

In order to define whether two given communities (i.e. $C_i$ and $C_j$) should be merged, the algorithm uses a greedy function denoted as $g(C_i, C_j)$. This function takes into account the number of edges that starts and ends in nodes belonging to $C_i$ and $C_j$, and the size of the resulting community (Eq. (1)).

$$g(C_i, C_j) = \frac{|\{(u, v) \in E \ \forall u, v \in C_i \cup C_j\}|}{|C_i \cup C_j|} \tag{1}$$

Although the construction phase is based on the greedy algorithm, the multi-objective optimization is used when the built solution has to be included in the reference front. Not all constructed solutions are finally included in the approximation of the pareto front but only those that are non-dominated solutions according to Negative Ratio Association (NRA) and Ratio Cut (RC). It is worth mentioning that a solution is non-dominated if it is

better than other solution already in the front in any of the two objectives. When a non-dominated solution is included in the front, all those solutions that are dominated by the new one are removed.

### 4.2. Perturbing solutions for the CDP

The *Shake* procedure is responsible for escaping from local optima within the VNS framework. In order to do so, the method resorts to a random solution in the neighborhood of the solution under exploration. In order to properly adapt this method to the context of multi-objective optimization, it is needed to previously define a basic movement of MOCDP, which consists in removing a vertex $v$ from its current community, say for instance $C_j$, and inserting it in a different community, for example $C_i$, with $i \neq j$. More formally, given a solution $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$, a vertex $v \in C_j \subset V$, and a community $C_i$, with $i \neq j$:

$$Move(\mathcal{C}, v, C_i) = \begin{cases} C_i \leftarrow C_i \cup \{v\} \\ C_j \leftarrow C_j \setminus \{v\} \end{cases}$$

The neighborhood of a solution is defined as the set of solutions that can be reached by performing the aforementioned move. Specifically,

$$N(\mathcal{C}) = \{\mathcal{C} \leftarrow Move(\mathcal{C}, v, C_i) : \forall v \in V \setminus C_i \wedge 1 \leq i \leq c\}$$

We rely on this definition to introduce the neighborhood $N_k(\mathcal{C})$. In particular, $N_k(\mathcal{C})$ is conformed with the set of solutions that can be obtained when performing exactly $k$ consecutive basic moves to $\mathcal{C}$.

Notice that solution $\mathcal{C}'$ obtained in the neighborhood $N_k$ of solution $\mathcal{C}$ is usually worse than $\mathcal{C}$ (in terms of objective functions value). However, the main objective of the *Shake* method is to escape from local optima, continuing the search from a completely different region of the search space. Additionally, all the solutions explored in the *Shake* methods are guaranteed to be feasible, being unnecessary to check the constraint of the problem, which is one of the most time consuming parts of the algorithm.

The output solution obtained in a *Shake* procedure is not necessarily a local optimum with respect to the defined neighborhood and, therefore, the local search method is applied to locally optimize the newly generated solution.

### 4.3. Local optimization

Population-based metaheuristics (i.e., Genetic Algorithms, Particle Swarm Optimization, Differential Evolution, etc.), are extended in the multi-objective context since they consider a set of solutions, which can be easily identified with the efficient set. Symmetrically, trajectory-based metaheuristics (VNS, Tabu Search, Simulated Annealing, etc.), use only one solution. It is possible to overcome this situation by considering the whole set of non-dominated solutions as the incumbent solution to a multi-objective problem [7].

We propose in this paper an improvement procedure (see step 4 in Algorithm 1) that receives the perturbed set of solutions, obtained with the *Shake* method, and returns a locally optimal set. Specifically, this method randomly scans each solution and then improves it with a local search algorithm. We consider two different strategies that follow the first improvement approach, which means that the search will restart when an improvement in the current solution under evaluation has been found.

Both strategies try to optimize NRA and RC of each solution, but they differ in the way that these two objectives are considered. The first strategy, named Independent Search Procedure (ISP), independently improves each objective starting from the very same solution (in the reference front). The second strategy is called Combined Search Procedure (CSP) since both metrics (NRA and RC) are alternatively considered in the procedure.

Algorithm 3 shows a general scheme of the improvement method that needs to be particularized for each strategy. Specifically, this method considers a generic objective function denoted with $f$. On the one hand, the first local search strategy takes into account $f(\mathcal{C}) = NRA(\mathcal{C})$ and then $f(\mathcal{C}) = RC(\mathcal{C})$. On the other hand, the second strategy alternatively takes in each iteration of the while-loop either $f(\mathcal{C}) = NRA(\mathcal{C})$ or $f(\mathcal{C}) = RC(\mathcal{C})$. After finishing any of the two local search strategies, non-dominated solutions found are tested to be admitted (or not) in the reference set.

**Algorithm 3** *Improve* ($\mathcal{C} = \{C_1, C_2, \ldots, C_c\}, S$)

```
1:  Improve ← True
2:  while Improve do
3:      Improve ← False
4:      for C_i ∈ C do
5:          for v ∈ V \ C_i do
6:              S' ← Move(C, v, C_i)
7:              updateNDS(S, C')
8:              if f(C') < f(C) then
9:                  C ← C'
10:                 Improve ← True
11:                 Restart the search at step 3
12:             end if
13:         end for
14:     end for
15: end while
```

Algorithm 3 receives as input parameter a solution. For the sake of brevity, we omit the inclusion of either NRA and RC as input parameter since they are generalized by the function $f$. The method iterates over all the communities and vertices (steps 4–14). In each iteration, we evaluate the impact of moving $v$ from its current community to a different one in the solution under evaluation. In order to do so, we use the defined $Move(\mathcal{C}, v, C_i)$, see step 6. The procedure tries to include the neighbor solution in the reference front by using the procedure *updateNDS* (step 7). It basically tests whether $\mathcal{C}'$ is dominated by other solution belonging to $S$. If so, the efficient front remains unaltered; otherwise, $\mathcal{C}'$ is included in $S$, removing those solutions in $S$ that become dominated.

After that, if an improvement is found with respect to the criteria under evaluation (either NRA and RC), the incumbent solution is updated and the search restarts again. The method ends when no improvement is found. It is worth mentioning that no return is needed since step 7 already includes all the non-dominated solutions found during the search.

The local optimization process in a multi-objective optimization problem is usually designed to simultaneously optimize all the considered objectives. In the context of MOCDP, we propose two different local optimization strategies: Combined Search Procedure (CSP) and Independent Search Procedure (ISP).

The first one, CSP, follows the traditional approach, where the two considered objectives are optimized at the same time. Specifically, each iteration of the local search method, presented in Algorithm 3, is focused on optimizing either NRA or RC. In particular, the even iterations finds a local optimum with respect to NRA, while the odd iterations focuses on finding a local optimum with respect to RC.

The second method, ISP, follows a different criterion with the aim of finding better solutions for each objective. In particular, each solution is optimized with each considered objective, i.e., NRA and RC, as in a single-objective optimization problem. In other words, given a certain solution, the ISP finds a local

optimum with respect to NRA and then with respect to RC, starting from the same initial solution. All the solutions found during the optimization process are evaluated to be included in the set of non-dominated solutions.

The termination criterion for both, ISP and CSP, is the same: the search stops when no new non-dominated solutions have been found after a complete execution of the local optimization method.

### 4.4. Neighborhood change

The main objective of the Neighborhood Change method within VNS is the selection of the next neighborhood to be explored. In the context of single objective optimization, the Neighborhood Change method usually receives three input parameters: the best solution found so far, the candidate solution to be evaluated, and the current neighborhood being explored ($k$). Then, it verifies whether the incumbent solution outperforms the best one. If so, the search is restarted from the first neighborhood, updating the best solution found so far. Otherwise, the search continues in the next neighborhood.

The Neighborhood Change method has been adapted to the multi-objective nature of the problem considered in this work. The most significant modification affects to the concept of improvement. In particular, we introduce the method *isNotDominated* in charge of comparing two non-dominated set of solutions. Algorithm 4 illustrates the pseudo-code of this procedure. It tests whether points in $S'$ are dominated, or not, by any point in $S$. If there is at least one non-dominated point in $S$, then the method returns `True`; otherwise (i.e., all points in $S$ are dominated), it returns `False`.

**Algorithm 4** *isNotDominated* ($S'$, $S$)

```
for all C ∈ S' do
    if (C ∉ S ∧ ¬Dominated(C, S')) then
        return True
    end if
end for
return False
```

The pseudo-code of the Neighborhood Change method is shown in Algorithm 5. The input parameters are now the current best non-dominated set $S'$, the reference front under evaluation $S$, and the neighborhood under exploration ($k$). As it was aforementioned, the most significant modification affects to step 1. The second relevant modification consists in updating the non-dominated set of solutions by merging $S'$ and $S$ with the procedure *updateNDS* (step 2). See Section 4.3 for further details. As it is customary in this method, if we do not find an improvement, the search continues in the next neighborhood (step 5).

**Algorithm 5** *NeighborhoodChange* ($E^\star$, $E$, $k$)

```
1: if isNotDominated(E*, E) then
2:     updateNDS(E*, E)
3:     k ← 1
4: else
5:     k ← k + 1
6: end if
7: return k
```

### 4.5. Computational complexity

In this section, the computational complexity of each component is analyzed and, then, the complete complexity of the proposed algorithm is computed. First of all, it is necessary to evaluate the cost of generating the initial non-dominated set of solutions with the constructive procedure. Analyzing the pseudocode presented in Algorithm 2, the complexity of constructing the candidate list CL is $O(n^2)$, since it requires to traverse the complete set of communities (initially one community per node) and, then, to create a candidate community to be merged with every other community. Notice that when the construction evolves, the number of available communities is reduced since it merges two communities in each iteration. Also, as this construction is included in a while loop until the CL contains a single community, the computational complexity is bounded $O(n^3)$.

We now compute the complexity of the local optimization method presented in Algorithm 3. In each iteration of the local search procedure, the method needs to traverse all the nodes, resulting in a complexity of $O(n)$. Each node is evaluated to enter in every other community, resulting in a complexity of $O(n)$ (in the worst case, there is a community for each node). A naive implementation of the move operator would result in a complexity of $O(m)$, since updating the inter and intra-community edges requires to evaluate every edge after performing the move. However, the proposed algorithm leverages the data structure called UnionFind, which basically assigns a representative node for each community in such a way that the evaluation of the modifications in inter and intra-community edges can be performed in $O(1)$. Therefore, the computational complexity of each iteration of local search procedure is bounded by $O(n^2)$ instead of $O(n^2 \cdot m)$ which would be obtained by the naive implementation. Since the local search is executed while an improvement is found, it is not possible to determine the complexity of the complete method since it highly depends on how close to a local optimum is the input solution.

The perturbation method presented in Section 4.2 again leverages the UnionFind structure to reduce complexity, which is bounded by $O(k)$, since $k$ moves are performed, being the complexity of each move $O(1)$.

Finally, the complexity of the complete VNS algorithm is computed. As it was aforementioned, the complexity of generating the initial front is bounded by the maximum between the constructive, $O(n^3)$, and the local search procedure, $O(n^2)$ in each iteration. This results in a complexity $O(n^3)$ for generating the initial front. Then, $k$ iterations are performed, where it is executed a shake procedure, with a complexity of $O(k)$, a local improvement, with a complexity of $O(n^2)$ in each iteration, and the neighborhood change method which presents a complexity of $O(1)$ since it only requires to select the next neighborhood to be explored. Therefore, the complexity of the complete algorithm is $O(k) \cdot \max\{O(n^3), O(n^2), O(1)\} = O(k \cdot n^3)$.

## 5. Experiments and results

In this section we will expose the experiments performed to test the effectiveness and efficiency of the proposed algorithm and to compare it with the best method found in the related literature [17]. All algorithms are executed over two different datasets: synthetic and real-world networks. For the former, we have used the network generator developed by Lancichinetti et al. [51] to construct synthetic instances,[1] where the node degree distribution and the community size follow a power-law highly configurable. The main advantage of these instances is

[1] Lancichinetti, Fortunato, and Radicchi (LFR) networks.

that the optimal ground truth for the community structure is known by construction. We have considered different configurations for the network generator and we have generated different network instances for each configuration (totalizing 52 different networks). In particular, we have considered networks with a range from 500 to 7500 nodes, and the edge probability $p$ is defined as $p \in [0.1, 0.8]$ with an interval of 0.1 for instances with 500, 1000 and 5000 nodes, and as $p \in [0.1, 0.3]$ with an interval of 0.1 for instances with 5500 to 7500 nodes.

To complement these instances, we additionally consider 12 real-world networks: Zacharys karate club [52] (32 nodes and 78 edges), dolphin social network [53] (64 nodes and 159 edges), American college football [54] (115 nodes and 613 edges), jazz [55] (with 198 nodes and 2742 edges), and netscience [56,57] (1589 nodes and 2742 edges), facebook large page-page network [58] (22 570 nodes and 171 002 edges), and the set of Twitch Social Networks [58] (with number of nodes in range from 1912 to 9498 and edges in range from 31 299 to 153 138). Notice that the ground truth for karate, dolphin, and football networks are known beforehand.

The computational experiment is divided into two different phases: on the one hand, we carry out a set of preliminary experiments to tune the parameters of our algorithms. In these experiments a subset of all instances will be used (18 out of 62), with the aim to avoid the overfitting of the algorithm. On the other hand, we run the final experiments over the whole benchmark to compare our best identified method with those presented in the state of the art. We additionally compare the proposed MOBVNS with the most extended algorithms for solving the CDP following a single-objective approach, with the aim of evaluating the relevance of modeling the CDP as a multi-objective optimization problem.

For sake of fairness, we have executed both algorithms with a time limit of 1800 s. Both algorithms have been executed in a computer with an AMD Ryzen 5 3600 AM4 core (3.6 GHz) with 16GB RAM. All algorithms were implemented using Java 9. With the aim of facilitating further comparisons, the dataset and the source code of the proposed algorithm are publicly available at http://grafo.etsii.urjc.es/mocdp.

### 5.1. Multi-objective metrics

In this paper, we deal with the variant of the Community Detection Problem, where the Negative Ratio Association (NRA) and Ratio Cut (RC) are optimized simultaneously. Notice that these two objectives have been already proven that are in conflict. Then, in order to compare the performance of the proposed algorithms we use metrics that evaluate the quality of an approximation of the Pareto front. Specifically, we have considered four of the most extended multi-objective metrics [59]: coverage, hypervolume, $\epsilon$-indicator, and inverted generational distance. Given two non-dominated set of solutions ($S$ and $S'$), the **coverage** metric, $CV(S, S')$, evaluates the number of solutions within the approximation front $S$ that are dominated by solutions in $S'$. In our experiments, we evaluate the quality of $S$ derived from an specific algorithm with respect to a reference set constructed with all non-dominated solutions found with all algorithms tested in the corresponding experiment. Given this definition, the smaller the value, the better. For the sake of brevity, we denote $CV(S, S')$ as $CV$, being $S$ the set of non-dominated solutions under evaluation and $S'$ the reference set (as indicated above).

The **hypervolume** metric, $HV$, measures the size of the space covered by the set of non-dominated solutions. In other words, it computes the hypervolume of the portion of the objective space that is weakly dominated by an approximation front. Then, large values of $HV$ implies that the set of non-dominated solutions obtained with the algorithm is better.

The $\epsilon$-**indicator**, $EPS(S, S')$, evaluates the smallest distance needed to transform every point of the approximation front under evaluation ($S$) in the closest point of the reference set $S'$ (equivalent to the coverage metric). Therefore, if we obtain low values of $\epsilon$-indicator, it indicates that the reference front generated by the algorithm under evaluation is better than others. As indicated above, we denote $EPS(S, S')$ as $EPS$.

Finally, the **inverted generational distance**, $IGD+(S, S')$, is an inversion of the well-known generational distance metric with the aim of measuring the distance from the incumbent set of non-dominated solutions ($S$) to the reference set obtained during the experiment ($S'$). Therefore, small values of $IGD+$ indicate a high proximity to the reference front, which is better. Finally, the computing time of all the algorithms is also presented, with the aim of evaluating the efficiency of the procedures. As it was aforementioned, we simplify the notation of $IGD+(S, S')$ as $IGD+$.

### 5.2. Context-based metrics

In social network analysis, there exists two popular performance metrics usually referred to as normalized mutual information (NMI) [60] and the modularity (Q) [10]. NMI requires for a ground truth since it evaluates the difference between the community structure detected for the incumbent algorithm and the true one. It is worth mentioning that the ground truth is known by construction for all LFR instances. Additionally, it is also available for karate, dolphin, and football instances.

The modularity can be evaluated in any network since it does not depend on the ground truth. This metric compares the structure of the communities against a random graph. More precisely, this metric measures how likely the communities are created at random. For this reason, modularity metric is particularly useful for real-world instances such as jazz or netscience, where the ground truth is unknown.

Notice that we are dealing with a multi-objective optimization problem. Therefore, instead of having a single solution, we have a set of non-dominated solutions. In order to provide a value of either NMI or Q, we follow the methodology proposed in [17]; i.e., to traverse the complete front finding the solution that presents the largest value in each metric. It implies that the solution that reaches the best Q value is not necessarily the one that provides the best result in terms of Normalized Mutual Information (NMI).

### 5.3. Preliminary experimentation

The first experiment is oriented to determine the best value of the $\alpha$ parameter (see Section 4.1). In particular, we test $\alpha = \{0.25, 0.50, 0.75, RND\}$, where $RND$ indicates that the value is selected randomly in the range [0, 1] for each construction. These values cover from an almost greedy constructive method to a semi-random one. For each instance used in this experiment, we execute the constructive algorithm for 100 independent iterations, returning the best solution found. Table 1 shows the associated results, where we report average values across the subset of preliminary instances for the $CV$, $HV$, $EPS$, and $IGD+$. We additionally include the average computing time required by the algorithms (column $T(s)$).

In view of these results, the best configuration of the constructive method is the one that uses $\alpha = 0.25$. In particular, based on the results obtained for the coverage metric, we can affirm that most of the points in the reference front also belong to the constructive method executed with a $\alpha = 0.25$. Also, the hypervolume metric ($HV$) is larger than the one attained with the other approaches, though it is closely followed by the constructive method configured with $RND$. The same occurs with

**Table 1**
Comparison of the reference front built by the constructive procedure with different values for the $\alpha$ parameter. Best results are highlighted with bold font.

| $\alpha$ | *CV* | *HV* | *EPS* | *IGD+* | *T* (s) |
|---|---|---|---|---|---|
| 0.25 | **0.38** | **0.73** | **0.05** | **0.01** | 119.04 |
| 0.50 | 0.81 | 0.68 | 0.09 | 0.04 | 123.04 |
| 0.75 | 0.81 | 0.59 | 0.18 | 0.13 | 170.30 |
| *RND* | 0.65 | 0.71 | 0.05 | 0.02 | **117.06** |

**Table 2**
Comparison of the reference front obtained with the local search procedures designed in this work. Best results are highlighted with bold font.

| Algorithm | *CV* | *HV* | *EPS* | *IGD+* | *T* (s) |
|---|---|---|---|---|---|
| *C*(0.25)+ISP | **0.14** | **0.70** | **0.03** | **0.00** | 117.99 |
| *C*(0.25)+CSP | 0.81 | 0.45 | 0.43 | 0.23 | **116.86** |

**Table 3**
Comparison of the reference front obtained with different values of $k$ for the MOBVNS method. Best results are highlighted with bold font.

| $k_{\max}$ | *CV* | *HV* | *EPS* | *IGD+* | *T* (s) |
|---|---|---|---|---|---|
| 0.1 | 0.71 | 0.71 | 0.07 | 0.03 | 118.76 |
| 0.2 | 0.67 | **0.72** | 0.06 | 0.02 | **118.43** |
| 0.3 | **0.66** | **0.72** | **0.05** | 0.02 | 119.14 |
| 0.4 | 0.67 | **0.72** | 0.07 | 0.02 | 122.75 |
| 0.5 | 0.68 | **0.72** | 0.06 | **0.01** | 123.80 |

the $\epsilon$-indicator, with $\alpha = 0.25$ the solutions provides the smallest value in the comparison, and the *RND* value is the second best approach. Regarding the inverted generational distance, $\alpha = 0.25$ again obtains the best results, closely followed by *RND*. Analyzing the computing time, we can clearly see that there are no differences among all considered variants, as expected.

Once we know what the best configuration for our constructive method is, we conduct an additional experiment to determine the performance of the proposed local search algorithms. From now one, we will refer to constructive algorithm with $\alpha = 0.25$ simply as $C(0.25)$. The results of the metrics obtained with both local search approaches (described in Section 4.3) are shown in Table 2.

As we can clearly see, the ISP performs better in this problem. Specifically, with respect to *IGD+*, coupling the ISP with the best version of our constructive method lead the algorithm to find an efficient set of solutions which is much closer to the reference front than CSP. Attending to hypervolume, we can see that ISP obtains a considerably larger value than the second variant. Analogously, the $\epsilon$-indicator is also the smallest one in the comparison, being heavily smaller than its competitor. Although the computational time required by CSP is smaller than ISP, there is not significant differences between these two procedures, and the great results obtained by ISP with the other metrics clearly justifies the choice of ISP.

In the next experiment, we test the best $k$ value for the *MOBVNS* algorithm. Table 3 shows the obtained results for each configuration of the algorithm.

Analyzing these results, we can see that all of them are quite similar, becoming difficult to choose the best value for $k_{\max}$ parameter. In particular, the hypervolume metric is not deterministic since almost all the variants report the same value. However, the coverage and $\epsilon$-indicator metrics suggest that the best value is $k_{\max} = 0.3$. Additionally, we can clearly see that the larger the value of $k_{\max}$, the more computationally demanding. Therefore, we select $k_{\max} = 0.3$ as the best value for the *MOBVNS* algorithm.

For the sake of brevity, we refer with MOBVNS as the multi-objective VNS variant that uses $C(0.25)$, ISP, and $k_{\max} = 0.3$.

**Table 4**
Comparison of the reference fronts obtained when applying the constructive method and when coupling it with the local search procedure ISP. Best results are highlighted with bold font.

| Algorithm | *CV* | *HV* | *EPS* | *IGD+* |
|---|---|---|---|---|
| *C*(0.25)+ISP | **0.03** | **0.67** | **0.00** | **0.00** |
| *C*(0.25) | 0.89 | 0.31 | 0.61 | 0.39 |

**Table 5**
Comparison of the reference front obtained with full MOBVNS framework and MOBVNS framework without Local Search Procedure. Best results are highlighted with bold font.

| Algorithm | *CV* | *HV* | *EPS* | *IGD+* |
|---|---|---|---|---|
| MOBVNS | **0.28** | **0.63** | **0.17** | **0.10** |
| MOBVNS (without ISP) | 0.29 | 0.48 | 0.42 | 0.11 |

### 5.4. Analysis of the effect of each component of the proposed algorithm

This section is devoted to clarify the contribution of each component of the algorithm in the final configuration. The algorithm is conformed with three main components: constructive procedure, local improvement method, and the combination of both of them in the VNS framework.

The first experiment is designed to evaluate the effect of the local search procedure over the results obtained by the constructive procedure isolatedly. Table 4 shows the results obtained in this experiment.

As it can be seen, the use of a local search procedure after constructing an initial non-dominated front with constructive method significantly improves the quality of the final non-dominated front obtained. In particular, the coverage of 0.03 obtained when coupling the constructive procedure with the local search method indicates that almost all the initial solutions are improved, while the value of 0.89 obtained by the constructive procedure indicates that almost all the initial solutions are dominated by the ones obtained with the local search procedure. The hypervolume, $\epsilon$-indicator, and inverted generational distance values supports these results.

The second experiment is intended to study the influence of the local search procedure within the VNS framework. Table 5 shows the obtained results in this comparison.

In this case, the coverage metric is not determinant since both algorithms present similar values, although considering the local search obtains better results, as well as when considering the inverted generational distance. However, analyzing the hypervolume and the $\epsilon$-indicator metrics, the relevance of the local search inside the VNS framework is confirmed, being the variant with local search twice better than the one without local improvement phase. Therefore, adding an improvement method result in more robust non-dominated sets of solutions.

Finally, having shown the relevance of the constructive and local search procedure, it is interesting to evaluate the effect of the initial front generated by GRASP algorithm in the final MOBVNS. In order to do so, the algorithm with GRASP for generating the initial front is compared with the same VNS algorithm but considering an initial random population of the reference front. Table 6 show the results obtained in this comparison.

The results clearly show that the contribution of starting from a good initial front to the final algorithm is justified. In particular, the coverage is reduced to 0.00 when considering a GRASP generation of the initial front, and the hypervolume of the random initial front is close to 0.00. Additionally, the $\epsilon$-indicator also confirms the superiority of the GRASP initialization. If we analyze the inverted generational distance, we can see that considering a

**Table 6**
Comparison of the reference front obtained by the final proposed algorithm when considering an initial population generated with GRASP and Random constructions. Best results are highlighted with bold font.

| Algorithm | *CV* | *HV* | *EPS* | *IGD+* |
|---|---|---|---|---|
| GRASP initial front | **0.00** | **0.32** | **0.58** | 0.39 |
| Random initial front | 0.11 | 0.01 | 0.99 | **0.32** |

**Table 7**
Comparison of the reference front obtained with the best configuration for MOBVNS and the LMOEA proposed by [17]. Best results are highlighted with bold font.

| Algorithm | *CV* | *HV* | *EPS* | *IGD+* | *T* (s) |
|---|---|---|---|---|---|
| MOBVNS | **0.07** | **0.14** | **0.86** | 0.22 | **214.64** |
| LMOEA | 0.36 | 0.02 | 0.27 | **0.21** | 1800.00 |

random initial front provides more diversity, resulting in slightly better results when considering this metric.

Therefore, the contribution of each component of the main algorithm (constructive procedure, local search method, and complete VNS framework) is confirmed.

### 5.5. Final experiments

Having made the necessary adjustment in the proposed algorithm and having chosen the best parameters for each configuration of the procedure, we proceed to make the comparison with the best previous method found in the literature, denoted as LMOEA [17]. To do this, we will execute both algorithms over the full set of 62 instances. We first compare the obtained results with those metrics employed in preliminary experiments.

The parameter setting of LMOEA considers a population of 100 individuals, 100 generations, a crossover probability of 0.9, a mutation probability of 0.1 and a neighborhood size of 40 (see [17] for further details). Considering that we are comparing heuristics algorithms, we include an additional termination criterion based on the maximum allowed computing time. Specifically, we fix 1800 s as the maximum time spent in a single instance. If this additional termination condition is met, we halt the corresponding algorithm, returning the best solution found during that time horizon.

Table 7 shows the results obtained for the considered multi-objective metrics *CV*, *HV*, *EPS*, and *IGD+* as well as the average computational time required for each algorithm under evaluation. Notice that the results of each metric is the average value obtained across the complete set of 62 instances. Due to the different sizes of the considered instances, all the metrics are normalized in the range [0, 1] in the comparison.

As we can see, the proposed MOBVNS provides the best results in all four measures but IGD+, where the results are rather similar. Analyzing the computational time, there is a significant difference between the performance of both algorithms. Whereas LMOEA spends the budget time of 1800 s for every instance under evaluation, MOBVNS needs, in average, only 214.64 s, obtaining higher quality solutions in considerably smaller computing times.

Once we have compared both methods using the classical multi-objective metrics, we further analyze the quality of their solution by considering the experimental framework described in [17]. In particular, we first graphically depict the NMI for each LFR instance.

In order to have more robust conclusions, both algorithms were executed for 20 independent executions, reporting the average results. Fig. 2 shows these results for $n = 500$ and $n = 1000$, where $p$ varies from 0.1 to 0.8 in steps of 0.1. As we can observe in this figure, MOBVNS presents high quality solutions for $p < 0.5$ with values of NMI close to 1.0. As expected, for larger values of $p$ the behavior gets worse since these networks are harder to be solved. LMOEA seems to be more stable in these instances, ranging the NMI values from 0.55 to 0.7. Indeed, LMOEA is able to outperform MOBVNS in $p = 0.8$ and $n = 1000$.

The value of $p$ in the LFR generator indicates the average ratio between inter-community edges and the total edges in the optimal community detection provided by construction. Therefore, a large value of $p$ results in communities with several edges to nodes in other communities when comparing it with the total number of edges of the considered community. This value leads to networks which do not accurately represent real-world networks since, in them, the number of edges to nodes in other communities is usually small.

For that reason, in the instances in which $p \geq 0.5$, the number of edges to other communities is considerably larger than the number of edges in the same community, resulting in networks where the community structure is not necessarily preserved. Since the MOBVNS is designed for detecting communities in networks that present community structure, the main ideas of its design are not useful for these special instances, presenting similar or even worse performance than LMOEA.

Notice that there exists a considerable performance difference between the results of LMOEA that we present in these figures and those reported in [17]. This discrepancy might come from the fact that we have considered an additional termination condition (i.e., 1800 s of CPU time), not allowing the algorithm to reach the maximum number of generations. Indeed, in our computer, LMOEA is able to evolve the population for less than 50 generations (on average) in 1800 s, which is half of number used in [17].

In the next experiment, we compare both algorithms over the set of real-world instances. As it was aforementioned, the results obtained in this benchmark must be separated into two groups. On the one hand, those where the ground truth is known and, on the other hand, those where the ground truth is unknown. Therefore, for karate, dolphin, and football, we report NMI and Q. Whereas for jazz and netscience, we only show the Modularity.

We report in Table 8 the average NMI of 20 independent executions (and the associated standard deviation) for both, MOBVNS and LMOEA, over each instance. We additionally include the CPU time (notice that both algorithms were executed in the same computer). To facilitate the comparison, we additionally include the NMI values published in [17], where the LMOEA is executed for 100 generations (without time limit). As we can see in this experiment, MOBVNS obtains competitive results in all networks by spending few seconds of computing time. It consistently finds better results than LMOEA (executed for 1800 s) and it is relatively close to LMOEA executed without time limit. The standard deviation smaller than 0.01 on average reached by MOBVNS confirms the robustness of the method, reaching the best values or close to best in most of the executions.

We now show in Table 9 the modularity obtained with MOBVNS and LMOEA when considering the whole set of real-world networks. As was aforementioned, the maximum CPU time is limited to 1800 s. As before, we also include the results of LMOEA reported in [17] where the authors did not consider any time limit. We report for each network the average Q value of 20 executions. Notice that, in the case of LMOEA, no solutions are generated after 1800 s for two instances, which is indicated with an asterisk in the corresponding cell of the table. The dashes in the LMOEA [17] indicates that these are new instances which were not tested in the original work and, therefore, there are not results for them. This experiment again confirms the good performance of the proposed algorithm. As can be observed, our method consistently produces better outcomes. The proposed

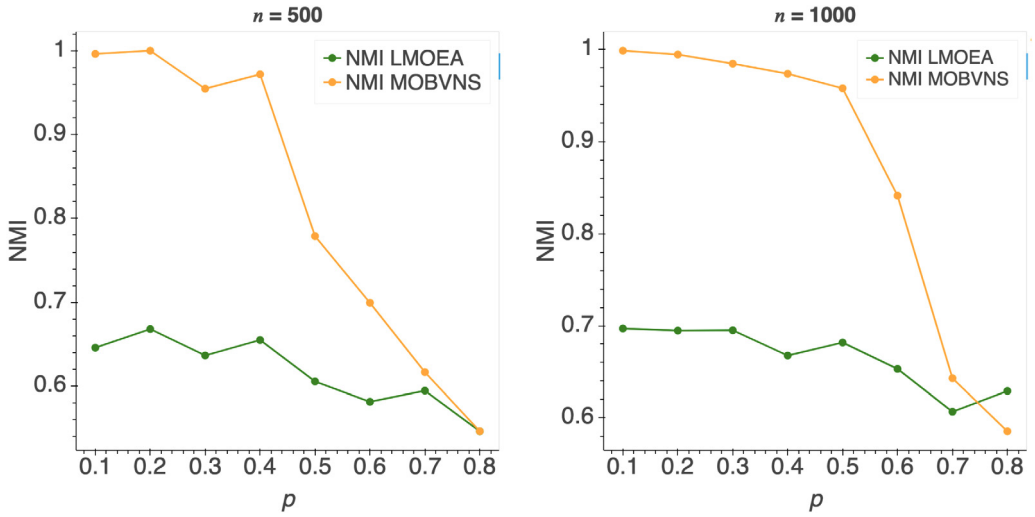



**Fig. 2.** NMI for $n = 500$ and $n = 1000$ instances.

**Table 8**
Summary of the results of NMI metric obtained by the proposed MOBVNS and LMOEA when solving the real world instances.

| Instance | LMOEA | | | | MOBVNS | | | |
|---|---|---|---|---|---|---|---|---|
| | Avg. NMI | Avg. Time (s) | Best NMI | Best Time (s) | Avg. NMI | Avg. Time (s) | Best NMI | Best Time (s) |
| dolphin | 0.050 | 1800.00 | 0.069 | 1800.00 | 0.751 | 0.41 | 0.770 | 0.11 |
| football | 0.020 | 1800.00 | 0.033 | 1800.00 | 0.864 | 1.80 | 0.877 | 0.27 |
| karate | 0.100 | 1800.00 | 0.100 | 1800.00 | 0.439 | 0.07 | 0.439 | 0.03 |

**Table 9**
Summary of the results of modularity metric obtained by the proposed MOBVNS and LMOEA when solving the real world instances.

| Instance | LMOEA | | | | MOBVNS | | | |
|---|---|---|---|---|---|---|---|---|
| | Avg. Q | Avg. Time (s) | Best Q | Best Time (s) | Avg. Q | Avg. Time (s) | Best Q | Best Time (s) |
| dolphin | 0.059 | 1800.00 | 0.067 | 1800.00 | 0.728 | 0.41 | 0.736 | 0.11 |
| football | 0.029 | 1800.00 | 0.067 | 1800.00 | 0.789 | 1.80 | 0.827 | 0.27 |
| karate | 0.061 | 1800.00 | 0.069 | 1800.00 | 0.500 | 0.07 | 0.508 | 0.03 |
| jazz | 0.027 | 1800.00 | 0.027 | 1800.00 | 0.899 | 11.99 | 0.899 | 10.87 |
| netscience | 0.058 | 1800.00 | 0.058 | 1800.00 | 0.972 | 1800.00 | 0.972 | 1800.00 |
| musae_DE | 0.001 | 1800.00 | 0.012 | 1800.00 | 0.032 | 1800.00 | 0.043 | 1800.00 |
| musae_ENGB | – | 1800.00 | – | 1800.00 | 0.095 | 1800.00 | 0.108 | 1800.00 |
| musae_ES | 0.001 | 1800.00 | 0.003 | 1800.00 | 0.067 | 1800.00 | 0.069 | 1800.00 |
| musae_FR | 0.001 | 1800.00 | 0.001 | 1800.00 | 0.030 | 1800.00 | 0.030 | 1800.00 |
| musae_RU | – | 1800.00 | – | 1800.00 | 0.201 | 1800.00 | 0.201 | 1800.00 |

**Table 10**
Summary of the results obtained by the proposed MOBVNS and LMOEA when solving the 62 instances considered in this work.

| Algorithm | #Best Q | #Best NMI | #Avg. Q | Avg. NMI |
|---|---|---|---|---|
| MOBVNS | **62** | **41** | **0.27** | **0.77** |
| LMOEA | 0 | 14 | 0.01 | 0.66 |

MOBVNS method reaches the best values of Modularity in the 5 considered networks. Indeed, only in netscience network, our method spends the whole budget of CPU time. Again, the standard deviation is smaller than 0.01, confirming the robustness of the proposal, which is able to reach the best value in most executions and stay close to it in those cases in which the best value is not found.

We summarize the results over the whole set of instances in Table 10. Specifically, we report in this table, the number of instances in which each algorithm shows the best results using the metrics just mentioned, as well as the average value of these two metrics.

As it can be seen in this table, taking into account Modularity the proposed algorithm (MOBVNS) provides the best solution in all the instances compared with LMOEA executed with a time limit of 1800 s. Regarding the NMI, MOBVNS obtains the best results in 41 out of 62 networks (66% of the instances). Analyzing the average modularity and average NMI, MOBVNS provides again better results as these mean values are higher.

In order to validate these results, we have conducted the well-known non-parametric Wilcoxon statistical test for pairwise comparisons, which answers the question: do the solutions generated by both algorithms represent two different populations? We consider a typical level of significance of 1%. The resulting value is smaller than 0.0005 when comparing MOVNS with LMOEA, confirming the superiority of the proposed algorithm. Therefore, MOVNS emerges as one of the most competitive algorithms for the MOCDP, being able to reach high quality solutions in reduced computing times.

Finally, we have performed an additional experiment to validate the results obtained by MOBVNS. In particular, we have included in the comparison the most extended algorithms for community detection in social networks, which are focused in

**Table 11**
Comparative of MOBVNS Algorithm against other classical heuristic methods found in literature.

| | EB | FG | LP | LE | ML | WT | IM | CL | MOBVNS |
|---|---|---|---|---|---|---|---|---|---|
| 1000_0.1_prev | 0.011 | 0.014 | 0.011 | 0.021 | 0.012 | 0.011 | 0.011 | 0.012 | **0.450** |
| 1000_0.1network | 0.433 | 0.428 | 0.433 | 0.273 | 0.433 | 0.433 | 0.433 | 0.433 | **0.454** |
| 1000_0.2_prev | 0.009 | 0.018 | 0.009 | 0.025 | 0.009 | 0.009 | 0.009 | 0.009 | **0.435** |
| 1000_0.2network | 0.385 | 0.367 | 0.385 | 0.221 | 0.386 | 0.385 | 0.385 | 0.386 | **0.428** |
| 1000_0.3_prev | 0.007 | 0.024 | 0.007 | 0.054 | 0.008 | 0.007 | 0.007 | 0.008 | **0.396** |
| 1000_0.3network | 0.339 | 0.319 | 0.339 | 0.220 | 0.340 | 0.339 | 0.339 | 0.340 | **0.414** |
| 1000_0.4_prev | 0.008 | 0.035 | 0.008 | 0.019 | 0.009 | 0.008 | 0.008 | 0.009 | **0.369** |
| 1000_0.4network | 0.285 | 0.277 | 0.290 | 0.186 | 0.293 | 0.290 | 0.290 | 0.293 | **0.407** |
| 1000_0.5_prev | 0.006 | 0.041 | 0.009 | 0.038 | 0.009 | 0.006 | 0.006 | 0.009 | **0.391** |
| 1000_0.5network | 0.176 | 0.230 | 0.241 | 0.141 | 0.246 | 0.240 | 0.241 | 0.246 | **0.411** |
| 1000_0.6_prev | 0.004 | 0.048 | 0.019 | 0.027 | 0.011 | 0.010 | 0.007 | 0.011 | **0.357** |
| 1000_0.6network | 0.086 | 0.203 | 0.197 | 0.154 | 0.204 | 0.188 | 0.193 | 0.204 | **0.386** |
| 1000_0.7_prev | 0.004 | 0.054 | 0.249 | 0.068 | 0.017 | 0.032 | 0.249 | 0.017 | **0.311** |
| 1000_0.7network | 0.040 | 0.169 | 0.249 | 0.144 | 0.150 | 0.126 | 0.249 | 0.150 | **0.317** |
| 1000_0.8_prev | 0.047 | 0.054 | 0.249 | 0.065 | 0.023 | 0.039 | 0.249 | 0.023 | **0.275** |
| 1000_0.8network | 0.057 | 0.158 | 0.249 | 0.142 | 0.125 | 0.089 | 0.249 | 0.125 | **0.272** |
| 500_0.1_prev | 0.023 | 0.023 | 0.023 | 0.020 | 0.023 | 0.023 | 0.023 | 0.023 | **0.446** |
| 500_0.1network | 0.419 | 0.414 | 0.419 | 0.325 | 0.419 | 0.419 | 0.419 | 0.419 | **0.424** |
| 500_0.2_prev | 0.020 | 0.026 | 0.020 | 0.039 | 0.020 | 0.020 | 0.020 | 0.020 | **0.411** |
| 500_0.2network | 0.376 | 0.367 | 0.376 | 0.235 | 0.376 | 0.376 | 0.376 | 0.376 | **0.432** |
| 500_0.3_prev | 0.017 | 0.028 | 0.020 | 0.061 | 0.017 | 0.017 | 0.017 | 0.017 | **0.378** |
| 500_0.3network | 0.330 | 0.323 | 0.329 | 0.219 | 0.330 | 0.330 | 0.330 | 0.330 | **0.382** |
| 500_0.4_prev | 0.015 | 0.032 | 0.015 | 0.053 | 0.016 | 0.015 | 0.015 | 0.016 | **0.390** |
| 500_0.4network | 0.264 | 0.265 | 0.279 | 0.206 | 0.280 | 0.279 | 0.279 | 0.280 | **0.362** |
| 500_0.5_prev | 0.013 | 0.041 | 0.025 | 0.027 | 0.017 | 0.015 | 0.015 | 0.017 | **0.383** |
| 500_0.5network | 0.162 | 0.222 | 0.239 | 0.168 | 0.239 | 0.232 | 0.233 | 0.239 | **0.374** |
| 500_0.6_prev | 0.008 | 0.054 | 0.249 | 0.036 | 0.016 | 0.015 | 0.015 | 0.016 | **0.365** |
| 500_0.6network | 0.044 | 0.180 | 0.248 | 0.135 | 0.187 | 0.174 | 0.248 | 0.187 | **0.342** |
| 500_0.7_prev | 0.056 | 0.051 | 0.249 | 0.068 | 0.022 | 0.019 | 0.249 | 0.022 | **0.317** |
| 500_0.7network | 0.066 | 0.175 | 0.249 | 0.133 | 0.145 | 0.127 | 0.249 | 0.145 | **0.329** |
| 500_0.8_prev | 0.004 | 0.047 | 0.249 | 0.043 | 0.026 | 0.049 | 0.249 | 0.026 | **0.323** |
| 500_0.8network | 0.066 | 0.151 | 0.249 | 0.116 | 0.124 | 0.151 | 0.249 | 0.124 | **0.310** |
| dolphins | 0.513 | 0.484 | 0.471 | 0.487 | 0.507 | 0.477 | 0.512 | 0.507 | **0.727** |
| football | 0.587 | 0.540 | 0.591 | 0.481 | 0.592 | 0.591 | 0.588 | 0.592 | **0.788** |
| karate | 0.318 | 0.286 | 0.307 | 0.306 | 0.342 | 0.289 | 0.307 | 0.342 | **0.500** |

single-objective optimization. Specifically, we have tested: Edge-Betweenness (EB) [61], Fast-Greedy (FG) [62], Label Propagation (LP) [63], Leading Eigenvector (LE) [57], MultiLevel (ML) [64], Walktrap (WT) [65], InfoMap (IM) [66], Cluster Louvain (CL) [64]. These algorithms are included in every community detection framework due to their popularity. This comparison allows us to evaluate the relevance of dealing with the CDP following a multi-objective approach. Table 11 shows the individual results obtained for each considered instance in terms of modularity.

It is worth mentioning that most of the algorithms included in the comparison are directly focused on optimizing modularity, although some of them such as the Label Propagation uses a different criterion as optimization metric. As it can be seen in the table, MOBVNS consistently obtains the best results in terms of modularity. Although in some instances, such as `1000_0.1network`, the improvement obtained is negligible, in most of the instances the multi-objective modeling of the problem allows the algorithm to reach considerably better modularity values. These results support the interest of tackling the community detection as a multi-objective optimization problem.

## 6. Conclusions and future work

In this paper, we have proposed a new metaheuristic method for community detection in social network based on Variable Neighborhood Search (VNS), where the set of initial set of non-dominated solutions is generated with a constructive procedure based on Greedy Randomized Adaptive Search Procedure methodologies (GRASP). The use of GRASP for the initial set of non-dominated solutions allows VNS to start the search from a promising region of the search space. The problem is addressed by optimizing the Radio Cut (RC) and the Negative Ratio

**Table A.12**
Information about the number of nodes, edges, and density of the instances derived from the LFR dataset.

| Instances | Nodes | Edges | Density |
|---|---|---|---|
| 500_0.1 | 500 | 10 674 | 0.08 |
| 500_0.1_prev | 500 | 10 386 | 0.08 |
| 500_0.2 | 500 | 9940 | 0.07 |
| 500_0.2_prev | 500 | 9444 | 0.07 |
| 500_0.3 | 500 | 9684 | 0.07 |
| 500_0.3_prev | 500 | 9870 | 0.07 |
| 500_0.4 | 500 | 10 338 | 0.08 |
| 500_0.4_prev | 500 | 10 326 | 0.08 |
| 500_0.5 | 500 | 10 310 | 0.08 |
| 500_0.5_prev | 500 | 9894 | 0.07 |
| 500_0.6 | 500 | 10 312 | 0.08 |
| 500_0.6_prev | 500 | 10 258 | 0.08 |
| 500_0.7 | 500 | 10 232 | 0.08 |
| 500_0.7_prev | 500 | 9442 | 0.07 |
| 500_0.8 | 500 | 10 194 | 0.08 |
| 500_0.8_prev | 500 | 9798 | 0.07 |
| 1000_0.1 | 1000 | 20 868 | 0.04 |
| 1000_0.1_prev | 1000 | 19 330 | 0.03 |
| 1000_0.2 | 1000 | 20 142 | 0.04 |
| 1000_0.2_prev | 1000 | 20 032 | 0.04 |
| 1000_0.3 | 1000 | 19 432 | 0.03 |
| 1000_0.3_prev | 1000 | 19 218 | 0.03 |
| 1000_0.4 | 1000 | 20 014 | 0.04 |
| 1000_0.4_prev | 1000 | 19 668 | 0.03 |
| 1000_0.5 | 1000 | 20 770 | 0.04 |
| 1000_0.5_prev | 1000 | 19 122 | 0.03 |
| 1000_0.6 | 1000 | 20 084 | 0.04 |
| 1000_0.6_prev | 1000 | 19 940 | 0.03 |
| 1000_0.7 | 1000 | 20 150 | 0.04 |
| 1000_0.7_prev | 1000 | 19 900 | 0.03 |
| 1000_0.8 | 1000 | 20 640 | 0.04 |
| 1000_0.8_prev | 1000 | 19 072 | 0.03 |

**Table A.13**
Information about the number of nodes, edges, and density of the instances derived from the LFR dataset.

| Instances | Nodes | Edges | Density |
|---|---|---|---|
| 5000_0.1 | 5000 | 100 430 | 0.01 |
| 5000_0.2 | 5000 | 1 011 070 | 0.08 |
| 5000_0.3 | 5000 | 103 662 | 0.01 |
| 5000_0.4 | 5000 | 101 732 | 0.01 |
| 5000_0.5 | 5000 | 99 770 | 0.01 |
| 5500_0.1 | 5500 | 112 858 | 0.01 |
| 5500_0.2 | 5500 | 111 868 | 0.01 |
| 5500_0.3 | 5500 | 111 592 | 0.01 |
| 6000_0.1 | 6000 | 121 486 | 0.01 |
| 6000_0.2 | 6000 | 121 692 | 0.01 |
| 6000_0.3 | 6000 | 119 742 | 0.01 |
| 6500_0.1 | 6500 | 134 384 | 0.01 |
| 6500_0.2 | 6500 | 132 336 | 0.01 |
| 6500_0.3 | 6500 | 133 520 | 0.01 |
| 7000_0.1 | 7000 | 142 156 | 0.01 |
| 7000_0.2 | 7000 | 141 552 | 0.01 |
| 7000_0.3 | 7000 | 142 424 | 0.01 |
| 7500_0.1 | 7500 | 151 060 | 0.01 |
| 7500_0.2 | 7500 | 152 778 | 0.01 |
| 7500_0.3 | 7500 | 152 972 | 0.01 |

**Table A.14**
Information about the number of nodes, edges, and density of the instances derived from the real-world instances dataset.

| Instances | Nodes | Edges | Density |
|---|---|---|---|
| dolphins | 62 | 159 | 0.08 |
| football | 115 | 613 | 0.09 |
| karate | 34 | 78 | 0.13 |
| netscience | 1589 | 2742 | 0.01 |
| jazz | 198 | 2742 | 0.14 |
| musae_DE_edgesnetwork | 9498 | 153 138 | 0.01 |
| musae_ENGB_edgesnetwork | 7126 | 35 324 | 0.01 |
| musae_ES_edgesnetwork | 4648 | 59 382 | 0.01 |
| musae_FR_edgesnetwork | 6549 | 112 666 | 0.01 |
| musae_RU_edgesnetwork | 4385 | 37 304 | 0.01 |

Association (NRA) metrics simultaneously as a bi-objective optimization problem. The quality of the solutions are evaluated by considering classic multi-objective metrics (Coverage, Hypervolume, $\epsilon$-indicator, and Inverted Generational Distance) and two well-known metrics in the context of social network analysis (Normalized Mutual Information and Modularity).

The performed experiments show that the combination of GRASP with VNS in a multiobjective optimization framework is able to produce high quality solutions for the Multi-Objective Community Detection Problem (MOCDP), outperforming the best method found in the literature, which is based on a multiobjective evolutionary algorithm (LMOEA, Local Multi-Objective Evolutionary Algorithm). Additionally, the efficient implementation of the proposed algorithm is almost ten times faster than the original LMOEA, becoming more suitable for large scale networks.

In future works, it would be interesting to analyze the performance of a multi-objective variant when compared with the best single-objective methods found in the state of the art, even including different conflicting objectives to be compared. Additionally, the VNS approach presented in this work will be tested in new variants of the Community Detection Problem, such as Dynamic Community Detection or Overlapping Community Detection, to validate the potential of this framework for dealing with Community Detection Problems.

## CRediT authorship contribution statement

**Sergio Pérez-Peló:** Conceptualization, Methodology, Software, Implementation, Writing. **Jesús Sánchez-Oro:** Conceptualization, Methodology, Software, Implementation, Writing. **Antonio Gonzalez-Pardo:** Conceptualization, Methodology, Software, Implementation, Writing. **Abraham Duarte:** Conceptualization, Methodology, Software, Implementation, Writing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This research was funded by "Ministerio de Ciencia, Innovación y Universidades, Spain" under grant ref. PGC2018-095322-B-C22, "Comunidad de Madrid" and "Fondos Estructurales" of European Union with grant refs. S2018/TCS-4566, Y2018/EMT-5062.

## Appendix. Instances description

See Tables A.12–A.14.

## References

[1] D. Camacho, Á. Panizo-LLedot, G. Bello-Orgaz, A. Gonzalez-Pardo, E. Cambria, The four dimensions of social network analysis: An overview of research methods, applications, and software tools, Inf. Fusion 63 (2020) 88–120.
[2] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004) 026113.
[3] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[4] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[5] Q. Cai, M. Lijia, M. Gong, A survey on network community detection based on evolutionary computation, Int. J. Bio-Inspired Comput. 8 (2014).
[6] P. Hansen, N. Mladenović, Variable neighborhood search, in: Search Methodologies, Springer, 2014, pp. 313–337.
[7] A. Duarte, J.J. Pantrigo, E.G. Pardo, N. Mladenovic, Multi-objective variable neighborhood search: an application to combinatorial optimization problems, J. Global Optim. 63 (3) (2015) 515–536.
[8] E. de Siqueira, M. Souza, S. de Souza, A multi-objective variable neighborhood search algorithm for solving the hybrid flow shop problem, Electron. Notes Discrete Math. 66 (2018) 87–94.
[9] L. Peel, D.B. Larremore, A. Clauset, The ground truth about metadata and community detection in networks, Sci. Adv. 3 (5) (2017) e1602548.
[10] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004) 026113.
[11] X. Li, C. Gao, A novel community detection algorithm based on clonal selection, J. Computational Information Systems 9 (2013) 1899–1906.
[12] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, D. Wagner, Maximizing modularity is hard, 2006, arXiv preprint Physics/0608255.
[13] J. Reichardt, S. Bornholdt, Statistical mechanics of community detection, Phys. Rev. E 74 (2006) 016110.
[14] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. 104 (1) (2007) 36–41.
[15] S. Pérez-Peló, J. Sánchez-Oro, R. Martín-Santamaría, A. Duarte, On the analysis of the influence of the evaluation metric in community detection over social networks, Electronics 8 (1) (2018).
[16] A. Gonzalez-Pardo, J.J. Jung, D. Camacho, ACO-based clustering for ego network analysis, Future Gener. Comput. Syst. 66 (2017) 160–170.
[17] F. Cheng, T. Cui, Y. Su, Y. Niu, X. Zhang, A local information based multi-objective evolutionary algorithm for community detection in complex networks, Appl. Soft Comput. 69 (2018) 357–367.
[18] N.P. Nguyen, T.N. Dinh, Y. Shen, M.T. Thai, Dynamic social community detection and its applications, PLoS One 9 (4) (2014) 1–18.

[19] V. Labatut, J.-M. Balasque, Detection and interpretation of communities in complex networks: Practical methods and application, in: Computational Social Networks: Tools, Perspectives and Applications, Springer London, London, 2012, pp. 81–113.
[20] A. Fang, I. Ounis, P. Habel, C. Macdonald, N. Limsopatham, Topic-centric classification of twitter user's political orientation, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2015, pp. 791–794.
[21] J. Borge-Holthoefer, W. Magdy, K. Darwish, I. Weber, Content and network dynamics behind egyptian political polarization on twitter, in: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, ACM, 2015, pp. 700–711.
[22] E. Osaba, J. Del Ser, D. Camacho, M. Nekane Bilbao, X.-S. Yang, Community detection in networks using bio-inspired optimization: Latest developments, new results and perspectives with a selection of recent meta-heuristics, Appl. Soft Comput. 87 (2020) 106010.
[23] S. Srinivas, C. Rajendran, Community detection and influential node identification in complex networks using mathematical programming, Expert Syst. Appl. 135 (2019) 296–312.
[24] E. Alinezhad, B. Teimourpour, M.M. Sepehri, M. Kargari, Community detection in attributed networks considering both structural and attribute similarities: two mathematical programming approaches, Neural Comput. Appl. 32 (8) (2020) 3203–3220.
[25] C. Pizzuti, Evolutionary computation for community detection in networks: A review, IEEE Trans. Evol. Comput. 22 (3) (2018) 464–483.
[26] A. Said, R.A. Abbasi, O. Maqbool, A. Daud, N.R. Aljohani, CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks, Appl. Soft Comput. 63 (2018) 59–70.
[27] M. Moradi, S. Parsa, An evolutionary method for community detection using a novel local search strategy, Physica A 523 (2019) 457–475.
[28] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan, L. Jiao, Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. 316 (2015) 503–516.
[29] Z. Yang, R. Algesheimer, C.J. Tessone, A comparative analysis of community detection algorithms on artificial networks, Sci. Rep. 6 (2016) 30750.
[30] Y.-N. Guo, X. Zhang, D.-W. Gong, Z. Zhang, J.-J. Yang, Novel interactive preference-based multiobjective evolutionary optimization for bolt supporting networks, IEEE Trans. Evol. Comput. 24 (4) (2019) 750–764.
[31] Y. Guo, H. Yang, M. Chen, J. Cheng, D. Gong, Ensemble prediction-based dynamic robust multi-objective optimization methods, Swarm Evol. Comput. 48 (2019) 156–171.
[32] P. Bedi, C. Sharma, Community detection in social networks, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 6 (3) (2016) 115–135.
[33] M. Gong, L. Ma, Q. Zhang, L. Jiao, Community detection in networks by using multiobjective evolutionary algorithm with decomposition, Physica A 391 (15) (2012) 4050–4060.
[34] C. Shi, Z. Yan, Y. Cai, B. Wu, Multi-objective community detection in complex networks, Appl. Soft Comput. 12 (2) (2012) 850–859.
[35] B. Amiri, L. Hossain, J.W. Crawford, R.T. Wigand, Community detection in complex networks: Multi–objective enhanced firefly algorithm, Knowl.-Based Syst. 46 (2013) 1–11.
[36] M. Gendreau, J.-Y. Potvin, Handbook of Metaheuristics, second ed., Springer Publishing Company, Incorporated, 2010.
[37] A. Herrán, M.J. Colmenar, A. Duarte, A variable neighborhood search approach for the vertex bisection problem, Inform. Sci. 476 (2019) 1–18.
[38] J. Sánchez-Oro, A.D. López-Sánchez, M.J. Colmenar, A general variable neighborhood search for solving the multi-objective open vehicle routing problem, J. Heuristics (2017) 1–30.
[39] A. Duarte, J. Sánchez-Oro, N. Mladenović, R. Todosijević, Variable neighborhood descent, Handb. Heuristics (2018) 341–367.
[40] J. Sánchez-Oro, A. Martínez-Gavara, M. Laguna, A. Duarte, R. Martí, Variable neighborhood descent for the incremental graph drawing, Electron. Notes Discrete Math. 58 (2017) 183–190, 4th International Conference on Variable Neighborhood Search.
[41] A. Duarte, L.F. Escudero, R. Martí, N. Mladenovic, J.J. Pantrigo, J. Sánchez-Oro, Variable neighborhood search for the vertex separation problem, Comput. Oper. Res. 39 (12) (2012) 3247–3255.
[42] J. Sánchez-Oro, A.D. López-Sánchez, J.M. Colmenar, A general variable neighborhood search for solving the multi-objective open vehicle routing problem, J. Heuristics (2017).
[43] P. Hansen, N. Mladenović, D. Perez-Britos, Variable neighborhood decomposition search, J. Heuristics 7 (4) (2001) 335–350.
[44] P. Hansen, N. Mladenović, Variable neighborhood search, in: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer US, Boston, MA, 2005, pp. 211–238.
[45] E.G. Pardo, N. Mladenović, J.J. Pantrigo, A. Duarte, Variable formulation search for the cutwidth minimization problem, Appl. Soft Comput. 13 (5) (2013) 2242–2252.
[46] X. Song, D. Jones, N. Asgari, T. Pigden, Multi-objective vehicle routing and loading with time window constraints: a real-life application, Ann. Oper. Res. (2019) 1–27.
[47] P. Hansen, N. Mladenović, J.A.M. Pérez, Variable neighbourhood search: methods and applications, Ann. Oper. Res. 175 (1) (2010) 367–407.
[48] T.A. Feo, M.G.C. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Oper. Res. Lett. 8 (2) (1989) 67–71.
[49] T.A. Feo, M.G.C. Resende, S.H. Smith, A greedy randomized adaptive search procedure for maximum independent set., Oper. Res. 42 (5) (1994) 860–878.
[50] M.G.C. Resende, C.C. Ribeiro, GRASP: Greedy randomized adaptive search procedures, in: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer US, Boston, MA, 2014, pp. 287–312.
[51] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (2008) 046110.
[52] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[53] S. Gregory, Finding overlapping communities in networks by label propagation, New J. Phys. 12 (10) (2010) 103018.
[54] M.E.J. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. 103 (23) (2006) 8577–8582.
[55] P.M. Gleiser, L. Danon, Community structure in jazz, Adv. Complex Syst. 6 (04) (2003) 565–573.
[56] R.A. Rossi, N.K. Ahmed, The network data repository with interactive graph analytics and visualization, in: AAAI, 2015.
[57] M.E. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (3) (2006) 036104.
[58] B. Rozemberczki, C. Allen, R. Sarkar, Multi-scale attributed node embedding, 2019.
[59] M. Li, X. Yao, Quality evaluation of solution sets in multiobjective optimisation: A survey, ACM Comput. Surv. 52 (2) (2019) 26.
[60] L. Danon, A. Díaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech. Theory Exp. 2005 (09) (2005) P09008.
[61] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, PNAS 99 (12) (2002) 7821–7826.
[62] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2004) 066111.
[63] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (2007) 036106.
[64] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) P10008.
[65] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. 105 (4) (2008) 1118–1123.
[66] M. Rosvall, D. Axelsson, C.T. Bergstrom, The map equation, Eur. Phys. J. Spec. Top. 178 (1) (2009) 13–23.

# Chapter 10

# Overlapping Community Detection: A hybrid approach with GRASP and Iterated Greedy

The journal paper associated to this part is:

- Pérez-Peló, S., Sánchez-Oro, J., Gonzalez-Pardo, A., & Duarte, A. (2021). "Overlapping Community Detection: A hybrid approach with GRASP and Iterated Greedy". Journal of Heuristics.
- Status: **Second Review.** View image 10.1
- Impact Factor (JCR 2020): 2.247
- Subject Category: Computer Science, Interdisciplinary Applications. Ranking 52/109 (Q2)

# Overlapping Community Detection: A hybrid approach with GRASP and Iterated Greedy

Sergio Pérez-Peló, Jesús Sánchez-Oro, Antonio Gonzalez-Pardo, Abraham Duarte

*Dept. Computer Science, Universidad Rey Juan Carlos, C/Tulipán, S/N,Móstoles, Spain*

**Abstract**

In recent years, obtaining information from different social networks, their users and their structure, has become a common practice among different research areas, such as marketing, data science, business analytics, biology, etc. This interest is mainly generated by the massive information that can be extracted from them. Several hard combinatorial optimization problems can be derived from social networks (user influence, topic popularity, polarization, etc.) and all of them belong to the well-known Social Network Analysis (SNA) research field. In this work, we focus on Community Detection Problem (CDP), that tries to group users in different clusters / communities based on their similarity, that can be denoted by different aspects such as their relations, preferences or membership of different groups. We will focus on a specific kind of CDP: Overlapping CDP, where one user could be part of more than one community simultaneously. The problem is tackled from a heuristic perspective, applying a Greedy Randomized Adaptive Search Procedure (GRASP) hybridized with an Iterated Greedy technique to obtain communities in different networks. The performance of this algorithm is compared against the best previous method found in the literature, which consists of a Density Peaks algorithm. To evaluate the performance of both algorithms, we have employed real-world and

*Email address:* {sergio.perez.pelo, jesus.sanchezoro, antonio.gpardo, abraham.duarte}@urjc.es (Sergio Pérez-Peló, Jesús Sánchez-Oro, Antonio Gonzalez-Pardo, Abraham Duarte)

synthetic instances that allow us to test their efficiency in different scenarios. Experimental phase reveals that the proposed algorithm outperforms the state of the art in terms of overlapped modularity, an evaluation metric that has been adapted from modularity metric to overlapping scenario.
*Keywords:* GRASP, Iterated Greedy, Overlapping Community Detection, Heuristics, Optimization

## 1. Introduction

Nowadays, social networks (SN) have become one of the most used kind of applications in our society. Its capability to connect people around all the globe is one of the main reasons why they are so extended in a broad spectrum of the population. This characteristic makes social networks one of the largest sources of data that can be analyzed, allowing us to extract relevant information about virtually all groups of the world's population (different ages, genders, ethnicity, cultures, etc.). These characteristics have attracted the attention of researchers in different fields, such as sociology, business, or computer science.

However, one of the most interesting aspects to analyze is the relationship that is established among different groups of users, and how these relationships inherently form a community. In some social networks, the assignment of an user to a community is not obvious. For example, Twitter users can not be directly assigned to a certain community without analyzing their friends/followers network. In others Social Networks these relationships are evident, such as on Facebook, where users can explicitly join different groups depending on their interests.

In a real-life scenario, users usually belong to more than one group at a time. For example, someone that is interested in jazz music, can be also interested in football or other topics not necessarily related with the first one. These groups are commonly known as communities, and the task of determining the different communities in a social network is referred to as Community Detection Problem (CDP) that belongs to the Social Network Analysis (SNA) research field.

Usually, research works focusing on CDP identify different communities by trying to ensure that each user belongs to only one community at the same time. In this work, we tackle the problem from the perspective in which a user can be grouped in more than one community, which is usually referred to as Overlapping Community Detection Problem (OCDP).

In order to solve this problem, our proposal consists of a hybrid approach that applies Greedy Randomized Adaptive Search Procedure (GRASP) [11] and Iterated Greedy (IG) [43], two well-known metaheuristics that has been successfully applied for solving a wide variety of $\mathcal{NP}$-hard problems.

The rest of the paper is organized as follows: Section 2 formally states the problem tackled in this work and the metric used to evaluate solutions. In Section 3, we summarize the literature review performed as knowledge base for this paper. In Section 4, we present a hybrid approach that uses GRASP and IG metaheuristics to reach high-quality solutions to the studied problem. Section 5 describes the experiments performed in order to evaluate the quality of our proposal, showing a high performance that overcome the state of the art. Finally, Section 6 sets out the conclusions drawn from the work carried out in this research.

## 2. Problem Description

A social network can be modeled as a graph $G = (V, E)$ where the set of vertices $V$, with $|V| = n$, corresponds to its users, while the set of edges $E$, with $|E| = m$, is conformed with 2-tuples $(u, v) \in E$, with $u, v \in V$, representing that there exists a relation between users $u$ and $v$. The specific relationship type is defined by the context of the social network itself. For example, it can indicate a friendship relationship (Facebook), a co-workers relationship (LinkedIn), or a follower-followed connection (Twitter). It is important to remark that these relations can be bidirectional or one-way connections, depending on the basal characteristics of the social network itself. In this work, we only consider bidirectional relations, i.e., if there is a connection between user `A` and user `B` (node

`A` and node `B` in the graph model), then there is also a connection between `B` and `A`.

In this paper, we try to solve the Overlapping Community Detection Problem (OCDP) which is a variant of the classical Community Detection Problem (CDP). The main characteristic of the OCDP is that each user may belong to more than one community at the same time. This version of the problem is more realistic because, for example, there might be certain users that are related to a scientific community and to a musicians community at the same time.

At this point, it is important to define the concept of community, which is a subset of users and the relations among them. More formally, a community $C_i$ is defined as the subgraph $C_i = (V_i, E_i)$, where $V_i \subseteq V$ contains all the users belonging to the community and $E_i$ is conformed with all the relations between users in $V_i$, i.e., $E_i = \{(u, v) \in E : u, v \in V_i\}$.

The main objective of CDP and, in particular, OCDP, is to provide an assignment of users to communities that fulfills the community structure found along the network under evaluation. Translated to graph terminology, the main objective is to find subgraphs highly connected, i.e., nodes of the same subgraph are densely connected among them, but sparsely connected to the other subgraphs. In other words, we try to maximize the number of edges between nodes belonging to the same subgraph while minimizing the number of edges connecting nodes in different subgraphs.

A solution $S$ for the OCDP is composed by a set of $k$ communities (i.e., $S = \{C_1, C_2, ..., C_k\}$) where $k$ (with $1 \leq k \leq n$) indicates the number of detected communities. Thus, $k = 1$ means that there is only one community that contains all nodes, while $k = n$ represents the case where there are $n$ communities with a node in each one. It is worth mentioning that the value of $k$ is not known *a priori*, and it must be determined by the algorithm.

The overlapping nature of the OCDP allows an user to belong to more than one community, so one node can appear more than once in a different $C_i \in S$. Then, a solution for the OCDP is feasible if and only if all nodes in the network have been associated to, at least, one community.

Once we know when a solution is feasible for the OCDP, we define the metric used to evaluate and compare different solutions. In Social Network Analysis, there are several metrics that have been accepted and applied to evaluate the quality of a solution. Perhaps, the most extended one is modularity [34], that has been successfully adapted to different approaches and applications of the context of CDP [18, 21, 26, 38, 48]. In this paper, we use the adaptation of the traditional modularity for the overlapping scenario described in [27]. It takes into account the possibility for a node to belong to several communities. The corresponding equation basically modifies the classical definition by adding a factor that represents the number of different communities to which a node belongs to. The evaluation is performed by independently calculating the modularity of each community and, then, averaging the results for all communities. Mathematically,

$$M_O(S) = \frac{\sum_{C_i \in S} M_O(C_i)}{|S|} \tag{1}$$

where $|S|$ is the number of detected communities in the incumbent solution. Given a community $C_i = (V_i, E_i)$ its overlapping modularity $M_O(C_i)$ is evaluated as:

$$M_O(C_i) = \frac{1}{|V_i|} \sum_{u \in V_i} \frac{|E_{\leftarrow}(u, C_i)| - |E_{\rightarrow}(u, C_i)|}{d_u \cdot s_u} \cdot \frac{|E_i|}{\frac{|V_i| \cdot (|V_i|-1)}{2}} \tag{2}$$

where $d_u$ indicates the degree of node $u$, and $s_u$ represents the total number of communities that contains node $u$. Terms $E_{\leftarrow}(u, C_i)$ and $E_{\rightarrow}(u, C_i)$ represent the set of intra-community and inter-community edges, respectively. More formally:

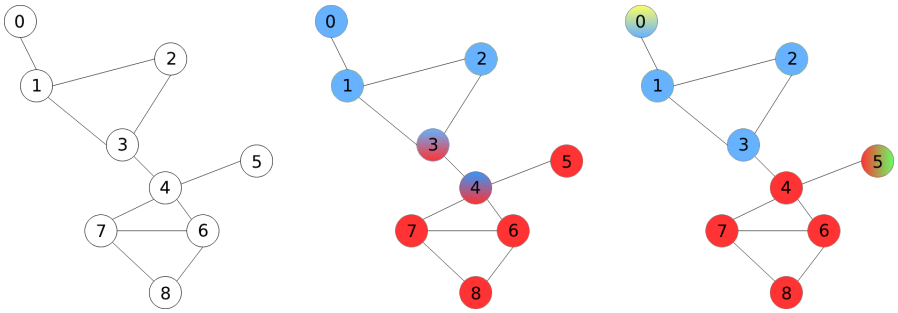
$$E_{\leftarrow}(u, C_i) = |\{v \in (V_i \setminus \{u\}) : (u, v) \in E\}|$$

$$E_{\rightarrow}(u, C_i) = |\{w \notin V_i : (u, w) \in E\}|$$

Finally, in order to normalize the result, since the size of each community could be rather different, this difference between intra and inter-community edges is multiplied by the ratio between the number of edges that actually exists in the community ($|E_i|$), and the number of edges that an ideal community would have ($\frac{|V_i| \cdot (|V_i|-1)}{2}$).

Let us illustrate with an example how we evaluate the aforementioned equations. Figure 1(a) shows a network with 9 nodes and 11 edges. Figure 1(b) depicts a solution $S$ conformed with two communities, $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$. These two communities, highlighted in blue and red, respectively, are then defined as $V_1 = \{0, 1, 2, 3, 4\}$, $E_1 = \{(0, 1), (1, 2), (1, 3), (2, 3), (3, 4)\}$; and $V_2 = \{3, 4, 5, 6, 7, 8\}$, $E_2 = \{(3, 4), (4, 5), (4, 6), (4, 7), (6, 7), (7, 8)\}$. Similarly, we show in Figure 1(c) the solution $S'$, composed of four communities, $C'_1 = (V'_1, E'_1)$, $C'_2 = (V'_2, E'_2)$, $C'_3 = (V'_3, E'_3)$, and $C'_4 = (V'_4, E'_4)$. These four communities, highlighted in yellow, blue, red and green, respectively, are then defined as $V'_1 = \{0\}$, $E'_1 = \emptyset$; $V'_2 = \{0, 1, 2, 3\}$, $E'_2 = \{(0, 1), (1, 2), (1, 3), (2, 3)\}$; $V'_3 = \{4, 5, 6, 7, 8\}$, $E'_3 = \{(4, 5), (4, 6), (4, 7), (6, 7), (7, 8)\}$; and $V'_4 = \{5\}$, $E'_4 = \emptyset$. Notice that, in $S$, nodes 3 and 4 belongs to communities $C_1$ and $C_2$ simultaneously, while in $S'$, node 0 belong to communities $C'_1$ and $C'_2$, and node 5 to $C'_3$ and $C'_4$ at the same time, emerging the overlapping nature of the OCDP.

The next step is to evaluate and compare the different solutions using the modularity metric previously described. On the one hand, following Equation 2, $M_O(C_1) = 0.335$, and $M_O(C_2) = 0.347$, resulting in a total modularity (Equation 1) of $M_O(S) = 0.336$. If we now evaluate the solution $S'$, the modularity of each community is evaluated as $M_O(C'_1) = 0$, $M_O(C'_2) = 0.472$, $M_O(C'_3) = 0.480$, and $M_O(C'_4) = 0$, resulting in $M_O(S') = 0.238$. Notice that the modularity of $C'_1$ and $C'_4$ is equal to zero since, following the suggestions in [54], those communities with a single node have a modularity equal to zero. Analyzing these results, we can conclude that $S$ is better than $S'$ in terms of




(a) Example network (b) Solution $\mathcal{S}$ (c) Solution $\mathcal{S}'$

Figure 1: Example network with 9 nodes and 11 edges and two different feasible solutions for the OCDP. Figure 1(a) shows a network with 9 nodes and 11 edges. Figure 1(b) depicts a solution $S$ conformed with two communities and two overlapping nodes. Figure 1(c) exposes a different solution $S$ consisting of four communities and two overlapping nodes.

modularity.

Having defined the objective function of the OCDP, the goal then is to find a solution $S^\star$ with the maximum $M_O(S^\star)$ value. In mathematical terms,

$$S^\star \leftarrow \arg\max_{S \in \mathbb{S}} M_O(S) \tag{3}$$

where $\mathbb{S}$ is the set of all possible community assignments that can be performed.

## 3. State-of-the-art

Nowadays, the relevance of social networks in society is undeniable. This is produced by several reasons: firstly, it is because of the high number of users connecting every day to social networks. Secondly, it is because of the services that these networks provide to the users. As a result, there is a huge amount of data generated by the users' actions that can be analyzed to extract some valuable knowledge.

The research area in charge of extracting this knowledge from social network is called *Social Network Analysis (SNA)* [3]. Its popularity has made that researchers from different research fields (such as Data Mining [1], Big Data [52], Machine Learning [4], or Complex Systems [36]) have applied SNA methods to

different application domains, like healthcare [18, 23, 45], marketing [19, 20, 48], cybersecurity [6, 12, 26], or politics [7, 21, 32]. These research fields that contribute to the SNA research is because of the wide variety of information that can be extracted from the networks. For example, it is possible to identify the most relevant users in the SN [29], measure any personality-based metric [26], analyze the speech used by the users [32], study how the misinformation is spread [49], or detect the groups of users with similar characteristics [14, 44].

Our revision of the relevant literature is focused on the extraction of the different groups, or communities, of users. This task is commonly known as *Community Detection Problems* (CDP) [22, 9, 47] and its main goal is to group users in different communities, or clusters, in such a way that users belonging to the same community are similar with each other, whereas they differ from users of other communities. To determine what kind of information is used to measure the similarity between two users, it is possible to use information provided in the users' profile, but the most common approach is to use the relationship in the network [14].

As it has been described in the previous section, there are two different types of CDP and differ in the number of communities to which an user can belong. The most extended method consist in determining that each user belongs only to one specific community at the same time, which is called *non-overlapping* CPD, or disjoint community detection. Classical algorithms for CDP were initially designed using the *non-overlapping* approach, some examples are Girvan-Newman [13] or the greedy approach proposed by Clauset et al. [5]. Nevertheless, the main problem is the high computational costs of solving CDP when the number of users increases. Thus, it is really common to use other approaches based on an iterated greedy algorithm [44], or bio-inspired algorithms [14], which provides good solutions in a reasonable amount of time.

The second type of CDP is called *overlapping* CDP, and it is the problem faced in this work. *Overlapping* CDP [53] describes problems in which users have to be grouped in different communities, but each user may belong to more than one community at the same time. This is done by including a belonging

factor or a soft membership vector in each node [15].

There are several approaches that allow the detection of overlapping communities. The most commonly known algorithms are CONGA and CONGO. CONGA stands from *Cluster-Overlap Newman Girvan Algorithm*, and it was proposed in [15]. This algorithm is based on the traditional Girvan-Newman algorithm [13] but CONGA incorporates a different vertex splitting procedure. The main problem with CONGA is that it has the same computational complexity that Girvan-Newman, which is $\mathcal{O}(m^3)$ (being $m$ the number of edges of the graph). In order to solve this problem, the same author proposed in 2008 the algorithm *CONGA Optimized*, known as CONGO [16], whose complexity is reduced to $\mathcal{O}(n \cdot \log n)$ where $n$ represents the number of nodes. Some examples of classical algorithms that have been adapted to the overlapping version are the *Clique-Percolation-Method* (CPM) [24] or Label Propagation [17]. There are other approaches based on greedy methods [51] or non-negative matrix factorization [55].

All these algorithms applied to Overlapping CDP suffer from the same drawbacks as the non-overlapping algorithms: the computational cost required to evaluate solutions. In order to alleviate this problem, it is quite common the usage of heuristics or metaheuristics approaches. In this paper, the Greedy Randomized Adaptive Search Procedure (GRASP) has been selected to solve the Overlapping CDP. One of the reasons about this decision is because this algorithm has been widely applied to solve a high range of application domains, providing good solutions in a reasonable amount of time [21, 39, 46]. One of these application domains is the Overlapping CDP and one of the latest research papers published that uses GRASP for this purpose is the one authored by Xu et al. [54].

In the just mentioned paper, the authors proposed an extended adaptive version of the density peak algorithm [41] for overlapping CDP, named EADP. This algorithm is in charge of detecting the overlapping communities, taking into account the distance between any pair of nodes. Given any two nodes, the distance between these nodes will depend on the number of nodes connected

to them. Once this distance is computed to any pair of nodes, the algorithm selects the centre for the different communities and allocates to those centres the nodes of the network according to the density peak algorithm. Although the algorithm provides promising results, its complexity is $\mathcal{O}(\bar{v} \cdot n^2)$, where $n$ is the number of nodes in the graph, and $\bar{v}$ is the average number of common nodes.

## 4. Algorithmic approach

There are two different ways for dealing with hard optimization problems: exact and heuristic approaches. The former are devoted to find the optimal value for the problem under consideration, without taking into account the required computing time to achieve that value. The latter, on the contrary, are designed for providing high quality solutions in reasonable computing times, without guaranteeing optimality.

In the context of CDP, the difficulty inherent to this family of problems, together with the need of reducing the computing times due to the continuous evolution of the social networks, make exact approaches not suitable for real-life networks, mainly due to the vast size of the solution space. Therefore, heuristic algorithms are a good option to be considered, given that they do not explore the entire search space, but certain promising regions of it. This behavior has the disadvantage that optimality cannot be guaranteed, but it is possible to achieve high quality solutions in reduced computing times. Another disadvantage of heuristic approaches is that a heuristic algorithm can eventually get trapped in a local optimum of the region under exploration. In this point is where metaheuristic algorithms come into play, been able to guide heuristics through the search space in a more intelligent way, allowing them to escape from local optima, thus exploring further regions of the search space. In this paper, we adapt the Iterated Greedy (IG) metaheuristic [43] to deal with the OCDP. IG is able to escape from local optima by partially destroying a solution and reconstructing it iteratively, allowing to explore a wider region of the search

space. This metaheuristic has been successfully applied in different optimization problems, such as [37, 40, 44].

The Iterated Greedy metaheuristic requires from an initial feasible solution to start the search. To perform this initial construction, several options are usually considered: from a random construction, to a greedy approach, through methodologies that hybridize both approaches. In this work, we apply a Greedy Randomized Adaptive Search Procedure (GRASP), originally presented in [10] and formally defined in [11] to generate the initial solution. GRASP consists of a multi-start procedure that tries to escape from local optima by restarting the search from a completely new point.

### *4.1. Initial solution*

As it was aforementioned, the initial solution for the Iterated Greedy is generated using GRASP, which consists of two differentiated phases: construction and improvement. In the construction phase, we try to generate from scratch a high quality solution, while the improvement phase is responsible for reaching a local optimum with respect to certain neighborhood starting from the constructed solution.

#### *4.1.1. Constructive procedure*

To construct an initial solution for OCDP, we must define a criterion that allows us to select the most appropriate communities for each node in the network. Algorithm 1 shows a pseudocode of the proposed constructive procedure.

The algorithm starts with an empty solution $S$ (step 1), where the nodes are not assigned to any community. Then, the candidate list $CL$ is constructed with all the nodes in the graph $G$ (step 2). The method iterates until every node has been assigned to, at least, a single community, i.e., $CL$ is empty (steps 3-13).

In each iteration, the constructive method selects the next node to start a new community. In order to do so, every node $v$ is evaluated following a greedy function $g(v)$ which determines the suitability of starting the community from $v$. The greedy function selected in this work is PageRank [35], originally proposed

**Algorithm 1** *Construction*$(G = (V, E), \alpha)$.

1: $S \leftarrow \emptyset$
2: $CL \leftarrow V$
3: **while** $CL \neq \emptyset$ **do**
4: $\quad g_{\min} \leftarrow \min_{v \in CL} g(v)$
5: $\quad g_{\max} \leftarrow \max_{v \in CL} g(v)$
6: $\quad \mu \leftarrow g_{\max} - \alpha \cdot (g_{\max} - g_{\min})$
7: $\quad RCL \leftarrow \{v \in CL : g(v) \geq \mu\}$
8: $\quad v \leftarrow Random(RCL)$
9: $\quad L \leftarrow DMF(v, G)$
10: $\quad CL \leftarrow CL \setminus L$
11: $\quad C \leftarrow InducedSubgraph(L, G)$
12: $\quad S \leftarrow S \cup \{C\}$
13: **end while**
14: **return** $S$

to rank the importance of a web page on the Internet, based on their incoming links (i.e., the number of web pages that reference the one under evaluation). Following this idea, the PageRank of every node in the *CL* is evaluated to decide which node has more relevance in the network, and therefore it should occupy a prior position in the *CL*, since this metric must be maximized. The PageRank provides a precise idea of which nodes have a central position in the potential communities, given that a node with a high PageRank will commonly have a high degree and several connections with other nodes that will usually be part of the same community. Given that in this work the networks analyzed have an invariant structure, the PageRank value for a node is only needed to be calculated once, reducing the computational effort.

The minimum ($g_{\min}$) and maximum ($g_{\max}$) value of this greedy function are computed (steps 4-5), with the aim of calculating a threshold $\mu$ (step 6) that limits the minimum value of PageRank that a node must have in order to be considered in the restricted candidate list *RCL*. The elements allowed to enter in the *RCL* are conditioned by an $\alpha$ parameter, resulting in a subset of the candidate nodes that are currently in the *CL*. The $\alpha$ value ranges from 0 to 1. On the one hand, the closer the value to 0, the more random the algorithm is, given that a larger number of elements that belongs to *CL* are able to enter

in the *RCL*. On the other hand, the closer the value to 1, the more greedy algorithm is, since only the nodes with the largest PageRank value are included in the *RCL*.

Once the *RCL* is defined, a node $v$ is randomly selected from it (step 8), considering $v$ as the origin of a new community. To build this community, we apply the *DMF* algorithm [30]. This algorithm applies a dynamic membership function to locally decide which node belongs to the community under construction. In particular, a breadth-first search is applied, starting from $v$, and includes in the community under construction those nodes that improve the ratio between intra-community and inter-community edges of the community under construction. The breadth-first search stops when no nodes found in the traversal are suitable to be included in the community. A more detailed explanation of its implementation can be found in [30]. Notice that this method is suitable for the overlapping nature of the OCDP, since a node that has already been included in a certain community, can be later included in a different one if it satisfies the criterion used during the breadth-first search when starting from a different node. The *DMF* method returns a set of nodes $L$, which is used to update the *CL* by avoiding that the nodes involved in the new community are considered as starting nodes for any new communities (step 10). Then, in step 11 it is constructed a community as the induced subgraph $C = G[L]$ formed with the vertices in $L$ and all of the edges connecting pairs of vertices in $L$. After that, the new community $C$ is included in the solution under construction $S$ (step 12). The method finally returns the constructed solution $S$ (step 14).

Notice that the randomness included in the selection of the next node from the *RCL* allows the algorithm to generate diverse solutions, which will eventually lead the method to find more promising regions of the search space.

#### *4.1.2. Improvement phase*

Once an initial solution is generated using the constructive procedure, the improvement phase is responsible for finding a local optimum with respect to a certain neighborhood by applying a local search procedure. It is conformed by

four main elements: the move operator considered in the search, the neighborhood to be explored, the strategy used to traverse the neighborhood, and the order in which the neighborhood is explored (if necessary).

First of all, the operator considered in this work consists in assigning a given vertex $u$ to a new community $C_j$, evaluating, at the same time, whether the removal from one of its current communities $C_i$ is positive or not. Specifically, the removal from its original community is performed if and only if the percentage of inter-community edges minus the percentage of intra-community edges of node $u$ in the current community is greater than a given threshold $\tau$, which is a parameter of the local search (see Section 5 for a detailed analysis of the effect of this parameter in the procedure). Otherwise, $u$ stays in the current community $C_i$ but it is also incorporated in $C_j$. The rationale behind this idea is that if a node has more edges to other communities than to nodes in the same community, then it should not belong to that community, since it is more related to nodes in other communities.

Notice that the node $u$ can eventually belong to additional communities, but the move only affects to one of those communities, $C_i$, in order to maintain the overlapping structure of the OCDP. Given a node $u$, assigned to a community $C_i$, that is being moved to a community $C_j$, and the threshold $\tau$, the move operator is formally defined as:

$$Move(u, C_i, C_j, \tau) = \begin{cases} C_i \leftarrow C_i \setminus \{u\} \\ C_j \leftarrow C_j \cup \{u\} \end{cases} \quad \text{if } \frac{E_{\rightarrow}(u,C_i)}{d_u} - \frac{E_{\leftarrow}(u,C_i)}{d_u} > \tau$$
$$C_j \leftarrow C_j \cup \{u\} \quad \text{otherwise} \tag{4}$$

The objective of this move is to evaluate if a node could be part of a better community than the one in which it is located. To this end, the ratio between intra-community and inter-community edges is computed, since the local search is based in the aforementioned idea that a good community structure is con-

formed by this situation: all nodes belonging to a community should be highly interconnected among them and sparsely connected to nodes in other communities.

The second element to be defined in a local search is the neighborhood to explore. In this case, the neighborhood of a given solution $S$ is conformed with all solutions that can be reached by making a single move. Mathematically,

$$N(S) = \{S' \leftarrow Move(u, C_i, C_j, \tau) \quad \forall u \in V, \quad \forall C_i,\ C_j \in S : u \in C_i \wedge u \notin C_j\} \tag{5}$$

It is worth mentioning that the exploration of this neighborhood maintains the overlapping nature of the problem, considering that a single node can belong to more than one community.

The third element required to define a local search is the strategy followed to explore the neighborhood. Two main strategies have been traditionally considered: first and best improvement. On the one hand, first improvement performs the first move that leads to a better solution than the incumbent one. On the other hand, the best improvement explores the complete neighborhood, performing the move that leads to the best solution in the neighborhood under exploration. Best improvement is considerably more computationally demanding than first improvement, since it requires to explore the complete neighborhood. Furthermore, the evaluation of a solution for the OCDP after performing a move is rather time-consuming. Therefore, we select first improvement strategy with the aim of reducing the computing time required by the local search procedure.

In the first improvement strategy the order in which the neighborhood is explored might be relevant since it could leads the search to explore some solutions before others, thus biasing the search to certain regions of the search space. In order to reduce this bias, the local search procedure randomly explores the corresponding neighborhood.

Once we have defined the four main elements of the proposed local search

for the OCDP, we propose a classical acceptance criterion to move to a new solution. Specifically, in each iteration, the current solution is replaced by the current one if a neighbor improves upon the objective function. The search ends when all neighbor solutions are worse in therms of the objective function, meaning that a local optimum is found.

*4.2. Iterated Greedy*

Iterated Greedy (IG) is a metaheuristic framework originally proposed in [43] in the context of job scheduling problem which has been used for a wide variety of problems. We refer the reader to [8, 28, 37] for recent successful research related to IG for different applications, even in a multi-objective optimization problem. The main idea of IG consists in balancing diversification and intensification by randomly destructing a feasible solution and, then, reconstructing it in a greedy manner to eventually reach new regions of the search space, thus leading to better solutions.

The traditional IG framework is based on two well-differentiated stages: destruction phase and reconstruction phase. Although canonical IG implementations do not consider any further improvement of the reconstructed solution, the use of a local improvement method is extended to find a local optimum starting from the reconstructed solution. With the aim of finding high quality solutions, we follow this approach in this work. Algorithm 2 shows the pseudocode of the proposed IG.

**Algorithm 2** $IG(S, \beta, \gamma)$.

1: **for** $i \in 1 \dots \gamma$ **do**
2: $S' \leftarrow Destruct(S, \beta)$
3: $S'' \leftarrow Reconstruct(S')$
4: $S''' \leftarrow LocalImprove(S'')$
5: **if** $M_O(S''') > M_O(S)$ **then**
6: $S \leftarrow S'''$
7: **end if**
8: **end for**

The method requires from three input parameters: $S$, the initial solution, $\beta$, the percentage of elements that are removed in the destruction phase, and $\gamma$,

the number of iterations to be performed. The initial solution in the traditional scheme of IG can be generated at random or following a more intelligent approach that leverages the information that can be extracted from the problem definition. In the context of OCDP, the initial solution is generated by considering the method described in Section 4.1. Therefore, the input solution $S$ is a local optimum with respect to the neighborhood $N(S)$ defined in Section 4.1.2. The use of GRASP to generate an initial solution allows IG to start the search from a promising region of the search space.

The method iteratively destructs and reconstructs a given solution until reaching a stopping criterion (steps 1-8). In the context of IG, two of the most extended stopping criteria are the computing time and the number of iterations. With the aim of providing a highly scalable approach, we decide to use the number of iterations ($\gamma$) as stopping criterion since the computing time highly depends on the complexity of the network analyzed.

Each iteration is conformed with two different stages: destruction and reconstruction. First, a percentage of the solution under exploration is destructed (step 2) generating an unfeasible solution $S'$, then, it is reconstructed (step 3) to recover again feasibility. After this perturbation, the resulting solution $S''$ is not necessarily a local optimum with respect to any neighborhood, so applying a local optimization method may lead to an improved solution. Following this idea, we apply the local search method described in Section 4.1.2 (step 4). After each complete iteration (destruction, reconstruction, improvement) of the for-loop it is tested whether the new local optimum $S'''$ is better than $S'$ or not when considering the overlapped modularity. If so, the incumbent solution is updated (step 6), starting the next iteration from the new solution; otherwise, IG discards $S'''$, continuing the search from the previous solution.

In the traditional scheme of IG, the destruction stage is focused on diversification, since the removal of elements is usually performed at random, allowing the algorithm to reach different regions of solution space, thus escaping from local optima. Then, the reconstruction phase is intended to increase intensification by adding elements to the solution explored in a greedy manner until

recovering a feasible solution. In order to evaluate the influence of intensification and diversification in the complete IG algorithm, we propose two different destruction approaches and two reconstruction methods, each one of them focused on a different goal. In the experimental phase (Section 5), we analyze all these possibilities, and their performances in this concrete problem.

*4.2.1. Destruction phase*

As it was aforementioned, the destruction phase consists in removing a certain number of elements of a given solution, with the aim of exploring further regions of the search space. In the context of OCDP, this perturbation consists in removing a node from all communities that it belongs to. More formally, given a node $u \in V$, let $\mathcal{C}$ the subset of communities of a feasible solution where the node $u$ belongs to. Then, for each $C_i \in \mathcal{C}$ with $C_i = (V_i, E_i)$ the removing operation updates each community as follows: $V_i \leftarrow V_i \setminus \{u\}$ and $E_i \leftarrow E_i \setminus \{(u, v) \in E_i : v \in V_i\}$ for $1 \leq i \leq \mathcal{C}$.

Having defined the method to remove elements from a solution in the destruction phase, it is necessary to define how many elements are being removed in this stage. The search parameter $\beta$, which is analyzed in Section 5, is responsible for this task. In particular, in each destruction stage, $\beta \cdot n$ elements are removed from the solution. The definition of $\beta$ as a parameter of the network size increases the scalability of the proposal, since the final number of elements removed depends on the network size. Otherwise, a large value of $\beta$ may result in the removal of many nodes in small networks and a negligible number of elements in large-scale networks.

Finally, it is necessary to define which elements are removed during the destruction phase. Traditional IG implementations consider the random removal of elements since it leads to increase diversification. Apart from this traditional implementation, we propose the use of a greedy destruction strategy, which removes those nodes that have a large ratio of inter-community edges with respect to its degree, i.e., $E_{\rightarrow}(u)/d_u$, for each community to which it belongs to. Therefore, when considering the random destruction, $\beta \cdot n$ elements are

randomly selected from the network, removing them from every community in which they participate, while in the greedy approach, the selected $\beta \cdot n$ elements are those with the maximum ratio of inter-community edges.

*4.2.2. Reconstruction phase*

Contrary to the destruction phase, the reconstruction phase is designed to recover a feasible solution from a destructed one. In order to do so, we must reassign the previously removed nodes to a new community, which can be the original or a completely different one. Notice that, in order to recover feasibility, each one of the $\beta \cdot n$ elements needs to be reassigned to a community.

It is worth mentioning that this stage assigns exactly a single community to each unassigned node. Although this approach could be opposite to the OCDP fundamentals, the local search applied after the reconstruction phase is responsible for assigning more than one community to the nodes that are suitable for it. This idea allows the algorithm to find the most appropriate community for each removed node.

In this case, the traditional IG scheme recommends adding the new elements to the solution following a greedy strategy, increasing the intensification to counter the diversification included in the destruction phase. Again, we propose both strategies to add a node to the solution. On the one hand, the random reconstruction proposed basically adds each removed node to a random community among the available ones. On the other hand, the greedy approach selects, for each node, the most appropriate community by selecting the one with the largest ratio of intra-community edges with respect to the degree of the node, i.e., $E_{\leftarrow}(u)/d_u$.

## 5. Experiments and results

This section shows the experiments carried out to test the effectiveness and efficiency of the proposed approach and perform the comparison between the most recent algorithm found in the literature [54]. Both algorithms have been executed in the same computer: an AMD Ryzen 5 3600 AM4 core (3.6 GHz)

with 16 GB RAM. The proposed algorithm is implemented using Java 9 while the source code of the previous method[1] is implemented in Matlab.

The dataset for these experiments is conformed with two kinds of instances: synthetic and real-world networks. For the former, we have used the network generator developed in [25] to construct synthetic instances[2]. In these networks the community size and node degree distribution follow a power-law highly configurable. We have considered different configurations for the network generator, and we have generated different network instances for each configuration (totalizing 57 different networks). In particular, we have considered networks with a range between 100 and 10000 nodes, and the edge probability $p$ is defined as $p \in [0.1, 0.3]$ with an interval of 0.1 to simulate real-life instances.

In order to test the behavior of our algorithm over real-world networks, we have selected a set of 11 well-known instances widely used in Social Network Analysis problems:

- Astro-ph [33], with 16706 nodes and 121251 edges
- Cond-math [33], with 16726 nodes and 47594 edges
- Dolphin social network [31], with 64 nodes and 159 edges
- American college football [13], with 115 nodes and 613 edges
- Hep-th [33], with 8361 nodes and 15751 edges
- Zacharys karate club [56], with 32 nodes and 78 edges
- Netscience [34, 42], with 1589 nodes and 2742 edges
- Polbooks [42], with 105 nodes and 441 edges
- Power [50], with 4941 nodes and 6594 edges
- School1 (sp-data-school-day-1) [2], with 236 nodes and 5899 edges

[1] We really thank Xu et al. [54] for kindly sent us the source code of EADP.
[2] Lancichinetti, Fortunato, and Radicchi (LFR) networks.

- School2 (sp-data-school-day-2) [2], with 238 nodes and 5539 edges

With the aim of facilitating further comparisons, all instances and the source code of the proposed algorithms can be downloaded from `https://grafo.etsii.urjc.es/ocdp`

The experimentation is divided into two different steps. On the one hand, we perform several preliminary experiments to evaluate the best configuration for our algorithms. On the other hand, we compare our best proposal with the most recent algorithm found in the state of the art. All the experiments report the following metrics: Avg., the average overlapped modularity obtained with the algorithm in the experiment; Dev. (%), the average deviation with respect to the best solution found in the experiment; Time (s), the total computing time required by each algorithm measured in seconds; and #Best, the number of times that an algorithm matches the best solution in the experiment.

### *5.1. Preliminary experimentation*

The preliminary experiments were performed to set the values of the key search parameters of our method as well as to show the merit of the proposed strategies. As it is customary, we select a subset of all instances (40 out of 68) with the aim of avoiding the overfitting of the algorithm.

The first experiment is performed with the aim to establish the best value of the $\alpha$ parameter in GRASP algorithm (see Section 4.1). Concretely, we test $\alpha = \{0.25, 0.50, 0.75, RND\}$, where $RND$ indicates that the value is selected randomly in the range $[0, 1]$ for each constructed solution. By using these values, we cover the full range: from a mostly greedy to a mostly random approach. We execute the constructive algorithm for 100 independent iterations, returning the best solution found for each instance under evaluation. Table 1 shows the obtained results for each $\alpha$ value under consideration.

As it can be seen, the best results are obtained when the $\alpha$ parameter has a value of 0.25. Even though the average of the objective function is quite similar for all approaches, looking at the computing times that one configuration reaches the best solution, it can be clearly seen that the more greedy approach is superior

| $\alpha$ | Avg. | Dev. (%) | Time(s) | #Best |
|---|---|---|---|---|
| 0.25 | **0.3282** | **1.58** | **33.25** | **31** |
| 0.50 | 0.3258 | 1.84 | 33.79 | 8 |
| 0.75 | 0.3147 | 5.20 | 34.70 | 11 |
| *RND* | 0.3241 | 2.82 | 34.14 | 10 |

Table 1: Comparison of the average results obtained by the constructive procedure with different values for the $\alpha$ parameter. Best results are highlighted with bold font.

to the others. In addition, the minimum computation time also corresponds to this strategy, followed by the configuration for $\alpha = 0.50$. Regarding at the average of deviation, these two settings are too close again, obtaining a low value of deviation in those instances where they can not reach the best solution. Therefore, it can be concluded that the best configuration of GRASP algorithm is the one that has $\alpha = 0.25$, so we use this setting to the constructive method.

The second experiment is designed to test the best $\tau$ parameter in the local search procedure described in Section 4.1.2. Once again, we execute 100 independent iterations of the algorithm with the best constructive configuration found in previous experiment and testing different values for the $\tau$ parameter. The best solution found for each instance under evaluation is saved. In Table 2 the obtained metrics are exposed.

| $\tau$ | Avg. | Dev. (%) | Time (s) | #Best |
|---|---|---|---|---|
| 0.2 | **0.3328** | **0.43%** | **33.60** | **38** |
| 0.3 | 0.3327 | 0.45% | 33.70 | 34 |
| 0.4 | 0.3326 | 0.85% | 34.90 | 28 |
| 0.5 | 0.3321 | 1.10% | 35.04 | 28 |

Table 2: Comparison of the average results obtained by the local search procedure with different values for the $\tau$ parameter. Best results are highlighted with bold font.

Looking at these results, it can be concluded that $\tau = 0.2$ is the best configuration for the local search procedure. This means that the smaller the percentage difference between inter-community and intra-community edges in a community that is established when producing overlaps, the better results in terms of evaluation metric will be obtained. In particular, with this configuration the algorithm reaches the best solution for 38 out of 40 instances, and it

is only at 0.43% difference on average with respect to the best solution found when it is not capable to reach it. Furthermore, using this configuration the computational time is lower.

The next experiment is devoted to test which configuration of Iterated Greedy algorithm works better in this problem. As it is described in Section 4.2, we can follow four different approaches in Iterated Greedy: *Greedy Greedy* (GG), which is designed with the phases of destruction and reconstruction following a greedy paradigm; *Random Random* (RR), following a random approach in both destruction and reconstruction phases; *Greedy Random* (GR), with greedy destruction and random reconstruction; and finally, *Random Greedy* (RG), with random destruction and greedy reconstruction. Table 3 shows the number of times in which certain algorithm reaches the best result, with different $\beta$ values, for each instance.

| $\beta$ value / Algorithm | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| Iterated Greedy GG | **8** | 8 | 9 | **10** | 9 |
| Iterated Greedy GR | 6 | 8 | 6 | 7 | 8 |
| Iterated Greedy RG | 7 | **9** | **10** | **10** | **14** |
| Iterated Greedy RR | **8** | 7 | 9 | 7 | 8 |

Table 3: Number of times that best solution is reached by the Iterated Greedy procedure with different approaches and different values for the $\beta$ parameter. Best results are highlighted with bold font.

As it can be seen, the best results are reached with a RG approach and $\beta = 0.5$ value, reaching the best value in 14 of total instances used in the preliminary experiments. Additional experiments were performed in order to study if the results when a higher $\beta$ value is chosen, but the results show that the extra computing time it is not worth using values higher than 0.5, since the results do not are improved.

Summarizing, the best variant is composed of a constructive algorithm with $\alpha = 0.25$, the local search configured with $\tau = 0.2$, the IG algorithm that follows the RG paradigm, and the destruction/reconstruction parameter $\beta = 0.5 \cdot n$.

*5.2. Final experiments*

In this final experimentation, the proposed algorithm is compared with the best method found in the literature [54]. In particular, the authors propose an Extended Adaptive Density Peaks (EADP) algorithm for detecting overlapped communities. As it has been previously stated, the experiments have been performed over the complete set of 68 instances. Firstly, by comparing over the set of LFR instances (57) and then over the real-world networks (11). For sake of fairness, we have executed EADP with the parameters suggested by the authors, adapting the number of iterations of our approach ($\gamma$ parameter) to fulfill, as much as possible, its CPU time.

In Table 4, we report the results for the synthetic LFR networks grouped by the number of nodes in each network considered. As it can be derived from the table, IG consistently produces better results than EADP. In particular, it is able to reach the best solution in all instances but 2, which belong to the set of small instances (those with a number of nodes ranging from 0 to 2500). Indeed, in the instances in which the best value is not found, the average percentage deviation is close to 3%, indicating that even in those networks IG generates a solution whose quality remains really close to the best value achieved with EADP. On the contrary, the average percentage deviation of EADP is close to 37%, indicating that, in those instances in which it does not reach the best value, it is not necessarily close to it, being IG a more robust approach. Regarding the computing times, we can see that both methods are highly scalable with the increase in the number of nodes in the network, being IG slightly slower than EADP but in the same order of magnitude. Nevertheless, the remarkable increase in the quality of the solutions could justifies the increase in the computing time. The average objective function value obtained by IG on average (0.364) indicates that it is able to find solutions with more community structure than those found by EADP, which provides a smaller average objective function value (0.231).

Finally, both IG and EADP are tested over the dataset conformed with 11 real-life networks described at the beginning of this section. The results are

| | Iterated Greedy | | | | EADP | | | |
|---|---|---|---|---|---|---|---|---|
| | Avg. | Dev (%) | Time (s) | #Best | Avg. | Dev (%) | Time (s) | #Best |
| $0 \leq n < 2500$ | **0.319** | **3.399** | 5.501 | **13** | 0.216 | 34.855 | **0.507** | 2 |
| $2500 \leq n < 5000$ | **0.383** | **0.000** | 27.736 | **15** | 0.238 | 37.061 | **3.243** | 0 |
| $5000 \leq n < 7500$ | **0.377** | **0.000** | 48.293 | **18** | 0.236 | 37.102 | **10.781** | 0 |
| $7500 \leq n \leq 10000$ | **0.377** | **0.000** | 87.124 | **9** | 0.234 | 37.171 | **36.943** | 0 |
| *Average* | **0.364** | **0.850** | 42.164 | **55** | 0.231 | 36.547 | **12.869** | 2 |

Table 4: Comparison of Iterated Greedy (IG) and EADP configured as stated in [54] for the synthetic LFR networks.

shown in table 5. In order to facilitate the comparison, we report the same information as [54], i.e., the overlapped modularity, $M_O$, and the associated computing time in seconds, Time (s).

| | Iterated Greedy | | | EADP | | |
|---|---|---|---|---|---|---|
| | $M_O$ | Time (s) | #Best | $M_O$ | Time (s) | #Best |
| astro-ph | 0.320 | 404.900 | 1 | 0.001 | 176.380 | 0 |
| cond-mat | 0.408 | 91.727 | 1 | 0.004 | 226.380 | 0 |
| dolphins | 0.082 | 0.102 | 0 | 0.154 | 0.505 | 1 |
| football | 0.246 | 0.701 | 0 | 0.261 | 0.988 | 1 |
| hep-th | 0.356 | 28.637 | 1 | 0.019 | 30.316 | 0 |
| karate | 0.099 | 0.060 | 1 | 0.000 | 0.437 | 0 |
| netscience | 0.584 | 1.763 | 1 | 0.053 | 0.891 | 0 |
| polbooks | 0.101 | 0.178 | 0 | 0.141 | 0.527 | 1 |
| power | 0.230 | 13.573 | 1 | 0.001 | 6.647 | 0 |
| School1 | 0.213 | 3.086 | 0 | 0.236 | 0.814 | 1 |
| School2 | 0.196 | 3.058 | 0 | 0.234 | 0.408 | 1 |
| *Average* | **0.258** | 49.799 | **6** | 0.100 | **40.390** | 5 |

Table 5: Comparison of IG with EADP configured as stated in [54] for the real world networks.

In this case, the results obtained with both algorithms are more similar, but IG still emerges as the best option when dealing with more complex networks. In particular, for the largest instances IG is able to reach the best value while EADP obtains slightly better values for the smallest instances. It is worth mentioning that IG is really close to the best value in those instances in which it does not reach it, while EADP is still far from the best value in the analogous case. Both algorithms require less than 50 seconds on average to finish, but the comparison of the average objective function value obtained by IG (0.258) and EADP (0.100) suggest that IG is able to find better community structures than

EADP in these real-life instances

## 6. Conclusions

In this paper, we have proposed a new meta-heuristic method for overlapping community detection in social network based on hybridizing Greedy Randomized Adaptive Search Procedure (GRASP) and Iterated Greedy (IG). We use as evaluation metric the modularity, a well-known social network analysis metric, adapted to the context of overlapping communities.

The performed experiments show that the proposed algorithm is able to produce high quality solutions in terms of Overlapping Community Detection Problem (OCDP), outperforming the best method found in the literature, which is based on an adaptation of Density Peaks Clustering algorithm (EADP, Extended Adaptive Density Peaks). Even the computing time in the proposed algorithm is slightly higher than the obtained with EADP, the quality in terms of objective function justifies the contribution of the proposal.

In future works, it would be interesting to analyze the performance of the approach in other similar problems, like Community Detection without possibility of overlapping, or even adapt it to a multi-objective scenario or dynamic social networks.

## Acknowledgments

This research was funded by "Ministerio de Ciencia, Innovación y Universidades" under grant ref. PGC2018-095322-B-C22, "Comunidad de Madrid" and "Fondos Estructurales" of European Union with grant refs. S2018/TCS-4566, Y2018/EMT-5062.

## References

[1] Aggarwal, C. C. (2011). An introduction to social network data analytics. In *Social network data analytics*, pages 1–15. Springer.

[2] Bhat, S. Y. and Abulaish, M. (2014). Hoctracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks. *IEEE Transactions on Knowledge and Data engineering*, 27(4):1019–1013.

[3] Camacho, D., Panizo-LLedot, A., Bello-Orgaz, G., Gonzalez-Pardo, A., and Cambria, E. (2020). The four dimensions of social network analysis: An overview of research methods, applications, and software tools. *Information Fusion*, 63:88 – 120.

[4] Chakraborty, T., Ghosh, S., and Park, N. (2019). Ensemble-based overlapping community detection using disjoint community structures. *Knowledge-Based Systems*, 163:241–251.

[5] Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6):066111.

[6] Cohen, K., Johansson, F., Kaati, L., and Mork, J. C. (2014). Detecting linguistic markers for radical violence in social media. *Terrorism and Political Violence*, 26(1):246–256.

[7] Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Eleventh international AAAI conference on web and social media*.

[8] Delgado-Antequera, L., Caballero, R., Sánchez-Oro, J., Colmenar, J., and Martí (2020). Iterated greedy with variable neighborhood search for a multiobjective waste collection problem. *Expert Systems with Applications*, 145:113101.

[9] Ding, X., Zhang, J., and Yang, J. (2018). A robust two-stage algorithm for local community detection. *Knowledge-Based Systems*, 152:188–199.

[10] Feo, T. A. and Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67 – 71.

[11] Feo, T. A., Resende, M. G. C., and Smith, S. H. (1994). A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Operations Research*, 42(5):860–878.

[12] Fernandez, M., Gonzalez-Pardo, A., and Alani, H. (2019). Radicalisation influence in social media. *Journal of Web Science*, 6.

[13] Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826.

[14] Gonzalez-Pardo, A., Jung, J. J., and Camacho, D. (2017). Aco-based clustering for ego network analysis. *Future Generation Computer Systems*, 66:160 – 170.

[15] Gregory, S. (2007). An algorithm to find overlapping community structure in networks. In Kok, J. N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., and Skowron, A., editors, *Knowledge Discovery in Databases: PKDD 2007*, pages 91–102. Springer Berlin Heidelberg.

[16] Gregory, S. (2008). A fast algorithm to find overlapping communities in networks. In Daelemans, W., Goethals, B., and Morik, K., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 408–423. Springer Berlin Heidelberg.

[17] Gregory, S. (2010). Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018.

[18] Guiñazú, M. F., Cortés, V., Ibáñez, C. F., and Velásquez, J. D. (2020). Employing online social networks in precision-medicine approach using information fusion predictive model to improve substance use surveillance: A lesson from twitter and marijuana consumption. *Information Fusion*, 55:150–163.

[19] Harrigan, P., Evers, U., Miles, M., and Daly, T. (2017). Customer engagement with tourism social media brands. *Tourism Management*, 59:597–609.

[20] Hudson, S., Huang, L., Roth, M. S., and Madden, T. J. (2016). The influence of social media interactions on consumer–brand relationships: A three-country study of brand perceptions and marketing behaviors. *International Journal of Research in Marketing*, 33(1):27–41.

[21] Jaki, S. and Smedt, T. D. (2019). Right-wing german hate speech on twitter: Analysis and automatic detection. *CoRR*, abs/1910.07518.

[22] Javed, M. A., Younis, M. S., Latif, S., Qadir, J., and Baig, A. (2018). Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111.

[23] Kim, L., Fast, S. M., and Markuzon, N. (2019). Incorporating media data into a model of infectious disease transmission. *PloS one*, 14(2).

[24] Kumpula, J. M., Kivelä, M., Kaski, K., and Saramäki, J. (2008). Sequential algorithm for fast clique percolation. *Phys. Rev. E*, 78:026109.

[25] Lancichinetti, A., Fortunato, S., and Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78:046110.

[26] Lara-Cabrera, R., Gonzalez-Pardo, A., Benouaret, K., Faci, N., Benslimane, D., and Camacho, D. (2017). Measuring the radicalisation risk in social networks. *IEEE Access*, 5:10892–10900.

[27] Lázár, A., Abel, D., and Vicsek, T. (2010). Modularity measure of networks with overlapping communities. *EPL (Europhysics Letters)*, 90(1):18001.

[28] Li, J.-Q., Du, Y., Gao, K.-Z., Duan, P.-Y., Gong, D.-W., Pan, Q.-K., and Suganthan, P. (2021). A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem. *IEEE Transactions on Automation Science and Engineering*.

[29] Lozano, M., García-Martínez, C., Rodriguez, F. J., and Trujillo, H. M. (2017). Optimizing network attacks by artificial bee colony. *Information Sciences*, 377:30 – 50.

[30] Luo, W., Zhang, D., Jiang, H., Ni, L., and Hu, Y. (2018). Local community detection with the dynamic membership function. *IEEE Transactions on Fuzzy Systems*, 26(5):3136–3150.

[31] Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., and Dawson, S. M. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405.

[32] Müller, K. and Schwarz, C. (2018). Fanning the flames of hate: Social media and hate crime. *Available at SSRN 3082972*.

[33] Newman, M. E. (2001). The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409.

[34] Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

[35] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

[36] Pastor-Satorras, R., Castellano, C., Van Mieghem, P., and Vespignani, A. (2015). Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925.

[37] Pérez-Peló, S., Sanchez-Oro, J., Lopez-Sanchez, A. D., and Duarte, A. (2019a). A multi-objective parallel iterated greedy for solving the p-center and p-dispersion problem. *Electronics*, 8(12):1440.

[38] Pérez-Peló, S., Sánchez-Oro, J., Martín-Santamaría, R., and Duarte, A. (2019b). On the analysis of the influence of the evaluation metric in community detection over social networks. *Electronics*, 8(1):23.

[39] Queiroga, E., Subramanian, A., and dos Anjos F. Cabral, L. (2018). Continuous greedy randomized adaptive search procedure for data clustering. *Applied Soft Computing*, 72:43–55.

[40] Ribas, I., Companys, R., and Tort-Martorell, X. (2019). An iterated greedy algorithm for solving the total tardiness parallel blocking flow shop scheduling problem. *Expert Systems with Applications*, 121:347–361.

[41] Rodriguez, A. and Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496.

[42] Rossi, R. and Ahmed, N. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

[43] Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

[44] Sánchez-Oro, J. and Duarte, A. (2018). Iterated greedy algorithm for performing community detection in social networks. *Future Generation Computer Systems*, 88:785–791.

[45] Smith, K. P. and Christakis, N. A. (2008). Social networks and health. *Annu. Rev. Sociol*, 34:405–429.

[46] Sohrabi, S., Ziarati, K., and Keshtkaran, M. (2020). A greedy randomized adaptive search procedure for the orienteering problem with hotel selection. *European Journal of Operational Research*, 283(2):426–440.

[47] Su, Y., Zhou, K., Zhang, X., Cheng, R., and Zheng, C. (2021). A parallel multi-objective evolutionary algorithm for community detection in large-scale complex networks. *Information Sciences*, 576:374–392.

[48] Swani, K., Milne, G. R., Brown, B. P., Assaf, A. G., and Donthu, N. (2017). What messages to post? evaluating the popularity of social media communications in business versus consumer markets. *Industrial Marketing Management*, 62:77–87.

[49] Wang, Y., McKee, M., Torbica, A., and Stuckler, D. (2019). Systematic literature review on the spread of health-related misinformation on social media. *Social Science & Medicine*, 240:112552.

[50] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *Nature*, 393(6684):440–442.

[51] Whang, J. J., Gleich, D. F., and Dhillon, I. S. (2016). Overlapping community detection using neighborhood-inflated seed expansion. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1272–1284.

[52] Wu, X., Zhu, X., Wu, G.-Q., and Ding, W. (2013). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107.

[53] Xie, J., Kelley, S., and Szymanski, B. K. (2013). Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Comput. Surv.*, 45(4).

[54] Xu, M., Li, Y., Li, R., Zou, F., and Gu, X. (2019). EADP: An extended adaptive density peaks clustering for overlapping community detection in social networks. *Neurocomputing*, 337:287–302.

[55] Yang, J. and Leskovec, J. (2013). Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, page 587–596.

[56] Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473.

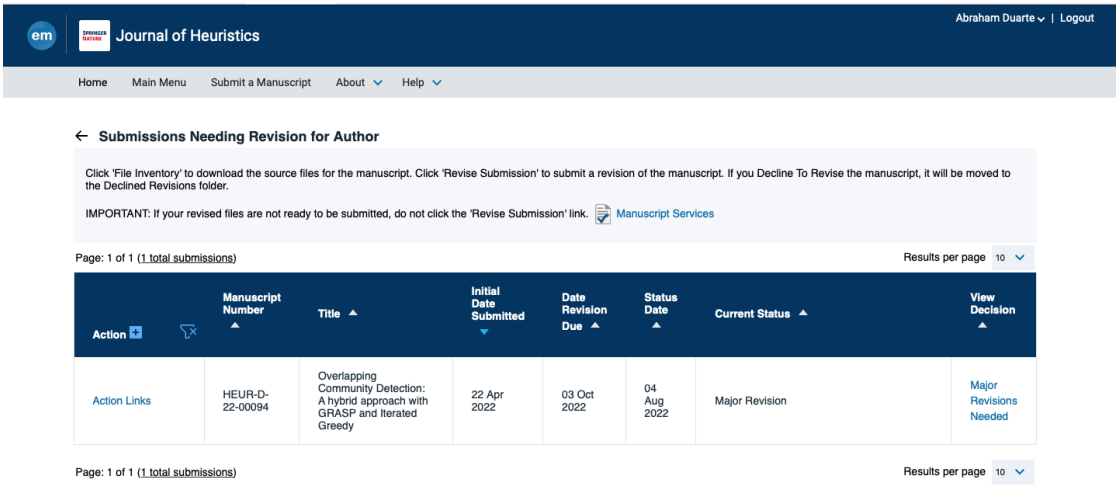

Journal of Heuristics
Abraham Duarte | Logout
Home
Main Menu
Submit a Manuscript
About
Help
Submissions Needing Revision for Author
Click 'File Inventory' to download the source files for the manuscript. Click 'Revise Submission' to submit a revision of the manuscript. If you Decline To Revise the manuscript, it will be moved to the Declined Revisions folder.
IMPORTANT: If your revised files are not ready to be submitted, do not click the 'Revise Submission' link.
Manuscript Services
Page: 1 of 1 (1 total submissions)
Results per page 10
Action
Manuscript Number
Title
Initial Date Submitted
Date Revision Due
Status Date
Current Status
View Decision
Action Links
HEUR-D-22-00094
Overlapping Community Detection: A hybrid approach with GRASP and Iterated Greedy
22 Apr 2022
03 Oct 2022
04 Aug 2022
Major Revision
Major Revisions Needed
Page: 1 of 1 (1 total submissions)
Results per page 10

Figure 10.1

# Part III

# Additional Publications published during thesis development

# Contents

# Chapter 11

# Journal articles indexed in the Journal of Citation Reports

Additional publications related to the problems and metaheuristics discussed in this dissertation have been published, either in terms of the methodology used, or in terms of the problems addressed. The purpose of this chapter is to present additional articles that have been published as a result of the research. Based on the date of publication, the articles are presented in different sections in ascending order.

## 11.1 A Multi-Objective Parallel Iterated Greedy for Solving the p-Center and p-Dispersion Problem

The journal paper associated to this part is:

- Pérez-Peló, S., Sánchez-Oro, J., López-Sánchez, A. D., & Duarte, A. (2019). "A Multi-Objective Parallel Iterated Greedy for Solving the p-Center and p-Dispersion Problem". Electronics, 8(12), 1440. (doi:10.3390/electronics8121440)
- Status: **Published.**
- Impact Factor (JCR 2019): 2.412
- Subject Category: Computer Science, Information Systems. Ranking 93/161 (Q3)

- Subject Category: Engineering, Electrical & Electronic. Ranking 145/273 (Q3)

## 11.2 A review on discrete diversity and dispersion maximization from an OR perspective

The journal paper associated to this part is:

- Martí, R., Martínez-Gavara, A., Pérez-Peló, S., & Sánchez-Oro, J. (2021). "A review on discrete diversity and dispersion maximization from an OR perspective". European Journal of Operational Research. (doi:10.1016/j.ejor.2021.07.044).

- Status: **Published.**

- Impact Factor (JCR 2020): 5.334

- Subject Category: Operations Research & Management Science. Ranking 15/84 (Q1)

## 11.3 A GRASP algorithm with Tabu Search improvement for solving the maximum intersection of $k$-subsets problem

The journal paper associated to this part is:

- Casado, A., Pérez-Peló, S., Sánchez-Oro, J., & Duarte, A. (2022). A GRASP algorithm with Tabu Search improvement for solving the maximum intersection of k-subsets problem. Journal of Heuristics, 28(1), 121-146. https://doi.org/10.1007/s10732-022-09490-8

- Status: **Published.**

- Impact Factor (JCR 2021): 2.247

- Subject Category: Computer Science, Artificial Intelligence. Ranking 95/190 (Q2)

## 11.4 Strategic oscillation for the balanced minimum sum-of-squares clustering problem

The journal paper associated to this part is:

- Martín-Santamaría, R., Sánchez-Oro, J., Pérez-Peló, S., & Duarte, A. (2022). "Strategic oscillation for the balanced minimum sum-of-squares clustering problem". Information Sciences, 585, 529-542. (doi:10.1016/j.ins.2021.11.048).
- Status: **Published.**
- Impact Factor (JCR 2020): 6.795
- Subject Category: Computer Science, Information Systems. Ranking 18/161 (Q1)

# Chapter 12

# Works presented in national and international conferences

This chapter lists the works presented in national and international conferences, differentiating between those that that have associated a Lecture Notes in Computer Science book chapter and those that does not have an associated publication.

## 12.1 Works with a Lecture Notes in Computer Science book chapter associated

1. Pérez-Peló, S., Sánchez-Oro, J., & Duarte, A. (2018, October). Detecting Weak Points in Networks Using Variable Neighborhood Search. In International Conference on Variable Neighborhood Search (pp. 141-151). Springer, Cham.

2. Pérez-Peló, S., Sánchez-Oro, J., & Duarte, A. (2018, November). A Metaheuristic Approach for the $\alpha$-separator Problem. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 336-343). Springer, Cham.

## 12.2 Works without a Lecture Notes in Computer Science book chapter associated

1. Peló, S. P., Oro, J. S., & Muñoz, A. D. (2018). Detección de puntos débiles en redes utilizando GRASP y Path Relinking. In XVIII Conferencia de la

Asociación Española para la Inteligencia Artificial (CAEPIA 2018): avances en Inteligencia Artificial. 23-26 de octubre de 2018 Granada, España (pp. 531-536). Asociación Española para la Inteligencia Artificial (AEPIA).

2. Peló, S. P., Oro, J. S., & Muñoz, A. D. (2018). On the analysis of the influence of the evaluation metric in community detection using GRASP. In XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2018): avances en Inteligencia Artificial. 23-26 de octubre de 2018 Granada, España (pp. 981-986). Asociación Española para la Inteligencia Artificial (AEPIA).

3. Pérez-Peló, S., Sánchez-Oro, J., & López-Sánchez, A.D (2019). The Bi-objective p-Center and p-Dispersion problem. Workshop of the EURO Working Group on Vehicle Routing and Logistics optimization (VeRoLog). 2-5 Jun 2019, Seville (Spain).

4. Pérez-Peló, S., Sánchez-Oro, J., & Duarte, A. (2018, October). Solving the Bi-objective p-Center and p-Dispersion problem using VNS. In International Conference on Variable Neighborhood Search (ICVNS). 3-5 Oct 2019, Rabat (Morocco).

5. Pérez-Peló, S., Sánchez-Oro, J., González-Pardo, A., & Duarte, A.(2020). Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9.

6. Pérez-Peló, S., Sánchez-Oro, J., & Duarte, A. (2021, March). Solving the Bi-objective p-Center and p-Dispersion problem using VNS. In International Conference on Variable Neighborhood Search (ICVNS). 22-24 Mar 2019, Abu Dhabi (United Arab Emirates).

7. Peló, S. P., Oro, J. S., & Muñoz, A. D. (2021). Detección de comunidades con un enfoque multi-objetivo utilizando VNS. In XIX Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2021): avances en Inteligencia Artificial. 22-24 de septiembre de 2021 Málaga, España.

# Part IV

# Appendix

# Appendix A

# Resumen en castellano

En una sociedad donde la instantaneidad está cada vez más establecida, existe una necesidad de obtener soluciones a problemas del mundo real de una forma rápida y precisa. Con respecto a la precisión, o más específicamente la optimalidad, la disciplina científica que lidia con este problema es la optimización. Este área del conocimiento puede verse como un punto de encuentro entre varias disciplinas, como la investigación operativa, la estadística, la informática o la inteligencia artificial. Existe una gran cantidad de problemas interesantes de la vida real que pueden ser abordados desde el punto de vista de la optimización: calcular la ruta más corta para ir de casa al trabajo, encontrar la mejor manera para colocar el máximo número de contenedores en un barco para enviar productos alrededor del mundo o saber cuáles son los puntos más débiles de una red de ordenadores para dedicar más recursos en su protección. Todos estos problemas pueden ser resueltos de dos maneras princpales: a través de algoritmos exactos o de algoritmos aproximados.

El principal problema al que se enfrentan los algoritmos exactos es que, cuando el espacio de soluciones a explorar es muy amplio, el tiempo de cómputo requerido para proporcionar una solución óptima al problema es inasumible. En este contexto, los algoritmos aproximados emergen como una alternativa, con la principal desventaja de que no garantizan alcanzar una solución óptima global. Sin embargo, si se selecciona la técnica adecuada, una solución de alta calidad puede garantizarse. Este es el caso de los algoritmos heurísticos.

El Problema de Detección de Comunidades (CDP) es un problema $\mathcal{NP}$-difícil que pertenece a la familia de problemas del Análisis de Redes Sociales (SNA). El principal objetivo en el CDP es agrupar usuarios dentro de una red social dependiendo de sus características, sus relaciones y otras propiedades de la red en

sí misma. Se dice que una buena solución para el CDP se caracteriza por una buena estructura de comunidad. La estructura de comunidad se considera buena cuando los grupos resultantes contienen nodos altamente conectados entre sí y escasamente conectados hacia nodos pertenecientes a otros grupos. Existen diferentes variantes del mismo problema en los que se consideran diferentes restricciones. Por ejemplo, el número de comunidades que contiene una solución es fijada *a priori*, o los nodos pueden ser asignados a diferentes grupos simultáneamente. El problema también se puede afrontar optimizando diferentes funciones objetivo, y teniendo en cuenta un único o múltiples objetivos. A pesar de todo lo anterior, el objetivo último siempre es el mismo: obtener soluciones que presentan una buena estructura de comunidad.

En esta Tesis Doctoral se proponen diferentes algoritmos heurísticos para resolver diferentes variantes del CDP. Se aplican diferentes metodologías, como la búsqueda de vecindad variable (VNS), *Iterated Greedy* (IG), o la búsqueda dispersa (SS). Cada propuesta se ha evaluado tanto contra redes sintéticas como del mundo real, para comprobar su utilidad y aplicación en estos contextos. Los resultados obtenidos superan las propuestas del estado del arte en todas las variantes del CDP estudiadas.

## A.1 Antecedentes

La evolución de las redes sociales en las últimas décadas ha despertado el interés de científicos de diferentes campos, desde la psicología hasta la informática. Millones de personas comparten constantemente toda su información personal y profesional en diferentes redes sociales [27]. Además, las redes sociales se han convertido en una de las fuentes de información más utilizadas, sobre todo por su capacidad de ofrecer al usuario contenidos en tiempo real. Las redes sociales no sólo son una nueva forma de comunicación, sino también una poderosa herramienta que puede utilizarse para recopilar información relacionada con cuestiones relevantes, como por ejemplo: cuál es el partido político favorito para las próximas elecciones, cuáles son las películas más comentadas del último año, cuál es el restaurante mejor valorado en una zona determinada, etc. La extracción de información relevante de las redes sociales es un tema de interés, principalmente debido a la enorme cantidad de datos potencialmente disponibles. Sin embargo, las técnicas tradicionales de análisis de redes se están quedando obsoletas debido al crecimiento exponencial de las redes sociales, en términos de número de usuarios activos y de relaciones entre ellos. El Análisis de Redes Sociales (SNA por sus siglas en inglés, *Social Network Analysis*) se ha convertido en una de las tareas más populares y desafiantes en la ciencia de datos [28]. Uno de los problemas más abordados en las redes sociales

es el análisis de la relevancia de los usuarios en una determinada red social [29]. La relevancia de un usuario se relaciona comúnmente con el número de seguidores o amigos que tiene el usuario en una determinada red social. Sin embargo, este concepto puede ampliarse, ya que un usuario puede ser relevante no sólo si está conectado a un gran número de usuarios, sino también si está conectado a usuarios relevantes. Se han propuesto varias métricas para analizar la relevancia de un usuario en una red social, siendo PageRank una de las más utilizadas [30]. Además, es interesante saber qué usuarios serán los más relevantes antes de convertirse en influyentes [31]. Por último, en el campo de la analítica de marketing, existe un especial interés en generar el perfil de un usuario dado un conjunto de tuits escritos por dicho usuario [32].

La evaluación de la relevancia de un usuario se ha convertido en un problema más complejo que consiste en detectar usuarios específicos (a menudo denominados influenciadores) con ciertos atributos que pueden ser personales (credibilidad o entusiasmo) o relacionados con sus redes sociales (conectividad o centralidad).

Estos atributos les permiten influir en un gran número de usuarios, ya sea directa o indirectamente [33]. Otro tema importante relacionado con la influencia de las personas sobre otros usuarios es el análisis del sentimiento en las redes sociales. Este análisis se centra en descubrir lo que la gente piensa sobre un tema determinado mediante el análisis de la información que publican en las redes sociales. Se puede encontrar un estudio completo sobre las técnicas de análisis de sentimientos en [34].

Los problemas descritos anteriormente sólo se refieren a usuarios individuales. Sin embargo también hay algunos problemas relacionados con la estructura de la red, dedicados a encontrar atributos y propiedades específicas que pueden ayudar a inferir información adicional de la red social en su conjunto. En este contexto, la detección de comunidades surge como uno de los problemas más estudiados, que es el tema principal que concierne a esta Tesis.

## A.2 Hipótesis y objetivos

Una vez identificado el problema a resolver, el siguiente punto a tratar durante el desarrollo de un proyecto de investigación es la formulación de una hipótesis inicial. Esta hipótesis es una propuesta tentativa que busca formular una solución al problema planteado. La hipótesis representará un elemento fundamental en el proceso de investigación, ya que servirá de guía.

La hipótesis propuesta para el desarrollo de esta Tesis Doctoral se puede resumir en los siguientes términos: el Problema de Detección de Comunidades en

redes sociales es una tarea con interés práctico en diferentes disciplinas científicas, pero con un alto coste computacional. Por ello, es interesante desarrollar algoritmos que sean capaces de resolverlo de forma eficiente, obteniendo soluciones óptimas si es posible o, al menos, de alta calidad, en un tiempo razonable. Por este motivo, los algoritmos heurísticos cobran relevancia para resolver este tipo de problemas. En los últimos años se han propuesto heurísticas bioinspiradas para evitar los requerimientos computacionales de las implementaciones exactas. En el área de la computación bioinspirada, los enfoques evolutivos son los algoritmos más populares. Es importante destacar la revisión realizada por Pizzuti en [35] sobre técnicas de Computación Evolutiva (CE) para detectar comunidades en redes. Un trabajo interesante sobre CE es el publicado por Said et. al [36], donde los autores diseñaron un algoritmo genético basado en coeficientes de clustering capaz de detectar grupos cohesivos de grafos densos y también, comunidades en redes dispersas. Otro trabajo relevante es [37] que presenta un algoritmo genético que utiliza una función de cruce basada en el aprendizaje de conjuntos multiindividuales. El algoritmo se mejora con una estrategia de búsqueda local para acelerar su convergencia.

Otros algoritmos bioinspirados muy conocidos son los pertenecientes a la inteligencia de enjambre. En este nuevo grupo, los algoritmos más populares son la optimización por enjambre de partículas (PSO) y la optimización por colonia de hormigas (ACO). Estos dos algoritmos se inspiran en el comportamiento social de los pájaros dentro de una bandada, y en el comportamiento de las hormigas que buscan un camino desde el nido hasta la fuente de alimento, respectivamente. PSO se ha utilizado con éxito para CDP en [38], donde se utiliza un algoritmo PSO discreto para extraer las comunidades en redes sociales a gran escala mediante la optimización de la modularidad. En cuanto al algoritmo ACO, este algoritmo se ha utilizado para extraer comunidades de alta calidad en Ego Networks [39].

Sin embargo, esta familia de algoritmos tiene la desventaja de que no sacan partido de la información específica obtenida de la red. En esta Tesis, el desarrollo de algoritmos basados en metaheurísticas tradicionales, realizando un estudio en profundidad de las redes evaluadas, abre una nueva perspectiva de investigación futura para los problemas de detección de comunidades.

Los algoritmos heurísticos propuestos harán uso de técnicas metaheurísticas, que han demostrado ser procedimientos eficaces cuando se trata de problemas de optimización. En particular, las metaheurísticas basadas en poblaciones constituyen un subconjunto de este tipo de técnicas, que se caracterizan por tomar en consideración más de una solución simultáneamente y proporcionar mecanismos de combinación entre ellas. Técnicas como la búsqueda dispersa son un claro ejemplo de este tipo de metaheurísticas. Por otro lado, las llamadas meta-

heurísticas basadas en la trayectoria parten de una solución inicial y son capaces de generar una trayectoria en el espacio de soluciones. Ejemplos de este tipo de metaheurísticas son GRASP (Greedy Randomized Adaptive Search Procedure) o Path Relinking. El objetivo principal de la Tesis Doctoral es desarrollar algoritmos que resuelvan diversos problemas de optimización relacionados con el análisis de redes sociales y la detección de comunidades en las mismas, abordándolos desde un punto de vista heurístico, tanto desde la perspectiva monoobjetivo como multiobjetivo.

Para lograr el objetivo principal descrito anteriormente, es necesario cubrir los siguientes objetivos parciales:

- **Estudiar el estado del arte del problema.** Se debe realizar una revisión exhaustiva de la literatura para analizar tanto los algoritmos propuestos como las instancias a evaluar.

- **Diseñar y desarrollar un algoritmo metaheurístico para resolver el problema**. En este punto se desarrollan uno o varios algoritmos metaheurísticos, modelando el procedimiento constructivo inicial (normalmente una heurística), el método de mejora (normalmente una búsqueda local) y el marco metaheurístico que guía a la heurística.

- **Para validar el algoritmo heurístico.** Una vez diseñado el algoritmo propuesto, hay que implementarlo y validarlo. Este proceso de validación incluye comprobar si está generando soluciones factibles y, en caso de hacerlo, analizar la relevancia de cada etapa del algoritmo y su contribución a la solución final.

- **Comparar experimentalmente el algoritmo propuesto con los algoritmos del estado del arte.** En esta etapa, la propuesta se compara con los métodos del estado del arte utilizando un conjunto de datos de referencia. A través de esta experimentación se pueden estudiar los puntos fuertes y débiles del algoritmo desarrollado y su aportación a la literatura. En esta comparación, será necesario utilizar pruebas estadísticas para confirmar que el algoritmo propuesto es una contribución científica relevante para el área.

- **Someter los resultados parciales del trabajo de investigación a procesos de revisión por instituciones independientes.** Los resultados obtenidos durante todo el proceso se envían a congresos y revistas de alto impacto en el área de investigación para su posible publicación.

## A.3 Resultados

En esta sección se presenta una discusión de los resultados obtenidos para todos los problemas de optimización abordados en esta tesis. También se expone un breve resumen de las diferentes propuestas. Para cada problema resuelto, los resultados resumen el valor medio de la función objetivo que se tiene en cuenta cuando se aborda un problema de un solo objetivo y el promedio de las métricas multiobjetivo más utilizadas cuando se trata un problema multiobjetivo. También se presentan otras métricas que justifican la calidad de la propuesta (como el tiempo de CPU requerido por el algoritmo). En todas las tablas, los mejores resultados encontrados se destacan en negrita. En la sección A.3.1, se exponen los resultados de una primera aproximación a la detección de comunidades. La sección A.3.2 resume los resultados obtenidos al resolver el CDP clásico. Los resultados para el CDP multiobjetivo se muestran en la Sección A.3.3, mientras que los resultados para el CDP con solape se ilustran en la Sección A.3.4. Por último, en la sección A.3.5 se presentan los resultados obtenidos para el Problema de Detección de Comunidades Dinámicas multiobjetivo. Estos resultados se extraen de las publicaciones que se pueden encontrar en la Parte II de esta Tesis.

### A.3.1 Resultados en el problema del separador alfa

Como ya se ha dicho, un buen punto de partida para resolver los problemas de detección de comunidades es encontrar *cliques* en la red bajo estudio [46, 47, 48]. Otra buena aproximación es detectar cuáles son los nodos críticos cuya eliminación produce la división de la red en diferentes componentes conectados, generando comunidades iniciales. Este problema es $\mathcal{NP}$-difícil por sí mismo, denominado problema $\alpha$-separador, y se aborda en [134]. La función objetivo de este problema es encontrar el mínimo separador que divide la red dada en componentes conexas con un tamaño inferior a $\lceil \cdot n \rceil$ nodos, donde $n$ es el número total de nodos en la red. Los resultados sobre el problema del separador $\alpha$ se han publicado en una revista del JCR; concretamente, se ha publicado en Expert Systems, una revista con un factor de impacto de 2.587 y situada en el Q2 del JCR. Se pueden encontrar más detalles sobre la revista en el capítulo 7 de la parte II. En este trabajo se propone un algoritmo basado en *Greedy Randomized Adaptive Search Procedure* (GRASP) combinado con la metaheurística *Path Relinking*. En esta propuesta, se utiliza GRASP como metaheurística constructiva siguiendo dos enfoques diferentes: por un lado, un enfoque *Greedy Random*, donde el primer elemento que forma parte de una solución se selecciona al azar y la lista restringida de candidatos se construye siguiendo un criterio greedy. Por otro lado, un enfoque *Random Greedy*, donde el primer elemento que forma parte de la solución se selecciona de forma greedy,

mientras que la lista restringida de candidatos se construye de forma aleatoria. El criterio greedy utilizado es una métrica bien conocida y ampliamente utilizada en el contexto del Análisis de Redes Sociales: la cercanía o *closeness*. En la versión *Greedy Random*, el primer nodo añadido al separador se selecciona al azar, con el objetivo de diversificar el procedimiento de búsqueda. La lista de candidatos se construye utilizando la métrica de *closeness* para cada nodo, y la lista restringida de candidatos se construye con los nodos que tienen asociado un valor grande de esta función. A continuación, el siguiente vértice que se añade al separador se selecciona aleatoriamente de la lista restringida de candidatos. El enfoque *Random Greedy* funciona de forma similar, pero la fase greedy y la aleatoria se intercambian: el RCL se construye con un conjunto de elementos seleccionados aleatoriamente de la lista de candidatos y, a continuación, el siguiente elemento incluido en la solución se selecciona de forma greedy. En cuanto al tiempo de cálculo requerido por cada algoritmo, la versión *Random Greedy* es más rápida que la *Greedy Random*, ya que no requiere una evaluación de toda la lista de candidatos. Además, como se ha demostrado experimentalmente, proporciona mejores resultados en promedio que el enfoque *Greedy Random*.

En la fase de mejora, se define un procedimiento de búsqueda local para encontrar un óptimo local con respecto a una vecindad predefinida. La vecindad para esta búsqueda local es la definida por todas las soluciones que se pueden generar aplicando un movimiento en el que se eliminan dos vértices de la solución y se añade uno nuevo. Con respecto al orden en el que se explora la vecindad, se aplica un método *primera mejora*. Hay dos razones principales para esta decisión: la primera es que, normalmente, un enfoque *Primera Mejora* consume mucho menos tiempo que el *Mejor Mejora*, dado que el primero no recorre todo el vecindario, sino que sólo realiza el primer movimiento de mejora encontrado. La segunda es que, cuando se realiza un movimiento exitoso, implica necesariamente una mejora en la función objetivo, dado que se está resolviendo un problema de minimización, y la búsqueda local realiza movimientos en los que se eliminan dos nodos presentes en la solución y se sustituyen por uno solo. El método de búsqueda local propuesto explora aleatoriamente la vecindad, aumentando la diversificación de la búsqueda. Esta fase se detiene cuando no se encuentra ninguna mejora después de explorar toda la vecindad.

El procedimiento GRASP genera un conjunto de soluciones diversas y de alta calidad, y se construye un Conjunto de Élite con las más prometedoras. Una vez generado el Conjunto de Élite, se aplica el Path Relinking con el objetivo de explorar las trayectorias entre cada par de soluciones incluidas en él. Esto se conoce como diseño estático. En un diseño dinámico, el Conjunto de Élite se actualiza cada vez que se explora una trayectoria. En este problema, la trayectoria

entre dos soluciones se genera realizando un movimiento *Swap*. Este movimiento elimina un nodo que está presente en la solución inicial pero que no está incluido en la solución guía, y lo sustituye por uno presente en la solución guía y que aún no se ha añadido al separador inicial. Mediante estos intercambios, la solución inicial se aproxima a la solución guía, hasta que las dos soluciones acaban siendo exactamente iguales. En cada iteración, se genera una nueva solución intermedia. Esta solución intermedia será inviable con una alta probabilidad, por lo que debe ser reparada para que vuelva a ser factible. En este trabajo se realiza un proceso de reparación añadiendo aleatoriamente nodos al separador hasta alcanzar de nuevo la viabilidad, aumentando la diversificación de la búsqueda. Una vez realizada la reparación, el separador podría contener nodos redundantes (es decir, nodos que ya no son necesarios para tener una solución factible), por lo que se recorre la solución buscando los nodos redundantes que se eliminan.

Normalmente, para seleccionar el siguiente nodo que se elimina del separador, existen dos estrategias diferentes de Path Relinking. Por un lado, Random Path Relinking (RPR) genera la siguiente solución en la trayectoria seleccionando aleatoriamente uno de los movimientos *Swap* en la trayectoria. Por otro lado, Greedy Path Relinking (GPR) selecciona el mejor movimiento *Swap* disponible. En este trabajo se propone una tercera variante: *Greedy Randomized Path Relinking* (GRPR). En esta estrategia se mezclan las estrategias RPR y GPR, de forma que se generan todas las posibles soluciones intermedias (como ocurre en GPR), pero se selecciona una de las mejores soluciones en lugar de la mejor, de forma similar a como lo hace GRASP. Por último, se presenta una nueva estrategia de PR, denominada Exterior Path Relinking (EPR). Esta versión, propuesta originalmente en [135], trata de reducir la generación de trayectorias cortas generadas por el enfoque clásico, eliminando de la solución inicial elementos que están presentes en la solución guía. De esta forma, la solución inicial se va alejando de la guía en cada iteración. Esta estrategia termina cuando la solución inicial no tiene ningún componente común con la solución guía. Es importante destacar que las soluciones obtenidas por cualquiera de estos métodos no son necesariamente un óptimo local con respecto a la vecindad que se explora, por lo que a las soluciones obtenidas se les aplica el procedimiento de búsqueda local antes mencionado.

Los experimentos computacionales se han realizado sobre un conjunto de 50 instancias extraídas del modelo de Erd "os-Renyi. Estos grafos están construidos de forma que cada nuevo nodo insertado tiene la misma probabilidad de estar conectado a cualquier nodo existente en el grafo. La experimentación preliminar se encarga de analizar el rendimiento de los procedimientos GRASP propuestos y de seleccionar la mejor variante, y de hacer lo mismo con los diferentes procedimientos de búsqueda local y las versiones Path Relinking propuestas. La experimentación

final analiza la contribución de la propuesta comparándola con el método más avanzado encontrado en la literatura, un algoritmo basado en Random Walks (RW). La tabla A.1 muestra los resultados comparando la media del valor de la función objetivo, el tiempo de cálculo requerido, el porcentaje de desviación cuando no se alcanza la mejor solución y las veces que el algoritmo encuentra la mejor solución.

| Algoritmo | Avg. | Tiempo (s) | Dev. (%) | #Mejores |
|---|---|---|---|---|
| GRASP | 63.18 | **137.41** | 5.90 | 14 |
| GRASP+PR | **62.00** | 822.29 | **3.68** | **34** |
| RW | 71.78 | 1070.36 | 18.26 | 18 |

Table A.1: Comparación final entre GRASP, GRASP+PR, y el mejor método encontrado en el estado del arte.

La principal conclusión que se obtiene de estos resultados es que el enfoque seguido, basado en la alta diversificación, permite encontrar soluciones de alta calidad en un tiempo de computación bajo. En este trabajo, la mejor versión del algoritmo propuesto es la que combina el Exterior Path Relinking con la versión *Random Greedy* de GRASP con un valor $\alpha$ de 0,25. Este enfoque se compara con el algoritmo basado en *Random Walks*, mostrando claramente una superioridad en cuanto a los valores de la función objetivo y los tiempos de cálculo requeridos. Estos resultados están respaldados por una prueba estadística no paramétrica. En concreto, se ha realizado la prueba de rangos con signo de Wilcoxon por pares, con un valor $p$ resultante inferior a 0,001, lo que indica que existen diferencias estadísticamente significativas entre la propuesta y el algoritmo de última generación, con un nivel de significación del 95%. Más concretamente, la propuesta es capaz de alcanzar 34 veces las mejores soluciones para el conjunto de 50 instancias, teniendo un bajo valor de desviación respecto al mejor valor encontrado cuando no es capaz de encontrarlo (3,68% de desviación en promedio).

### A.3.2 Resultados del problema clásico de detección de comunidades

El clásico Problema de Detección de Comunidades (CDP) se estudia en [136] (Capítulo 3.3, Parte II) utilizando un enfoque GRASP. El algoritmo propuesto explota la versatilidad de diversificación que proporciona el GRASP para alcanzar soluciones de alta calidad. Para ello, en la fase constructiva de comprensión, se sigue un enfoque aglomerativo. El algoritmo comienza localizando cada nodo en una única comunidad conformada por él mismo. A continuación, se genera la

lista de candidatos con todas las comunidades presentes en la red. La RCL se construye utilizando la modularidad como función objetivo, y se ordena de forma decreciente respecto a este valor. Una vez construidas la lista de candidatos y la lista restringida de candidatos, se selecciona una comunidad al azar de la RCL. A continuación, el algoritmo busca la mejor comunidad para unirla con la primera. Para decidir cuál es la mejor comunidad para ser conjunta, se realiza una simulación de la fusión de la comunidad candidata con todas las demás. A continuación, se fusionan la comunidad candidata y la mejor encontrada. Si este movimiento da lugar a una mejora de la modularidad, la nueva comunidad se convierte en un nuevo candidato y se actualiza la solución actual. En caso contrario, la comunidad candidata se elimina de la lista de candidatos, dado que no puede fusionarse con otra comunidad sin empeorar la función objetivo. Cuando no hay más movimientos de fusión posibles, la fase de construcción termina.

En la fase de mejora local del algoritmo, se propone un procedimiento de búsqueda local. En este procedimiento, el vecindario que se explora está conformado por todas las soluciones que se pueden alcanzar realizando un movimiento que remueva un nodo de su comunidad actual y lo inserte en una nueva. Es importante destacar que este movimiento puede llevar a que una comunidad quede vacía si se eliminan todos sus nodos. Del mismo modo, podría generarse una nueva comunidad si este movimiento mejora la modularidad de la solución general. Esta situación hace que, tras la búsqueda local, una solución pueda contener un número diferente de comunidades (mayor o menor). Para decidir qué nodo va a ser eliminado de su correspondiente solución, se aplica un criterio heurístico. Concretamente, se evalúa el porcentaje de aristas intracomunitarias con respecto al número total de aristas del grafo para cada vértice. A continuación, el algoritmo selecciona el vértice con un valor más bajo de esta métrica, y lo traslada a la comunidad que maximiza la modularidad. La búsqueda local propuesta sigue un enfoque de primera mejora, reiniciando la búsqueda cuando se encuentra la primera mejora del vecindario bajo exploración, y deteniéndola cuando se encuentra una mejora.

Los experimentos computacionales se han realizado sobre un conjunto de 100 instancias extraídas del conjunto de datos SNAP de Twitter y del repositorio Network. La fase experimental se divide en dos fases diferentes. La primera está dedicada a afinar el valor $\alpha$ para el procedimiento GRASP. Este experimento se realiza sobre un subconjunto de 20 instancias e indica que la selección de un valor aleatorio de alfa en cada iteración del algoritmo da como resultado la mejor configuración para el algoritmo.

La experimentación final se ha llevado a cabo para comparar la propuesta con un conjunto de algoritmos clásicos bien conocidos en el ámbito de la detección

de comunidades. Concretamente, se ha comparado con los algoritmos Edge Betweenness (EB), Fast-Greedy (FG), Label Propagation (LB), Multi-level (ML), Walktrap (WT), InfoMap (IM) y Louvain (CL). La tabla A.2 muestra los valores medios de modularidad y conductancia de cada algoritmo en todo el conjunto de instancias. Es importante destacar que, en este trabajo, el valor reportado es el opuesto a la conductancia, evaluado como $1 - C_o(S, G)$, con el objetivo de tener una comparación directa con la métrica de modularidad.

| | Modularidad | | Conductancia | |
|---|---|---|---|---|
| | Avg. | #Best | Avg. | #Best |
| EB | 0.20176 | 0 | 0.03363 | 7 |
| FG | 0.29441 | 3 | 0.44062 | 17 |
| LP | 0.15170 | 2 | 0.43734 | 6 |
| ML | 0.28843 | 2 | 0.43433 | 19 |
| WT | 0.26663 | 2 | 0.25224 | 7 |
| IM | 0.20611 | 2 | 0.37829 | 16 |
| CL | 0.31181 | 33 | 0.48002 | 9 |
| *GRASPAGG* | **0.31331** | **78** | **0.49483** | **37** |

Table A.2: Comparación de las métricas consideradas sobre los algoritmos clásicos y la propuesta.

La primera conclusión que se extrae es que la propuesta, a pesar de ser la combinación de dos heurísticas sencillas, su potente diversificación y rápido cálculo permite obtener soluciones de alta calidad y muy competentes respecto a los algoritmos clásicos de detección de comunidades. Además, el equilibrio entre avaricia y aleatoriedad en el procedimiento constructivo, permite al procedimiento de búsqueda local realizar la fase de intensificación en una amplia región del espacio de soluciones, obteniendo buenos resultados gracias a la definición de vecindades basada en el problema.

Los resultados se apoyan en pruebas estadísticas no paramétricas. En particular, se realizaron el test de Friedman y el test de Wilcoxon. El test de Friedman clasificó los algoritmos comparados desde 1 (mejor algoritmo) hasta $n$. Los algoritmos de la propuesta y de Lovaina obtuvieron las dos primeras posiciones del ranking. Ambas pruebas estadísticas dieron como resultado un valor $p$ inferior a 0,00001, lo que confirma que existen diferencias estadísticamente significativas entre los algoritmos. Estos resultados se han publicado en la revista Electronics JCR, situada en el tercer cuartil (Q3) con un factor de impacto de 1.764.

### A.3.3 Resultados del problema de detección de comunidades multiobjetivo

En [137] se aborda el problema de detección de comunidades desde un punto de vista multiobjetivo. El texto completo se puede encontrar en el capítulo 9, parte II. En este trabajo se ha utilizado una metodología de Búsqueda de Vecinos Variables. En concreto, se ha implementado un algoritmo VNS básico (BVNS). Dado que el marco VNS fue diseñado originalmente para resolver problemas de optimización de un solo objetivo, debe ser adaptado para el escenario multiobjetivo [138]. En este trabajo, esta adaptación se realiza considerando el frente no dominado de soluciones en lugar de uno solo como la solución que debe devolver el algoritmo. El algoritmo BVNS se ejecuta hasta que se explora la vecindad máxima establecida como parámetro. En cada iteración se ejecutan los procedimientos de *shake*, mejora y cambio de vecindad, actualizando el conjunto no dominado de soluciones.

Dado que BVNS requiere de una solución inicial para funcionar (es decir, un frente poblado de soluciones no dominadas), debe ser construido de alguna manera. Para este problema, este conjunto inicial se construye siguiendo un procedimiento GRASP. En este trabajo, sólo se tiene en cuenta la fase constructiva del GRASP. En esta fase, la solución inicial está conformada por $n$ comunidades, donde $n$ es el número de nodos del grafo. La lista de candidatos se construye asignando un valor de la función greedy a cada comunidad presente en la solución. A continuación, las comunidades se fusionan por parejas, seleccionando aleatoriamente un elemento de la RCL y fusionándolo con la mejor comunidad existente en la solución actual. Para seleccionar la mejor comunidad con la que unirse, la función codiciosa seleccionada corresponde a la relación entre las aristas intracomunitarias y las aristas comunitarias (intraclúster e interclúster) que habría si se fusionaran dos comunidades. Esta metodología permite generar un conjunto poblado de soluciones no dominadas. Es importante señalar que no todas las soluciones construidas se incluyen en el frente de referencia, sino sólo las que no están dominadas.

En cuanto al procedimiento de *shake*, en la propuesta consiste en un movimiento en el que se retira un único nodo de su comunidad actual, insertándolo en otra seleccionada al azar. Siguiendo este procedimiento, las soluciones obtenidas serán probablemente peores que la solución original en términos de calidad, pero es importante destacar que el objetivo principal en los procedimientos de *shake* es escapar de los óptimos locales. En esta fase del algoritmo, lo que se busca es la diversificación sobre la intensificación. Dada la naturaleza del problema, no es necesario comprobar la viabilidad de la solución resultante, ya que está garantizado que un único nodo será asignado a al menos y sólo una comunidad cuando el procedimiento de *shake* finalice.

Con el objetivo de mejorar la calidad de las soluciones perturbadas y, por tanto, centrándose en la intensificación de las soluciones (es decir, intentando alcanzar un óptimo local), se aplica un procedimiento de búsqueda local a cualquier solución perturbada. En este trabajo, el procedimiento de mejora toma como entrada un conjunto de soluciones perturbadas y devuelve un frente no dominado con todos los óptimos locales alcanzados a partir del conjunto perturbado. En este trabajo se estudian dos métodos de mejora local diferentes, ambos siguiendo un enfoque de primera mejora. Por un lado, el primer método propuesto mejora cada función objetivo evaluada de forma independiente. Por otro lado, el segundo trata de optimizar ambas estrategias simultáneamente, considerando alternativamente una de ellas en cada iteración del procedimiento. De nuevo, cualquier solución mejorada se intenta incluir en el frente no dominado. El movimiento que define ambos procedimientos de búsqueda local es el mismo: se intenta incluir un nodo perteneciente a una comunidad en otra diferente. A continuación, se intenta añadir la nueva solución al frente de referencia. Si se puede incluir, significa que domina al menos una solución que está presente en el frente, por lo que se ha encontrado una mejora. El procedimiento finaliza cuando el frente de referencia no se actualiza tras una ejecución completa de la búsqueda local.

Una vez generado un conjunto no dominado de alta calidad, se ejecuta el método de cambio de vecindad. En este trabajo se ha adaptado este procedimiento para adecuarlo al escenario multiobjetivo. La principal adaptación realizada es la modificación del concepto de mejora. En este contexto, se considera una mejora cuando se actualiza el frente de soluciones no dominadas. Esta actualización indica que se ha encontrado una solución que domina a una presente en el frente. Por lo tanto, el algoritmo reinicia la búsqueda desde el vecindario inicial si se ha realizado una mejora. En caso contrario, se explora el siguiente vecindario, hasta que se alcanza el vecindario máximo, establecido como parámetro de entrada.

En cuanto a los experimentos computacionales, se ha utilizado un conjunto de 52 redes sintéticas y 12 redes del mundo real. En este trabajo se han evaluado dos tipos diferentes de métricas para comprobar la robustez del algoritmo. Por un lado, se han estudiado métricas multiobjetivo bien conocidas. Por otro lado, se ha tenido en cuenta la información mutua normalizada (NMI) y las métricas basadas en el contexto de modularidad. La experimentación se ha dividido en una fase preliminar y otra final. La primera se dedica a ajustar los parámetros del algoritmo GRASP, la mejor configuración del método de mejora local y el parámetro $k_{max}$ para el BVNS. La segunda se realiza para comparar la propuesta con el mejor algoritmo encontrado en el estado del arte. La tabla A.3 muestra la comparación respecto a la métrica multiobjetivo. En esta tabla se puede observar que la calidad y el tiempo de cálculo de la propuesta supera al mejor método

encontrado en la literatura. En la Tabla A.4 se muestran los resultados respecto a las métricas basadas en el contexto.

| Algorithm | *CV* | *HV* | *EPS* | *IGD+* | $T(s)$ |
|---|---|---|---|---|---|
| MOBVNS | **0.07** | **0.14** | **0.86** | 0.22 | **214.64** |
| LMOEA | 0.36 | 0.02 | 0.27 | **0.21** | 1800.00 |

Table A.3: Comparación del frente de referencia obtenido con la mejor configuración para MOBVNS y el LMOEA propuesto por [1]. Los mejores resultados están resaltados en negrita.

| | LMOEA | | | |
|---|---|---|---|---|
| Instancia | Avg. NMI | Avg. Tiempo (s) | Mejor NMI | Mejor Tiempo (s) |
| dolphin | 0.05 | 1800 | 0.069 | 1800 |
| footbal | 0.02 | 1800 | 0.033 | 1800 |
| karate | 0.1 | 1800 | 0.1 | 1800 |
| | MOBVNS | | | |
| Instancia | Avg. NMI | Avg. Tiempo (s) | Mejor NMI | Mejor Tiempo (s) |
| dolphin | **0.751** | **0.41** | **0.77** | **0.11** |
| footbal | **0.864** | **1.8** | **0.877** | **0.27** |
| karate | **0.439** | **0.07** | **0.439** | **0.03** |

Table A.4: Resumen de los resultados de la métrica NMI obtenidos por el MOBVNS y el LMOEA propuestos al resolver las instancias del mundo real.

Se puede concluir que la combinación del procedimiento constructivo GRASP con el marco VNS es un mecanismo potente para resolver problemas multiobjetivo. Es importante señalar que el marco VNS requiere ser adaptado para gestionar con éxito el escenario multiobjetivo. Más concretamente, respecto a las métricas multiobjetivo clásicas, se observa que la propuesta MOBVNS es mejor en todas ellas, excepto en *IGD+*, donde es ligeramente superior a LMOEA. En cuanto al tiempo medio de cálculo, la propuesta tarda aproximadamente nueve veces menos que el algoritmo anterior. En cuanto al rendimiento en tiempo real, la propuesta es capaz de alcanzar valores más altos de NMI en cuatro órdenes de magnitud menos de tiempo. Estos resultados se han publicado en Applied Soft Computing, una revista Q1 en JCR con un factor de impacto de 6.725.

### A.3.4 Resultados del problema de detección de comunidades con solape

Este trabajo está actualmente en revisión en Journal of Heuristics. Aunque los resultados están todavía en proceso de publicación, es interesante discutir los re-

sultados obtenidos en la investigación, así como las conclusiones extraídas. En este trabajo se propone una metaheurística Iterated Greedy para resolver el CDP en su variante de solapamiento. Para generar la solución inicial del marco, se ha aplicado una metaheurística GRASP. Para la fase de construcción, se genera una solución inicial sin comunidades. A continuación, se construye la lista de candidatos incluyendo todos los nodos de la red, y el algoritmo se ejecuta hasta que todos los nodos hayan sido asignados a, al menos, una comunidad, es decir, hasta que la CL quede vacía. Para dar una puntuación a los candidatos y construir la lista restringida de candidatos, se utiliza una función codiciosa. Más concretamente, se utiliza la métrica PageRank. Esta métrica se propuso originalmente con el objetivo de evaluar la importancia de una página web en Internet, basándose en los enlaces que hacen referencia a ella en toda la red. Esta idea puede aplicarse en el CDP, dado que un nodo que está conectado a muchos otros es un nodo relevante en la red y, previsiblemente, será un buen candidato para empezar a construir una comunidad. A continuación, se selecciona un candidato al azar de la RCL y se añade a la comunidad en construcción. Para decidir qué nodos se van a añadir a la misma comunidad, se aplica un algoritmo de función de pertenencia dinámica [139]. Este método es un buen enfoque para el OCDP, dado que un nodo puede ser incluido en diferentes comunidades si satisface el siguiente criterio de pertenencia: un nodo es incluido en una comunidad si mejora la relación entre las aristas intracomunitarias e intercomunitarias de la comunidad titular. Los nodos que se incluyen en la comunidad en construcción se eliminan de la CL para evitar que se consideren puntos de partida de una nueva comunidad.

Una vez construida la solución, se aplica la fase de mejora de GRASP. En este trabajo, el operador de movimiento que genera nuevas soluciones (que conforman el vecindario en exploración) consiste en asignar un nodo a una nueva comunidad o eliminarlo de su comunidad actualmente asignada. Un nodo se moverá dependiendo de si la resta entre el número de aristas hacia una comunidad distinta a la suya dividido por su grado y el número de aristas hacia su comunidad dividido por su grado es mayor que un umbral definido como parámetro del algoritmo. Si es así, el nodo se retira de su comunidad actual y se traslada a la nueva. Si no, el nodo se añade a la nueva comunidad, produciéndose la situación de solapamiento. En esta propuesta, se sigue una primera estrategia de mejora con el objetivo de reducir el tiempo de cálculo del algoritmo. El procedimiento de búsqueda local termina cuando todas las soluciones pertenecientes a la vecindad actual son peores que la actual en términos de la función objetivo, es decir, cuando se encuentra un óptimo local.

Una vez que se ha construido una solución inicial mediante GRASP, se aplica el marco metaheurístico Iterated Greedy. En la fase de destrucción, un porcentaje

de nodos de la solución son desasignados de las comunidades a las que pertenecen. Este porcentaje se define experimentalmente. Para decidir qué nodos deben ser eliminados de la solución, se han seguido dos estrategias diferentes. Por un lado, se realiza una destrucción aleatoria. En esta versión de la destrucción, los nodos que se eliminarán de la solución actual se seleccionan aleatoriamente. Por otro lado, se sigue una estrategia codiciosa. En concreto, se eliminan de sus comunidades los nodos con una mayor proporción de aristas interclúster respecto a su grado. Ambas resultan en una solución no factible, dado que algunos nodos no están asignados a ninguna comunidad cuando el procedimiento termina.

Para recuperar la viabilidad de la solución destruida, se ejecuta la fase de reconstrucción. Para ello, los nodos no asignados deben ser reasignados a, al menos, una comunidad de la red. En esta fase del algoritmo, los nodos sólo se asignarán a una única comunidad. La decisión de añadir un nodo a más de una comunidad es responsabilidad del procedimiento de búsqueda local, que se aplicará después de la reconstrucción. Para asignar los nodos a una comunidad, de nuevo, se han seguido dos estrategias diferentes: reasignar los nodos a una comunidad aleatoria y la estrategia opuesta a la seguida en la fase de destrucción, seleccionar la comunidad que maximiza el ratio de aristas intra-clúster con respecto a su grado como la comunidad más adecuada para un nodo. Por último, tras la fase de reconstrucción se aplica el mismo procedimiento de búsqueda local aplicado en la fase de mejora de GRASP, con el objetivo de encontrar un óptimo local en la región del espacio de soluciones que se está explorando.

Se han realizado experimentos computacionales para evaluar la calidad de la propuesta. Se ha utilizado un conjunto de 68 instancias. La experimentación preliminar se dedica a ajustar todos los parámetros del algoritmo: el valor $\alpha$ de GRASP y el porcentaje de destrucción para Iterated Greedy. Además, esta experimentación sirve para seleccionar la mejor configuración de las estrategias de construcción y destrucción. Como resultado de estos experimentos, se obtiene la mejor configuración del algoritmo. En la experimentación final, el algoritmo se compara con el mejor algoritmo encontrado en la literatura. Para evaluar la calidad de las soluciones se ha utilizado una versión adaptada de la métrica de modularidad al escenario de solapamiento. La tabla A.5 muestra la superioridad de la propuesta en un conjunto de instancias de diferente tamaño ($n$ representa el número de nodos presentes en la red).

Estos resultados muestran que la propuesta es, en promedio, una mejor opción para las redes evaluadas. Aunque el tiempo de cálculo requerido por el algoritmo EADP es menor que el requerido por la propuesta, las soluciones encontradas por Iterated Greedy superan al algoritmo anterior. Además, cuando Iterated Greedy no es capaz de encontrar las mejores soluciones, tiene un bajo

| | Iterated Greedy | | | | EADP | | | |
|---|---|---|---|---|---|---|---|---|
| | Avg. | Dev (%) | Tiempo (s) | #Mejores | Avg. | Dev (%) | Tiempo (s) | #Mejores |
| $0 \leq n < 2500$ | **0.319** | **3.399** | 5.501 | **13** | 0.216 | 34.855 | **0.507** | 2 |
| $2500 \leq n < 5000$ | **0.383** | **0.000** | 27.736 | **15** | 0.238 | 37.061 | **3.243** | 0 |
| $5000 \leq n < 7500$ | **0.377** | **0.000** | 48.293 | **18** | 0.236 | 37.102 | **10.781** | 0 |
| $7500 \leq n \leq 10000$ | **0.377** | **0.000** | 87.124 | **9** | 0.234 | 37.171 | **36.943** | 0 |
| *Average* | **0.364** | **0.850** | 42.164 | **55** | 0.231 | 36.547 | **12.869** | 2 |

Table A.5: Comparación de Iterated Greedy (IG) y EADP configurado como se indica en [2].

porcentaje de desviación en promedio (3,399% para el conjunto de instancias en las que no es capaz de encontrar la mejor solución, 0,850% en promedio para todo el conjunto de instancias). El algoritmo encuentra la mejor solución un total de 55 instancias, mientras que el método de vanguardia alcanza la mejor solución 2 veces. En cuanto a los tiempos de cálculo, Iterated Greedy sólo tarda una media de 30 segundos más que EADP, por lo que la mejora en la calidad de las soluciones justifica el interés de la propuesta.

Los resultados derivados de esta investigación se encuentran actualmente en revisión en Journal of Heuristics, una revista del Q2 JCR con un factor de impacto de 2.247.

### A.3.5 Resultados sobre el problema de detección de comunidades dinámicas multiobjetivo

El artículo que resume la investigación relativa a este problema está en segunda revisión en la revista Expert Systems With Applications. No obstante, es interesante exponer las estrategias seguidas para resolverlo. En este trabajo se adapta una metaheurística basada en poblaciones para resolver el problema multiobjetivo de Detección Dinámica de Comunidades: la Búsqueda por Dispersión. Este marco se ha modificado ligeramente con el fin de afrontar la naturaleza multiobjetivo del problema que se resuelve. Las principales modificaciones se han realizado en el método de mejora, el método de actualización *RefSet*, y el método de generación de subconjuntos. En el método de mejora, se mejora una solución de dos formas diferentes, cada una de ellas utilizando como función objetivo una de las métricas de optimización del problema: *ICS* (Inverted Community Score) y *AVG_ODF*. El procedimiento de actualización del *RefSet* se modifica de forma que se mantienen dos *RefSets*, uno considerando el objetivo *ICS* y el otro considerando *AVG_ODF*. El método de generación de subconjuntos se adapta para reducir el número de pares que se combinan, dividiendo el *RefSet* en cuatro subconjuntos diferentes: el que contiene soluciones de alta calidad con respecto al *ICS*, el que contiene soluciones de alta calidad con respecto al *AVG_ODF*, y los dos que contienen solu-

ciones diferentes con respecto a cada una de las funciones objetivo. Es importante destacar que todas las soluciones generadas se intentan incluir en el conjunto de soluciones no dominadas.

Con el objetivo de poblar el *RefSet*, se generan soluciones de alta calidad y soluciones diversas que se incluyen en el conjunto. En el contexto de MODCDP, la distancia entre dos soluciones se calcula como la suma de los nodos que se encuentran en diferentes comunidades en las soluciones comparadas. Esta es la métrica utilizada para generar los subconjuntos diversos *RefSet*.

En el método de generación diversa, se ha seguido un enfoque aleatorio codicioso. Partiendo de la idea de que encontrar una buena estructura de comunidad es una tarea computacionalmente exigente, se aplica un algoritmo basado en el trabajo de McAllister [140]. Más concretamente, se aplica un algoritmo modificado de búsqueda Breadth-First Search. La modificación tiene en cuenta el valor de la función McAllister asociado a cada nodo para decidir cuándo debe detenerse la construcción de la comunidad actual y construir una nueva. La función McAllister se utiliza porque da prioridad para ser conjunta a la comunidad actual a aquellos nodos que tienen más vecinos etiquetados, es decir, más vecinos añadidos a la comunidad en construcción. Este procedimiento termina cuando todos los nodos han sido asignados a una comunidad.

Una vez que se ha construido un conjunto de soluciones diversas, se ejecuta un procedimiento de mejora con el objetivo de encontrar un óptimo local con respecto a una determinada vecindad. Para alcanzar este objetivo se define un procedimiento de búsqueda local. El vecindario que se explorará está compuesto por todas las posibles soluciones que se pueden alcanzar cambiando un nodo de su comunidad original a otra diferente. Para decidir si un nodo es un candidato prometedor para realizar el movimiento, se evalúa el número de aristas intracomunitarias e intercomunitarias. Si un nodo tiene más aristas intercomunitarias que intercomunitarias, significa que podría ser asignado a una comunidad mejor. La comunidad de destino se selecciona evaluando el número de nodos adyacentes que los nodos eliminados tienen en la comunidad evaluada. Realizando este movimiento, se mejorará la calidad de la estructura de la comunidad, ya que un nodo siempre será reasignado a una comunidad con más aristas intraclúster de las que tenía en su asignación original. Se ha seguido una estrategia de mejora de la calidad para recorrer dicho vecindario, ya que la forma eficiente en que se ha implementado permite una exploración completa del mismo sin requerir un tiempo computacional elevado.

En el método de combinación, se aplica un enfoque de Path Relinking a cada par de soluciones derivadas del *RefSet*. Más concretamente, una solución

perteneciente al subconjunto de alta calidad de *RefSet* para una función objetivo se combina con otra que pertenece al subconjunto diverso de *RefSet* para la misma función objetivo. Este proceso se aplica para ambas funciones objetivo. Para crear un camino entre una solución inicial y una solución guía, se añaden a la solución inicial elementos que están incluidos en la solución guía y que no están presentes en la solución inicial. En este trabajo se sigue un enfoque de Reenlace Aleatorio de Caminos, es decir, se selecciona aleatoriamente el nodo cuya comunidad asignada será cambiada.

Es importante señalar que existen dos enfoques diferentes para resolver la variante dinámica del CDP. Por un lado, cada solución puede generarse desde cero para cada instantánea de la red. Por otro lado, la solución generada para una instantánea puede utilizarse como punto de partida para la siguiente. En este trabajo se han probado ambos enfoques, emergiendo la segunda estrategia como la mejor.

En la experimentación computacional se han estudiado tanto las métricas multiobjetivo clásicas como las basadas en el contexto. En concreto, se han seleccionado la Cobertura, el Hipervolumen y la Distancia Generacional Invertida + como métricas multiobjetivo y la modularidad como métrica basada en el contexto para evaluar la calidad de las soluciones. Se ha utilizado un conjunto de 69 instancias sintéticas y del mundo real para probar el rendimiento de los algoritmos. La fase experimental se ha dividido en dos conjuntos diferentes de experimentos. El primero está dedicado a seleccionar la mejor estrategia (empezar de cero en cada instantánea o aprovechar la solución generada para la instantánea anterior), y estudiar la contribución de cada parte del algoritmo a la solución final. En los experimentos finales, la propuesta se compara con el mejor método encontrado en la literatura. Analizando los resultados, la propuesta demostró ser mejor tanto en las redes sintéticas como en las del mundo real. La tabla A.6 muestra la comparación entre ambos algoritmos en cuanto a las métricas multiobjetivo en redes sintéticas. Como se muestra, la propuesta Scatter Search demuestra una superioridad con respecto a estas métricas. Más concretamente, el algoritmo Scatter Search es capaz de alcanzar los valores más bajos posibles para las métricas CV e IGD+ en todas las instancias sintéticas. En cuanto a la VH, se alcanzan valores más altos que los del algoritmo previo *Immigrants* para todas las instantáneas en todas las instancias.

Lo mismo ocurre para las instancias del mundo real evaluadas (Tabla A.7). Como se puede observar, la propuesta basada en Scatter Search obtiene los mejores resultados en cuanto a la métrica CV y tiene una diferencia significativa en promedio para las otras dos métricas multiobjetivo estudiadas.

| Snapshot | Scatter Search Repairing | | | Immigrants MOGA | | |
|---|---|---|---|---|---|---|
| | AVG. CV | AVG. HV | AVG. IGD+ | AVG. CV | AVG. HV | AVG. IGD+ |
| Snapshot 0 | **0.00** | **0.62** | **0.00** | 1.00 | 0.08 | 1.60 |
| Snapshot 1 | **0.00** | **0.54** | **0.00** | 1.00 | 0.14 | 1.17 |
| Snapshot 2 | **0.00** | **0.63** | **0.00** | 1.00 | 0.11 | 0.23 |
| Snapshot 3 | **0.00** | **0.63** | **0.00** | 1.00 | 0.11 | 0.25 |
| Snapshot 4 | **0.00** | **0.61** | **0.00** | 1.00 | 0.12 | 0.28 |
| Snapshot 5 | **0.00** | **0.59** | **0.00** | 1.00 | 0.18 | 0.29 |
| Snapshot 6 | **0.00** | **0.53** | **0.00** | 1.00 | 0.14 | 1.13 |
| Snapshot 7 | **0.00** | **0.55** | **0.00** | 1.00 | 0.14 | 8.65 |
| Snapshot 8 | **0.00** | **0.49** | **0.00** | 1.00 | 0.17 | 0.39 |
| Snapshot 9 | **0.00** | **0.49** | **0.00** | 1.00 | 0.17 | 3.23 |

Table A.6: Tabla comparativa de las métricas multiobjetivo obtenidas con los algoritmos Scatter Search e Immigrants MOGA en redes sintéticas.

| Dataset | Scatter Search Repairing | | | Immigrants MOGA | | |
|---|---|---|---|---|---|---|
| | AVG. CV | AVG. HV | AVG. IGD+ | AVG. CV | AVG. HV | AVG. IGD+ |
| Travian Market | **0.00** | **0.70** | **0.11** | 0.84 | 0.15 | 0.61 |
| Travian Messages | **0.00** | **0.62** | **0.27** | 0.33 | 0.13 | 0.31 |

Table A.7: Tabla comparativa de las métricas multiobjetivo obtenidas con los algoritmos Scatter Search e Immigrants MOGA en redes del mundo real.

En cuanto a las métricas basadas en el contexto, la superioridad de la propuesta también se puede confirmar mirando los valores de modularidad reportados en la Tabla A.8 y la Tabla A.9. Más concretamente, el algoritmo Scatter Search obtiene una media de valor de modularidad un orden de magnitud superior a la propuesta de MOGA en todas las instantáneas de instancias sintéticas. El mismo comportamiento se observa en las instancias del mundo real que se han utilizado en la fase experimental.

| Snapshot | Scatter Search Repairing | Immigrants MOGA |
|---|---|---|
| | AVG. Mod | AVG. Mod |
| Snapshot 0 | **0.3485** | 0.0242 |
| Snapshot 1 | **0.4879** | 0.0177 |
| Snapshot 2 | **0.5313** | 0.0167 |
| Snapshot 3 | **0.5328** | 0.0191 |
| Snapshot 4 | **0.5354** | 0.0168 |
| Snapshot 5 | **0.5349** | 0.0136 |
| Snapshot 6 | **0.5223** | 0.0116 |
| Snapshot 7 | **0.5131** | 0.0045 |
| Snapshot 8 | **0.4449** | 0.0090 |
| Snapshot 9 | **0.3543** | 0.0678 |

Table A.8: Tabla que compara los valores medios de modularidad obtenidos con los algoritmos Scatter Search e Immigrants MOGA en redes sintéticas.

| Dataset | Scatter Search Repairing | Immigrants MOGA |
|---|---|---|
| | AVG. Mod | AVG. Mod |
| Travian Market | **0.0714** | 0.0065 |
| Travian Messages | **0.2179** | 0.0899 |

Table A.9: Tabla que compara los valores medios de modularidad obtenidos con los algoritmos Scatter Search e Immigrants MOGA en redes del mundo real.

## A.4 Conclusiones

En este capítulo se presentan las conclusiones derivadas de la investigación realizada. Las conclusiones para cada variante de los problemas abordados se dividen en secciones. Además, se incluye una sección de trabajos futuros.

### A.4.1 Conclusiones sobre el problema del *alpha-separator*

Se ha estudiado el problema del *alpha-separator* como punto de partida para los problemas de detección de comunidades. Se ha propuesto un algoritmo basado en GRASP para encontrar nodos críticos en redes, siendo nodos críticos aquellos cuya eliminación implica la división de la red en $n$ componentes conexas con un tamaño determinado (dado como parámetro de la instancia). En este trabajo se ha demostrado el potencial de GRASP acoplado a una estrategia de combinación de Path Relinking. Especialmente, en lo que se refiere al tiempo de cómputo requerido por el algoritmo para alcanzar soluciones de alta calidad, la propuesta emergió como el mejor método en el estado del arte.

En concreto, la principal aportación de este trabajo radica en la fase experimental. Se han propuesto dos variantes diferentes de GRASP que utilizan la métrica de *closeness* como función voraz: *RandomGreedy* y *GreedyRandom*, y el experimento mostró que la estrategia *RandomGreedy* es más adecuada para este problema específico. Además, se han propuesto cuatro variantes diferentes de *Path Relinking*: *Greedy Randomized Path Relinking*, *Random Path Relinking*, *Greedy Path Relinking* y *Exterior Path Relinking*. Este último es el enfoque más novedoso, y también el mejor para resolver el problema del *alpha-separator*.

Un artículo derivado de este trabajo ha sido publicado en una revista JCR Q2, con un factor de impacto de 2.587 (JCR 2020): *Finding weaknesses in networks using greedy randomized adaptive search procedure and path relinking* [134], incluido en el capítulo 7 de la parte II.

### A.4.2 Conclusiones sobre el problema clásico de detección de comunidades

El objetivo principal del estudio del Problema de Detección de Comunidades (CDP) era desarrollar un algoritmo rápido que demostrara un mejor rendimiento que los algoritmos clásicos para la detección de comunidades en redes. Este objetivo se logró con una metodología GRASP, utilizando la conocida métrica de modularidad como función objetivo. Esta metaheurística ha demostrado ser un buen punto de partida para la resolución de problemas de detección de comunidades, dada su capacidad para equilibrar la diversificación en la fase de construcción y la intensificación en la fase de mejora, proporcionando soluciones de alta calidad en bajos tiempos de cómputo.

En este trabajo se han propuesto dos estrategias heurísticas: el procedimiento constructivo aglomerativo, que equilibra la voracidad y la aleatoriedad de la búsqueda, y el procedimiento de mejora, en el que se ha explotado con éxito

una vecindad basada en el problema para alcanzar los óptimos locales.

La fase experimental permite ajustar correctamente el valor $\alpha$ de GRASP, y evitar sobreajustar el algoritmo seleccionando un subconjunto del conjunto de instancias. Además, la fase experimental final demuestra que el objetivo de la investigación se ha cumplido, comparando la propuesta con siete conocidos algoritmos de detección de comunidades.

La investigación sobre el CDP se presenta en el trabajo denominado *On the Analysis of the Influence of the Evaluation Metric in Community Detection over Social Networks* [136], incluido en el capítulo 8 de la parte II. En concreto, este trabajo está publicado en la revista *Electronics*, un JCR Q3 con un factor de impacto de 1,764 (JCR 2018).

### A.4.3 Conclusiones sobre el problema de detección de comunidades multiobjetivo

En este trabajo se presenta un nuevo método metaheurístico basado en VNS para la detección de comunidades. Para resolverlo, se ha adaptado el VNS al escenario multiobjetivo, considerando un conjunto de soluciones no dominadas como solución completa para el algoritmo. Para generar el conjunto inicial de soluciones no dominadas, se ha aplicado una metodología GRASP. Esta metodología ha demostrado ser una técnica fiable y rápida para producir soluciones iniciales en el contexto del CDP. El uso de GRASP permite al algoritmo VNS iniciar la búsqueda desde una región prometedora del espacio de soluciones.

En este trabajo se han definido como función objetivo el *Ratio Cut* y la *Negative Ratio Association*, tratando de minimizar ambos simultáneamente. La evaluación de los frentes obtenidos se ha realizado utilizando métricas multiobjetivo clásicas, como Cobertura, Hipervolumen, Indicador $\epsilon$ y Distancia Generacional Invertida +.

La experimentación ha demostrado que la combinación de GRASP con VNS da lugar a soluciones de alta calidad en un contexto multiobjetivo, y más concretamente en el MOCDP. La eficiente implementación del algoritmo y la calidad de la heurística aplicada permiten a la propuesta superar los trabajos anteriores.

La investigación sobre el MOCDP se expone en el trabajo denominado *A fast variable neighborhood search approach for multi-objective community detection* [137], incluido en el capítulo 9 de la parte II. Más concretamente, se publica en *Applied Soft Computing*, una revista del Q1 JCR con un factor de impacto de 8,623 (JCR 2021).

### A.4.4 Conclusiones sobre el problema de detección de comunidades con solape

En este trabajo se propone un nuevo método metaheurístico para el problema de detección de comunidades con solape. En concreto, la propuesta hibridiza las metaheurísticas GRASP e *Iterated Greedy*. El algoritmo hace uso de una versión modificada de la clásica métrica de modularidad adaptada al escenario de solapamiento. Una de las principales aportaciones de este trabajo es el uso de un operador de movimiento inteligente definido para el procedimiento de búsqueda local. Éste permite mejorar la calidad de las soluciones generadas por el procedimiento GRASP, que hace uso de la métrica *PageRank* como función voraz. Mediante este procedimiento de búsqueda local se cubre el escenario de solapamiento, generando soluciones que permiten tener nodos asignados a más de una comunidad simultáneamente.

Otra contribución es el desarrollo de dos estrategias diferentes para las fases de destrucción y construcción de *Iterated Greedy*, respectivamente, y su combinación. Se ha propuesto una estrategia aleatoria y otra voraz para la fase de construcción, así como para la fase de reconstrucción. La sección experimental permitió seleccionar la mejor versión del procedimiento constructivo GRASP y la mejor configuración para la búsqueda local. También se ha establecido la mejor combinación de estrategias de las fases de construcción y destrucción. Los resultados muestran que la propuesta proporciona soluciones de mayor calidad que el método de vanguardia, aunque requiere un tiempo de cálculo ligeramente superior.

Este trabajo ha sido enviado para su revisión a *Journal of Heuristics*, revista de impacto indexada en el JCR (Journal Citation Reports), situada en el Q2 con un factor de impacto de 2,247.

### A.4.5 Conclusiones sobre el problema multiobjetivo de detección de comunidades dinámicas

En este trabajo se ha adaptado la metaheurística *Scatter Search* para que funcione en el contexto multiobjetivo. Además, la heurística voraz aleatorizada desarrollada permite generar soluciones iniciales para el marco de *Scatter Search* de forma rápida, combinando el potencial del conocido algoritmo *Breadth-First Search* y el cálculo rápido de la función voraz de McAllister. Utilizando esta combinación, y modificando el esquema de *Scatter Search* de forma que el conjunto de referencia tenga en cuenta todos los objetivos que se están optimizando, se alcanzan soluciones de alta calidad en tiempos computacionales bajos. Además, el uso de *Path Relinking* en su versión aleatoria para realizar el método de combinación de la metaheurística *Scatter Search* proporciona una mayor diversificación. Este

mecanismo permite poblar el conjunto no dominado de soluciones, lo que deriva en mejores resultados respecto a las métricas multiobjetivo clásicas y las métricas basadas en el contexto estudiadas en este trabajo.

En la fase experimental se demuestra la contribución de cada parte del algoritmo. Además, la propuesta reporta mejores soluciones en términos de calidad que los mejores trabajos anteriores encontrados en la literatura, dando la oportunidad a investigadores y profesionales de diferentes áreas de conocimiento de obtener soluciones con información útil y precisa, incluso cuando la red está en constante evolución.

Este trabajo está en progreso y será enviado a una revista indexada en JCR.

# Bibliography

[1] F. Cheng, T. Cui, Y. Su, Y. Niu, and X. Zhang, "A local information based multi-objective evolutionary algorithm for community detection in complex networks," *Applied Soft Computing*, vol. 69, pp. 357 – 367, 2018.

[2] M. Xu, Y. Li, R. Li, F. Zou, and X. Gu, "EADP: An extended adaptive density peaks clustering for overlapping community detection in social networks," *Neurocomputing*, vol. 337, pp. 287–302, 2019.

[3] C. A. Floudas and P. M. Pardalos, *Encyclopedia of optimization*. Springer Science & Business Media, 2008.

[4] K. Lange, *Optimization*, vol. 95. Springer Science & Business Media, 2013.

[5] A. Duarte, J. Pantrigo, and M. Gallego, "Metaheurísticas," *Madrid: Dykinson*, 2007.

[6] M. Grötschel, M. Padberg, E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Schmoys, "The traveling salesman problem," 1985.

[7] E. L. E. LAWLER, *The travelling salesman problem : A guided tour of combinatorial optimization*. Wiley, 1985.

[8] M. Epstein, "Number: the language of science. by prof. tobias dantzig. pp. xi+ 260+ ll plates.(lon-don: George allen and unwin, ltd., 1930.) los. net. the author of this interesting book has achieved," *Nature*, 1931.

[9] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[10] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[11] J. Hartmanis, "Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson)," *Siam Review*, vol. 24, no. 1, p. 90, 1982.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[13] S. S. Skiena, *The algorithm design manual*, vol. 2. Springer, 1998.

[14] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[15] M. Sharir, "A strong-connectivity algorithm and its applications in data flow analysis," *Computers & Mathematics with Applications*, vol. 7, no. 1, pp. 67–72, 1981.

[16] A. B. Kahn, "Topological sorting of large networks," *Communications of the ACM*, vol. 5, no. 11, pp. 558–562, 1962.

[17] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[18] G. Polya, *How to Solve It: A New Aspect of Mathematical Method: A New Aspect of Mathematical Method*. Princeton university press, 1945.

[19] L. A. Wolsey and G. L. Nemhauser, *Integer and combinatorial optimization*, vol. 55. John Wiley & Sons, 1999.

[20] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, vol. 13, no. 5, pp. 533–549, 1986.

[21] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67 – 71, 1989.

[22] T. A. Feo, M. G. C. Resende, and S. H. Smith, "A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set.," *Operations Research*, vol. 42, no. 5, pp. 860–878, 1994.

[23] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & operations research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[24] F. Glover, "Scatter search and path relinking," *New ideas in optimization*, vol. 138, 1999.

[25] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control and cybernetics*, vol. 29, no. 3, pp. 653–684, 2000.

[26] M. Laguna, R. Marti, and R. C. Martí, *Scatter search: methodology and implementations in C*. Springer Science & Business Media, 2003.

[27] S. P. Borgatti, M. G. Everett, and J. C. Johnson, *Analyzing social networks*. Sage, 2018.

[28] S. N. Dorogovtsev and J. F. Mendes, *Evolution of networks: From biological nets to the Internet and WWW*. OUP Oxford, 2013.

[29] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816, 2009.

[30] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[31] K. Almgren and J. Lee, "An empirical comparison of influence measurements for social network analysis," *Social Network Analysis and Mining*, vol. 6, no. 1, pp. 1–18, 2016.

[32] K. Ikeda, G. Hattori, C. Ono, H. Asoh, and T. Higashino, "Twitter user profiling based on text and community mining for market analysis," *Knowledge-Based Systems*, vol. 51, pp. 35–47, 2013.

[33] M. GLADWELL and T. Point, "How little things can make a big difference," 2000.

[34] R. K. Bakshi, N. Kaur, R. Kaur, and G. Kaur, "Opinion mining and sentiment analysis," in *2016 3rd international conference on computing for sustainable global development (INDIACom)*, pp. 452–455, IEEE, 2016.

[35] C. Pizzuti, "Evolutionary computation for community detection in networks: A review," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 464–483, 2018.

[36] A. Said, R. A. Abbasi, O. Maqbool, A. Daud, and N. R. Aljohani, "CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks," *Applied Soft Computing*, vol. 63, pp. 59–70, 2018.

[37] M. Moradi and S. Parsa, "An evolutionary method for community detection using a novel local search strategy," *Physica A: Statistical Mechanics and its Applications*, vol. 523, pp. 457–475, 2019.

[38] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan, and L. Jiao, "Greedy discrete particle swarm optimization for large-scale social network clustering," *Information Sciences*, vol. 316, pp. 503–516, 2015.

[39] A. Gonzalez-Pardo, J. J. Jung, and D. Camacho, "Aco-based clustering for ego network analysis," *Future Generation Computer Systems*, vol. 66, pp. 160–170, 2017.

[40] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[41] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[42] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *International symposium on computer and information sciences*, pp. 284–293, Springer, 2005.

[43] C. Cao, Q. Ni, and Y. Zhai, "An improved collaborative filtering recommendation algorithm based on community detection in social networks," in *Proceedings of the 2015 annual conference on genetic and evolutionary computation*, pp. 1–8, 2015.

[44] N. Zalmout and M. Ghanem, "Multidimensional community detection in twitter," in *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*, pp. 83–88, IEEE, 2013.

[45] D. Camacho, I. Gilpérez-López, A. Gonzalez-Pardo, A. Ortigosa, and C. Urruela, "Risktrack: a new approach for risk assessment of radicalisation based on social media data," in *CEUR Workshop Proceedings*, 2016.

[46] E. Gregori, L. Lenzini, and S. Mainardi, "Parallel k-clique community detection on large-scale networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1651–1660, 2012.

[47] F. Hao, G. Min, Z. Pei, D.-S. Park, and L. T. Yang, "$k$-clique community detection in social networks based on formal concept analysis," *IEEE Systems Journal*, vol. 11, no. 1, pp. 250–259, 2015.

[48] X. Wen, W.-N. Chen, Y. Lin, T. Gu, H. Zhang, Y. Li, Y. Yin, and J. Zhang, "A maximal clique based multiobjective evolutionary algorithm for overlapping community detection," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 363–377, 2016.

[49] W. Luo, N. Lu, L. Ni, W. Zhu, and W. Ding, "Local community detection by the nearest nodes with greater centrality," *Information Sciences*, vol. 517, pp. 377–392, 2020.

[50] G. Bello-Orgaz, S. Salcedo-Sanz, and D. Camacho, "A multi-objective genetic algorithm for overlapping community detection based on edge encoding," *Information Sciences*, vol. 462, pp. 290–314, 2018.

[51] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[52] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proceedings of the 2005 SIAM international conference on data mining*, pp. 274–285, SIAM, 2005.

[53] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *J. ACM*, vol. 51, p. 497–515, may 2004.

[54] H. Almeida, D. O. G. Neto, W. M. Jr., and M. J. Zaki, "Is There a Best Quality Metric for Graph Clusters?," in *ECML/PKDD (1)* (D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, eds.), vol. 6911 of *Lecture Notes in Computer Science*, pp. 44–59, Springer, 2011.

[55] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[56] C. Pizzuti, "Ga-net: A genetic algorithm for community detection in social networks," in *International conference on parallel problem solving from nature*, pp. 1081–1090, Springer, 2008.

[57] G. W. Flake, S. Lawrence, and C. L. Giles, "Efficient identification of web communities," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 150–160, 2000.

[58] L. Angelini, S. Boccaletti, D. Marinazzo, M. Pellicoro, and S. Stramaglia, "Identification of network modules by optimization of ratio association," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 17, no. 2, p. 023114, 2007.

[59] Y.-C. Wei and C.-K. Cheng, "Ratio cut partitioning for hierarchical designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 7, pp. 911–921, 1991.

[60] R. Li, S. Hu, H. Zhang, and M. Yin, "An efficient local search framework for the minimum weighted vertex cover problem," *Information Sciences*, vol. 372, pp. 428–445, 2016.

[61] M. Yannakakis, "Node-deletion problems on bipartite graphs," *SIAM Journal on Computing*, vol. 10, no. 2, pp. 310–327, 1981.

[62] M. Garey and D. Johnson, *Computers and Intractability - A guide to the Theory of NP-Completeness*. San Fransisco: Freeman, 1979.

[63] P. Bedi and C. Sharma, "Community detection in social networks," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 3, pp. 115–135, 2016.

[64] M. Gong, L. Ma, Q. Zhang, and L. Jiao, "Community detection in networks by using multiobjective evolutionary algorithm with decomposition," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 15, pp. 4050–4060, 2012.

[65] P. Ji, S. Zhang, and Z. Zhou, "A decomposition-based ant colony optimization algorithm for the multi-objective community detection," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 1, pp. 173–188, 2020.

[66] C. Shi, Z. Yan, Y. Cai, and B. Wu, "Multi-objective community detection in complex networks," *Applied Soft Computing*, vol. 12, no. 2, pp. 850–859, 2012.

[67] B. Amiri, L. Hossain, J. W. Crawford, and R. T. Wigand, "Community detection in complex networks: Multi–objective enhanced firefly algorithm," *Knowledge-Based Systems*, vol. 46, pp. 1–11, 2013.

[68] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Comput. Surv.*, vol. 45, no. 4, 2013.

[69] S. Gregory, "An algorithm to find overlapping community structure in networks," in *Knowledge Discovery in Databases: PKDD 2007* (J. N. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, eds.), pp. 91–102, Springer Berlin Heidelberg, 2007.

[70] S. Gregory, "A fast algorithm to find overlapping communities in networks," in *Machine Learning and Knowledge Discovery in Databases* (W. Daelemans, B. Goethals, and K. Morik, eds.), pp. 408–423, Springer Berlin Heidelberg, 2008.

[71] J. M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki, "Sequential algorithm for fast clique percolation," *Phys. Rev. E*, vol. 78, p. 026109, 2008.

[72] S. Gregory, "Finding overlapping communities in networks by label propagation," *New journal of Physics*, vol. 12, no. 10, p. 103018, 2010.

[73] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[74] A. Lázár, D. Abel, and T. Vicsek, "Modularity measure of networks with overlapping communities," *EPL (Europhysics Letters)*, vol. 90, no. 1, p. 18001, 2010.

[75] X. Zeng, W. Wang, C. Chen, and G. G. Yen, "A consensus community-based particle swarm optimization for dynamic community detection," *IEEE transactions on cybernetics*, vol. 50, no. 6, pp. 2502–2513, 2019.

[76] Z. Wang, C. Wang, C. Gao, X. Li, and X. Li, "An evolutionary autoencoder for dynamic community detection," *Science China Information Sciences*, vol. 63, no. 11, pp. 1–16, 2020.

[77] M.-G. Gong, L.-J. Zhang, J.-J. Ma, and L.-C. Jiao, "Community detection in dynamic social networks based on multiobjective immune algorithm," *Journal of computer science and technology*, vol. 27, no. 3, pp. 455–467, 2012.

[78] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, pp. P09008–P09008, sep 2005.

[79] F. Schoen, "Two-phase methods for global optimization," in *Handbook of global optimization*, pp. 151–177, Springer, 2002.

[80] R. Martí, J. M. Moreno-Vega, and A. Duarte, "Advanced multi-start methods," in *Handbook of metaheuristics*, pp. 265–281, Springer, 2010.

[81] R. Martí, J. A. Lozano, A. Mendiburu, and L. Hernando, "Multi-start methods," in *Handbook of heuristics*, pp. 155–175, Springer, 2018.

[82] J. P. Hart and A. W. Shogan, "Semi-greedy heuristics: An empirical study," *Operations Research Letters*, vol. 6, no. 3, pp. 107–114, 1987.

[83] J. L. G. Velarde, *Mathematical models for hitch assignments in intermodal transportation*. PhD thesis, The University of Texas at Austin, 1990.

[84] A. D. Fernández, J. G. Velarde, M. Laguna, P. Mascato, F. Tseng, F. Glover, and H. Ghaziri, "Optimización heurística y redes neuronales," *ed. Paraninfo SA, Madrid, España*, 1996.

[85] M. G. Resende and C. C. Ribeiro, "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications," in *Handbook of metaheuristics*, pp. 283–319, Springer, 2010.

[86] M. Prais and C. C. Ribeiro, "Parameter variation in grasp implementations," in *Extended abstracts of the third metaheuristics international conference*, pp. 375–380, 1999.

[87] M. Prais and C. C. Ribeiro, "Parameter variation in grasp procedures," *Investigación Operativa*, vol. 9, no. 1, pp. 1–20, 2000.

[88] S. Binato and G. C. Oliveira, "A reactive grasp for transmission network expansion planning," in *Essays and surveys in metaheuristics*, pp. 81–100, Springer, 2002.

[89] M. Prais and C. C. Ribeiro, "Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment," *INFORMS Journal on Computing*, vol. 12, no. 3, pp. 164–176, 2000.

[90] S. Binato, W. Hery, D. Loewenstern, and M. G. Resende, "A grasp for job shop scheduling," in *Essays and surveys in metaheuristics*, pp. 59–79, Springer, 2002.

[91] I. Charon and O. Hudry, "The noising method: a new method for combinatorial optimization," *Operations Research Letters*, vol. 14, no. 3, pp. 133–137, 1993.

[92] W. J. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver, "Combinatorial optimization," *Oberwolfach Reports*, vol. 5, no. 4, pp. 2875–2942, 2009.

[93] C. C. Ribeiro, E. Uchoa, and R. F. Werneck, "A hybrid grasp with perturbations for the steiner problem in graphs," *INFORMS Journal on Computing*, vol. 14, no. 3, pp. 228–246, 2002.

[94] J. L. Bresina, "Heuristic-biased stochastic sampling," in *AAAI/IAAI, Vol. 1*, pp. 271–278, 1996.

[95] C. Fleurent and F. Glover, "Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory," *INFORMS Journal on Computing*, vol. 11, no. 2, pp. 198–204, 1999.

[96] N. Mladenovic, "A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization," in *papers presented at Optimization Days*, vol. 12, 1995.

[97] P. Hansen, B. Jaumard, N. Mladenovic, and A. Parreira, "Variable neighborhood search for weighted maximum satisfiability problem," *Les Cahiers du GERAD*, vol. 62, 2000.

[98] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, *Variable Neighborhood Search*, pp. 61–86. Boston, MA: Springer US, 2010.

[99] F. Glover and M. Laguna, "Tabu search," in *Handbook of combinatorial optimization*, pp. 2093–2229, Springer, 1998.

[100] T. Jones *et al.*, "One operator, one landscape," *Santa Fe Institute Technical Report*, pp. 95–02, 1995.

[101] A. Herrán, M. J. Colmenar, and A. Duarte, "A variable neighborhood search approach for the vertex bisection problem," *Information Sciences*, vol. 476, pp. 1–18, 2019.

[102] J. Sánchez-Oro, A. D. López-Sánchez, and M. J. Colmenar, "A general variable neighborhood search for solving the multi-objective open vehicle routing problem," *Journal of Heuristics*, pp. 1–30, 2017.

[103] A. Duarte, J. Sánchez-Oro, N. Mladenović, and R. Todosijević, "Variable neighborhood descent," *Handbook of Heuristics*, pp. 341–367, 2018.

[104] J. Sánchez-Oro, A. Martínez-Gavara, M. Laguna, A. Duarte, and R. Martí, "Variable neighborhood descent for the incremental graph drawing," *Electronic Notes in Discrete Mathematics*, vol. 58, pp. 183 – 190, 2017. 4th International Conference on Variable Neighborhood Search.

[105] A. Duarte, L. F. Escudero, R. Martí, N. Mladenovic, J. J. Pantrigo, and J. Sánchez-Oro, "Variable neighborhood search for the vertex separation problem," *Computers & Operations Research*, vol. 39, no. 12, pp. 3247–3255, 2012.

[106] J. Sánchez-Oro, A. D. López-Sánchez, and J. M. Colmenar, "A general variable neighborhood search for solving the multi-objective open vehicle routing problem," *Journal of Heuristics*, Dec 2017.

[107] P. Hansen, N. Mladenović, and D. Perez-Britos, "Variable neighborhood decomposition search," *Journal of Heuristics*, vol. 7, pp. 335–350, Jul 2001.

[108] P. Hansen and N. Mladenović, *Variable Neighborhood Search*, pp. 211–238. Boston, MA: Springer US, 2005.

[109] E. G. Pardo, N. Mladenović, J. J. Pantrigo, and A. Duarte, "Variable formulation search for the cutwidth minimization problem," *Applied Soft Computing*, vol. 13, no. 5, pp. 2242 – 2252, 2013.

[110] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European journal of operational research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[111] L. Delgado-Antequera, R. Caballero, J. Sánchez-Oro, J. Colmenar, and Martí, "Iterated greedy with variable neighborhood search for a multiobjective waste collection problem," *Expert Systems with Applications*, vol. 145, p. 113101, 2020.

[112] J. Sánchez-Oro and A. Duarte, "Iterated greedy algorithm for performing community detection in social networks," *Future Generation Computer Systems*, vol. 88, pp. 785–791, 2018.

[113] J. D. Quintana, R. Martin-Santamaria, J. Sanchez-Oro, and A. Duarte, "Solving the regenerator location problem with an iterated greedy approach," *Applied Soft Computing*, vol. 111, p. 107659, 2021.

[114] M. Lozano, D. Molina, C. Garcı, *et al.*, "Iterated greedy for the maximum diversity problem," *European Journal of Operational Research*, vol. 214, no. 1, pp. 31–38, 2011.

[115] F. Glover, "A template for scatter search and path relinking," in *European conference on artificial evolution*, pp. 1–51, Springer, 1997.

[116] R. Martí and M. Laguna, "Scatter search: Diseno básico y estrategias avanzadas," *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, vol. 7, no. 19, p. 0, 2003.

[117] F. Glover, M. Laguna, and R. Marti, "Scatter search and path relinking: Advances and applications," in *Handbook of metaheuristics*, pp. 1–35, Springer, 2003.

[118] F. Glover, M. Laguna, and R. Martí, *New Ideas and Applications of Scatter Search and Path Relinking*, pp. 367–383. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[119] F. Glover, M. Laguna, and R. Martí, *Scatter Search and Path Relinking: Foundations and Advanced Designs*, pp. 87–99. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[120] T. Xifeng, Z. Ji, and X. Peng, "A multi-objective optimization model for sustainable logistics facility location," *Transportation Research Part D: Transport and Environment*, vol. 22, pp. 45–48, 2013.

[121] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[122] M. Lü, Z. Ning, X. Zhi, and X. Li, "Multi-objective optimization of key technologies in gasoline direct-injection engine based on fuel economy," *Fuel*, vol. 309, p. 122002, 2022.

[123] V. Pareto *et al.*, *Manual of political economy*. AM Kelley, 1971.

[124] K. Deb, "Multi-objective optimization," in *Search methodologies*, pp. 403–449, Springer, 2014.

[125] C.-L. Hwang and A. S. M. Masud, *Multiple objective decision making—methods and applications: a state-of-the-art survey*, vol. 164. Springer Science & Business Media, 2012.

[126] S. Natarajan and P. Tadepalli, "Dynamic preferences in multi-criteria reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*, pp. 601–608, 2005.

[127] L. Barrett and S. Narayanan, "Learning all optimal policies with multiple criteria," in *Proceedings of the 25th international conference on Machine learning*, pp. 41–47, 2008.

[128] K. Miettinen, *Nonlinear multiobjective optimization*, vol. 12. Springer Science & Business Media, 2012.

[129] A. Ellithy, O. Abdelkhalik, and J. Englander, "Multi-objective hidden genes genetic algorithm for multigravity-assist trajectory optimization," *Journal of Guidance, Control, and Dynamics*, pp. 1–17, 2022.

[130] F. Felten, G. Danoy, E.-G. Talbi, and P. Bouvry, "Metaheuristics-based exploration strategies for multi-objective reinforcement learning," in *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, pp. 662–673, SCITEPRESS-Science and Technology Publications, 2022.

[131] F. Xu and H. Zhang, "Energy efficiency and spectral efficiency tradeoff in irs-assisted downlink mmwave noma systems," *IEEE Wireless Communications Letters*, 2022.

[132] M. Laumanns, L. Thiele, and E. Zitzler, "An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method," *European Journal of Operational Research*, vol. 169, no. 3, pp. 932–942, 2006.

[133] G. K. Tutunchi and Y. Fathi, "Effective methods for solving the bi-criteria p-center and p-dispersion problem," *Computers & operations research*, vol. 101, pp. 43–54, 2019.

[134] S. Pérez-Peló, J. Sánchez-Oro, and A. Duarte, "Finding weaknesses in networks using greedy randomized adaptive search procedure and path relinking," *Expert Systems*, vol. 37, no. 6, p. e12540, 2020.

[135] A. Duarte, J. Sánchez-Oro, M. G. Resende, F. Glover, and R. Martí, "Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization," *Information Sciences*, vol. 296, pp. 46–60, 2015.

[136] S. Pérez-Peló, J. Sánchez-Oro, R. Martín-Santamaría, and A. Duarte, "On the analysis of the influence of the evaluation metric in community detection over social networks," *Electronics*, vol. 8, no. 1, p. 23, 2019.

[137] S. Pérez-Peló, J. Sanchez-Oro, A. D. Lopez-Sanchez, and A. Duarte, "A multi-objective parallel iterated greedy for solving the p-center and p-dispersion problem," *Electronics*, vol. 8, no. 12, p. 1440, 2019.

[138] A. Duarte, J. J. Pantrigo, E. G. Pardo, and N. Mladenovic, "Multi-objective variable neighborhood search: an application to combinatorial optimization problems," *Journal of Global Optimization*, vol. 63, no. 3, pp. 515–536, 2015.

[139] W. Luo, D. Zhang, H. Jiang, L. Ni, and Y. Hu, "Local community detection with the dynamic membership function," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 3136–3150, 2018.

[140] A. McAllister *et al.*, "A new heuristic algorithm for the linear arrangement problem," tech. rep., Faculty of Computer Science, University of New Brunswick, 1999.

[141] M. Y. Wang, "Deep graph library: Towards efficient and scalable deep learning on graphs," in *ICLR workshop on representation learning on graphs and manifolds*, 2019.