UNIVERSIDAD DE LA REPUBLICA
URUGUAY

GRASP Heuristics for the Stochastic Weighted Graph Fragmentation Problem

Nicole Rosenstock Cukrowicz

Programa de Maestría en Investigación de Operaciones
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Diciembre de 2018

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

GRASP Heuristics for the Stochastic Weighted Graph Fragmentation Problem

Nicole Rosenstock Cukrowicz

Tesis de Maestría presentada al Programa de Posgrado en Investigación de Operaciones, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magister en Investigación de Operaciones.

Director académico:
Dr. Franco Robledo
Directores de Tesis:
Dr. Pablo Romero
Dr. Juan Piccini

Montevideo – Uruguay
Diciembre de 2018

Rosenstock Cukrowicz, Nicole

GRASP Heuristics for the Stochastic Weighted Graph Fragmentation Problem / Nicole Rosenstock Cukrowicz. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2018.

VIII, 65 p. 29, 7cm.

Director académico:

Franco Robledo

Directores de Tesis:

Pablo Romero

Juan Piccini

Tesis de Maestría – Universidad de la República, Programa en Investigación de Operaciones, 2018.

Referencias bibliográficas: p. 58 – 65.

1. Optimización combinatoria, 2. Nodos críticos, 3. GRASP, 4. Path Relinking, 5. Complejidad computacional. I. Romero, Pablo, Piccini, Juan, . II. Universidad de la República, Programa de Posgrado en Investigación de Operaciones. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Montevideo – Uruguay
Diciembre de 2018

RESUMEN

Los nodos críticos juegan un rol fundamental en la conectividad de las redes. Su identificación es importante para el diseño de estrategias eficientes para prevenir que tanto un software malicioso como una epidemia se propaguen por la red. En este contexto, el Stochastic Weighted Graph Fragmentation Problem (SWGFP) es un problema de optimización combinatoria perteneciente a la clase de problemas $\mathcal{NP}-$Completos. El objetivo consiste en miniminizar el impacto de un ataque aleatorio en un nodo de la red, seleccionando adecuadamente nodos a inmunizar con un presupuesto acotado. En el SWGFP se asume que el ataque sigue una ley de probabilidad conocida en los nodos, y que afecta a toda la componente conexa del nodo seleccionado. En esta tesis se desarrolla una solución GRASP enriquecida con Path-Relinking para abordar el SWGFP. Se estudia el rendimiento de la propuesta ante tres escenarios de ataque, en comparación con una variante de GRASP anteriormente desarrollada de la literatura y una heurística aleatoria o *Random* para el problema en la cual los nodos son elegidos al azar. Los experimentos computacionales muestran que el algoritmo basado en Conjuntos Independientes que se desarrolla en esta tesis, presenta un mejor desempeño que los dos restantes, con valores inferiores del número esperado de pérdidas y mayor robustez.

Palabras claves:
Optimización combinatoria, Nodos críticos, GRASP, Path Relinking, Complejidad computacional.

ABSTRACT

Critical nodes play a major role in network connectivity. Identifying them is important to design efficient strategies to prevent malware or epidemics spread through a network. In this context, the Stochastic Weighted Graph Fragmentation Problem (SWGFP) is a combinatorial optimization problem that belongs to the $\mathcal{NP} - Complete$ class. Its objective consists in minimizing the impact of a random attack on a singleton, choosing appropiately a set of nodes to immunize given a restricted budget. In the SWGFP, it is assumed that the attack follows a known probability law and that it affects the whole connected component of the attacked node. In this thesis, a GRASP enriched with Path Relinking algorithm is developed to solve the SWGFP. Its performance is studied under three attack scenarios and compared with a GRASP variant that was previously developed in literature and with a *Random* heuristic for the problem that picks a set of nodes uniformly at random. Computational experiments show that the algorithm based on Independent Sets which is developed in this thesis, outperforms the other two, with lower expected loss scores and higher robustness.

Keywords:
Combinatorial optimization, Critical nodes, GRASP, Path Relinking, Computational complexity.

Contents

Chapter 1

Introduction

1.1. Motivation and Context

Over the last decades, the comprehension of networks' structure and functioning has awaken great interest from researchers. Networks are present everywhere and form part of our daily lives: communication networks like the Internet or cellular systems have been developing at a fast pace and have a huge impact on our society. Social networks have gained a lot of importance during the past decade; improvements in transportation networks have allowed to reach distant places more efficiently. Life itself depends on proper functioning of biological, such as a neural networks, and ecological networks, such as food webs. Diverse phenomena can be modeled using networks, where the items are called *nodes* or *vertices* and the connections between them are the *edges*. Nowadays, it is well-known that some nodes play a more important role than others in some processes that occurr throughout the network, such as spreading or cascading. The identification of these nodes is necessary for the design of strategies that ensure the proper functioning of the network over time and protect against disruptive events.
In this context, *Graph Fragmentation Problem* (GFP) is a combinatorial optimization problem that arised as a contribution from the doctoral dissertation of Dr.Juan Piccini, supervised by Dr.Franco Robledo and Dr.Pablo Romero (Piccini (2006)). The GFP seeks to minimize the expected loss in a newtork after a singleton has been infected by removing a set of B nodes. This

problem is formally defined in the following manner:

$$\min_{U \subseteq V} \sum_{i=1}^{k} p_i n_i$$
$$s.t. |U| = B,$$

where $p_i = \frac{n_i}{n}$ is the probability of the event $v \in V_i$, being v the node uniformly chosen at random. In this dissertation, the $\mathcal{NP}$-Hardness for GFP is also established and a first GRASP is developed for the problem.

In M.Sc.Natalia Castro's master's thesis, with external collaboration from Manuel Aprile, it is proved the non-existence of an approximation algorithm with factor lower than 5/3, unless $\mathcal{P} = \mathcal{NP}$ and optimal solutions for GFP are presented for acyclic graphs (Castro (2018); Aprile et al. (2017, 2018)).

Additionally, as a result of the doctoral dissertation of Dr. Graciela Ferreira, it was developed a Mixed Integer Linear Programming (MILP) for GFP and efficient bounds for the optimal value of any instance of the problem (Ferreira (2018)).

In Piccini et al. (2018) a GRASP for GFP is introduced while in Aprile et al. (2018) an analysis and a synthesis of the main up-to-date contributions of GFP are presented.

In this thesis, we will focus on the *Stochastic Weighted Graph Fragmentation Problem* (SWGFP), a combinatorial optimization problem whose main goal is to find a set of weighted nodes to immunize in order to stop an epidemic spread, once a node is infected. It is a generalization of the Graph Fragmentation Problem. Although this combinatorial optimization problem was originally thought to be applied in an epidemiological context, it can also be used to model other catastrophic events. The main contributions of this thesis can be summarized in the following items:

- A generalization of the GFP, called *Stochastic Weighted Graph Fragmentation Problem*, is here introduced.
- An adaptation of the GRASP introduced by Piccini et al. (2018) is here offered.
- A novel GRASP heuristic, called GRASP-MIS, is here developed. It exploits properties of maximal independent sets, and it is enriched with Path-Relinking as a post-optimization stage.

- A faithful comparison shows that the novel GRASP-MIS heuristics outperforms the previous proposals.
- A curious interplay is here discovered: the GFP is equivalent to the Critical Node Problem or CNP, which was originally conceived for its connectivity importance (Arulselvan et al. (2007)). Therefore, the problem was independently studied by different researchers, under different motivations and scenarios.

It is worth to remark that the CNP has an extensive record in the scientific literature. As a corollary, this thesis promotes the interplay between different research teams, that where incidentally working in the same problem.

1.2. Manuscript organization

This thesis is organized in the following manner: Chapter 1 reviews some basic concepts on Graph and Complexity theories which provide the foundations for the development of our work.
Chapter 2 describes the Critical Node Detection Problem, its potential applications and its variants. The Critical Node Problem, a variant of the Critical Node Detection Problem, shares common properties with the object under study, the SWGFP. A variant of this problem -the Critical Node Problem- is mentioned due to its equivalence with the Graph Fragmentation Problem and its closeness with the Stochastic Weighted Graph Fragmentation Problem.
Chapter 3 outlines the main features of the SIR epidemic model and the Node Immunization Problem (NIP). It is worth to remark that the NIP is the cornerstone of the problem under study since it represents a generalization of the Graph Fragmentation Problem. The Graph Fragmentation Problem is thoroughly explained in this chapter and the equivalence between GFP and CNP is demonstrated. Finally, a generalization of the GFP is here introduced: the *Stochastic Weighted Graph Fragmentation Problem*.
Chapter 4 reviews the heuristics that have been developed in literature for the Graph Fragmentation Problem and the Stochastic Weighted Graph Fragmentation Problem.
The main contributions of this thesis are presented in Chapter 5 and 6. Specifically, Chapter 5 presents an adaptation of the GRASP originally

conceived for GFP and introduces a new hybrid GRASP with Path Relinking (GRASP-MIS) to solve the SWGFP. Chapter 6 presents a faithful comparison with state-of-the-art heuristics. Finally, Chapter 7 presents concluding remarks and trends for future work.

1.3. Graph Theory

Graph theory is considered as a branch of mathematics. Euler is often called the "father of the graph theory" since the 18^{th} century, when he proposed the "Konigsberg Bridge Problem". Given two islands linked between themselves and linked to the banks of a river by seven bridges, the problem consisted in walking across each bridge only once and returning to the starting point. Euler represented each land area with a point and each bridge with a line joining the corresponding points, generating a *graph* (Harary (1969)). After Euler, several renowned scientific characters started working with graphs. This is how graph theory started gaining more and more importance.
A graph G consists of two elements:

- Vertices, also called nodes or points, denoted by $V(G)$
- Edges: the lines connecting two vertices, denoted by $E(G)$

This graph is usually referred as $G = (V, E)$. Vertices x and y are said to be *adjacent* or *neighbors* if there is an edge $e = \{x, y\}$. Also, the edge e is said to be incident on x and y. The degree of a vertex v, usually denoted as $deg(v)$ or $d(v)$, is the number of edges that are incident on v. A *directed graph* consists of a set V of vertices and a set A of ordered pairs of distinct vertices called directed edges or *arcs*. On the other hand, an *undirected graph* $G = (V, E)$ consists of a set of nodes V and unordered pairs of elements E called edges. A *walk* is an alternated sequence of vertices and edges $v_1, e_1, v_2, \ldots, e_{n-1}, v_n$ where each edge e_j connects the vertices v_j and v_{j+1}. It is closed if $v_1 = v_n$ and is open otherwise. It is a *trail* if all the edges are distinct, and a *path* if all the vertices (and therefore all the lines) are distinct. The set of vertices that can be reached from a vertex v by any path in the graph determines a *connected component*. A non-empty graph G is connected if any two of its vertices are linked by a path in G.
Given a graph $G = (V, E)$, a graph $G' = (V', E')$ is called a *subgraph* of G

if the vertices and edges of G' are contained in G: $V' \subseteq V$ and $E' \subseteq E$. If $G' \subseteq G$ and G' contains all the edges $xy \in E$ with $x, y \in V'$, G' is an induced subgraph of G.
Finally, we will present two definitions that are crucial for developing our work.

Definition 1 *Given an undirected graph $G = (V, E)$, a **vertex cover** of G is a subset V' of V such that if edge $\{x, y\}$ is an edge of G then either x or y (or both) belongs to V'. That is, if e is an edge of G then at least one of its endpoints lies in V'.*

Definition 2 *A set of vertices S is an **independent set** if there are no edges $\{x, y\}$ such that both x and y are in S. That is, there are no two adjacent vertices in the independent set. S is a **Maximal Independent Set(MIS)** if it is an independent set that is not a proper subset of any other independent set. No vertex can be added to S without violating independence.*

1.4. Complexity Theory

Complexity theory is a field of Computer Science that aims to study the computational resources which are necessary to solve a computational problem. A problem is defined as the question to be answered that possess parameters that are left unspecified. An *instance* of the problem consists of a set of particular values specified for the parameters. An algorithm consists of a sequence of instructions intended to solve these problems: when it is is applied to any instance of the problem, the algorithm produces a solution for that instance. Finding the fastest algorithm to solve a problem is of great concern among researchers. The time required by an algorithm will determine whether it is useful for solving problems. The time requirements are expressed in terms of the input size of an instance. The input in combinatorial optimization problems is a combinatorial object that must be encoded representing it as a sequence of symbols. The length of this sequence is the size of the input (Papadimitriou and Steiglitz (1998)).The time complexity function for an algorithm is determined by finding the largest amount of time needed to solve a problem instance of a certain size, for all possible input lengths (Garey (1979)). Computer scientists defined two categories in order to classify algorithms according to their execution time: *polynomial time* and *exponential time* algorithms (Papadimitriou and Steiglitz (1998)). A polynomial-time algorithm has a complexity function $O(p(n))$, where $p(n)$ is a polynomial function and n represents the input length. Algorithms for which time complexity function cannot be bounded by a polynomial are said to be exponential time algorithms. A problem is considered *intractable* if no polynomial algorithm can solve it. Intractability was first mentioned by Alan Turing, when he proved that certain problems are "undecidable". It was not until early 1970's that it was shown that there exists intractable but decidable problems (Garey (1979)).

1.4.1. Complexity Classes

We will now focus on decision problems: problems that accept a YES/NO answer for every instance. A complexity class is a set of problems that can be solved using a certain amount of some computational resource. The complexity class $\mathcal{P}$ contains all decision problems that can be solved by a deterministic algorithm within polynomial time. It is said that they are "efficiently solvable" in polynomial time. The complexity class $\mathcal{NP}$ refers to decision problems whose solution can be verified in polynomial time. If a problem can be solved in polynomial time, it is clear that can also be verified in polynomial time, so $\mathcal{P} \subseteq \mathcal{NP}$. The question about whether $\mathcal{P} = \mathcal{NP}$ has attracted much attention from computer scientists. This would imply that any problem that can be verified in polynomial time could also be solved in polynomial time.

A reduction is a conversion from one problem to another so that a solution to the second problem can be used to solve the first problem (Karp (1972)). A reduction from problem $\mathcal{A}$ to problem $\mathcal{B}$ is a polynomial-time algorithm f that transforms any instance I of $\mathcal{A}$ into an instance of $\mathcal{B}$, $f(I)$. Any solution S of $f(I)$ is transformed back into a solution of I, by a polynomial-time algorithm. If there is no solution for $f(I)$, then there is no solution for I (Dasgupta et al. (2008)).

The class of $\mathcal{NP}$-complete problems consists in those problems P that can be verified in polynomial time and that any other problem Q can be reduced to one of this in polynomial time. The $\mathcal{NP}$-complete class has two important properties (Papadimitriou and Steiglitz (1998)):

(i) No $\mathcal{NP}$-complete problem can be solved by any known polynomial algorithm

(ii) If there is a polynomial algorithm for any $\mathcal{NP}$-complete problem, then there are polynomial algorithms for all $\mathcal{NP}$-complete problems

Finally, the $\mathcal{NP}$-Hard class comprises all problems that are "at least as hard" as every problem in $\mathcal{NP}$. Formally, a problem $\mathcal{A}$ is $\mathcal{NP}$-hard, if all problems in $\mathcal{NP}$ can be reduced to $\mathcal{A}$ in polynomial time. These kind of problem are not necessarily in $\mathcal{NP}$, nor are decision problems. It also includes search problems and optimization problems (Sipser (1996)).

Chapter 2

Background

The scientific literature offers a vast number of works in the field of node criticality and network vulnerability analysis under random and targeted attacks. The goal of this chapter is to have a better understanding of node criticality and its potential applications.

This chapter is organized in the following manner. Section 2.1 presents the concept of critical nodes and its importance in different branches of knowledge. Specific centrality measures are covered in Section 2.2. The Critical Node Detection Problem (CNDP) is formally defined in Section 2.3. They key is to find a set of nodes whose removal degrades network connectivity. We also provide comments on a variation, called Critical Node Problem (CNP), and its main properties.

2.1. Node Criticality

Critical nodes are nodes whose removal results in a degradation of network functionality due to the disruption of network connectivity. The drop of the performance of the network when a node or set of nodes, is removed will determine their importance (Latora and Marchiori (2004); Zhao et al. (2005); Restrepo et al. (2006); Arulselvan et al. (2009)) . Since critical nodes play a key role in preserving network connectivity and robustness, identifying them is crucial either if the main goal is to delete them to stop the spread of undesirable events or to protect them in order to promote the difussion of desirable ones. The issue of identifying the most critical nodes in a network has long been addressed, in particular, for nodes that are important for network connectivity (Chen et al. (2012); Wei et al. (2013); Lalou et al. (2018)).

2.1.1. Importance of Critical Nodes

Identification of critical nodes is a fundamental problem that has several applications in different areas (Barabási and Bonabeau (2003); Crucitti et al. (2004); Chen et al. (2013); Malliaros et al. (2016)). In social network analysis, it might help to understand many properties of social interactions represented by networks and the role of certain actors in that graph (Kempe et al. (2005); Borgatti (2005, 2006)). The study of these properties could also help to assess the robustness and connectivity of communication networks and therefore better deal with communication breakdowns in human and telecommunication networks (Arulselvan et al. (2009); Dinh et al. (2010); Addis et al. (2016)). Finding the critical nodes in a network might contribute to protect a communication or transportation network against disruptive events such as terrorist attacks. A terrorist network can also be modelled as a graph whose critical nodes represent the terrorists that should be identified in order to produce a breakdown in communication between individuals in the network (Arulselvan et al. (2009)). Another area where finding key nodes has important applications is in the field of transportation engineering and planification of emergency evacuations, where it must be ensured an effective transportation of people and goods (Crucitti et al. (2004); Guimera et al. (2005); Arulselvan et al. (2007)). In Boginski and Commander (2009) the problem is presented in the context of computational biology, in particular protein-protein interaction. Identifying the proteins whose removal might help to stop proliferation of harmful organisms such as virus or bacteria, can be useful in drug design. Moreover, finding essential nodes in this kind of biological network can contribute to locate optimum drug targets and avoiding other targets which could be lethal (Billur Engin et al. (2014)). Also, in the biological field, network theory has been applied to the study of brain networks in order to find influential nodes that allows to better understand brain structure, properties and functionality (Joyce et al. (2010); Power et al. (2013)). Additionally, another application arises in the healthcare domain, more precisely, in the design of immunization strategies for preventing a pandemic disease spread (Ventresca and Aleman (2013); Gupta et al. (2015); Malliaros et al. (2016)). The critical nodes can be seen as individuals who are "most likely to be influential spreaders and maximally permit information spread through the network" (Ventresca and Aleman (2015)). Accurately identifying these individuals, in order to vaccinate them, is crucial to reduce the spread and impact of a disease throughout a network, since immunized individuals cannot propagate the disease (Arulselvan et al. (2009)). In this thesis, we are focused on the concept of critical nodes applied to the design of efficient immunization strategies to control an epidemic outbreak.

2.2. Centrality-based Approaches

Several centrality measures have been proposed to analyze the influence of nodes on a network (Freeman (1978); Newman (2005); Borgatti and Everett (2006)). Although centrality measures were first applied in the context of human communication (Bavelas (1948, 1950)), they soon began to be applied in other fields (Freeman (1978); Borgatti (2005); Estrada and Bodin (2008); Borgatti and Everett (2006); Iyer et al. (2013)). Among these fields, we can found epidemiology (Chen et al. (2012); Piraveenan et al. (2013); Shams and Khansari (2014)), where the design of targeted vaccination strategies based on centrality measures has attracted much attention from researchers. Even though there exist many centrality measures, the four best-known are:

(i) Degree Centrality

(ii) Betweenness Centrality

(iii) Closeness Centrality

(iv) Eigenvector Centrality

Degree centrality counts the number of edges incident on a specific node. It is thought that nodes with higher degree centrality, have higher influence on the network (Piraveenan et al. (2013)).
Betweenness centrality is defined as the fraction of shortest paths between node pairs that pass through the node under study. Formally, it is defined:

$$B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where σ_{st} is the number of shortest paths between nodes s and t and $\sigma_{st}(v)$ is the number of shortest paths between nodes s and t that pass through node v. This measure represents the "influence of a node over the information spread through the network" (Chen et al. (2012)).

Closeness centrality is the average geodesic distance from a node to the rest of the nodes in the graph. Formally, it is defined:

$$C(v) = \frac{1}{\sum_{i \neq v} d_G(v,i)},$$

where $d_G(v,i)$ is the shortest path distance between nodes v and i in the graph G. The node closest to the rest of the nodes will have the highest measure of closeness centrality (Piraveenan et al. (2013)). Closeness represents how long information will be spread from a given node to other reachable nodes in the network (Chen et al. (2012)).

Eigenvector centrality is given by:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j,$$

where x_i denotes the centrality of vertex i. This equation can be rewritten in its matricial form:

$$\lambda x = Ax,$$

where x is the vector of centralities $x = (x_1, x_2, \dots)$ and A is the adjacency matrix of graph G whose largest eigenvalue is λ. This measure assumes that connections to people who are influential will confer a person more influence than connections to less influential people (Newman (2016)).

Centrality measures are useful to describe nodes importance. However, they fail to accurately assess nodes impact on network connectivity and fragmentation (Arulselvan et al. (2009)). There is evidence that methods based on centrality measures perform poorly when compared to other stratagies to evaluate overall network vulnerability (Grubesic et al. (2008); Dinh et al. (2012)).

2.3. Critical Node Detection Problem

The Critical Node Detection Problem (CNDP) addresses the issue of finding critical nodes in order to disconnect a network. The CNDP aims to “identify a set of nodes within a graph whose deletion minimizes or maximizes a predefined connectivity metric on the remaining graph”. Although the main goal is to disrupt graph connectivity through node deletion, the solution will depend on the chosen connectivity metric. Some connectivity metrics to be satisfied are: maximization of number of components, minimization of component size, maximization of the number of smallest components, minimization of the number of largest components, minimization of pair-wise connectivity. Different variants of the CNDP are described in Lalou et al. (2018). Such variants differ in the considered connectivity metric. The main variants which are reviewed by the authors are:

- MaxNum: maximizes number of connected components
- CC-CNP: limits the maximal component size to a given bound
- MinMaxC: minimizes the largest component size
- β-vertex-disruptors-problem: bounds pairwise connectivity to a given threshold by deleting the minimal set of nodes
- CNP: minimizes pairwise connectivity by deleting k nodes

All the variants have received significant reserach attention and several works were done to analyze their computational complexity. Some of the algorithms that have been proposed for each one of the variants of the CNDP were: Greedy heuristic (Arulselvan et al. (2009, 2011)), Genetic Algorithm (Aringhieri et al. (2016a); Soria et al. (2017)), Variable Neighborhood Search (Aringhieri et al. (2016b)), GRASP with Path Relinking (Purevsuren et al. (2016)), Dynamic Programming (Shen and Smith (2012); Di Summa et al. (2011)), Iterated Local Search (Aringhieri et al. (2016b); Zhou and Hao (2017)), Simulateed Annealing (Soria et al. (2017)). The CNP (also known as Critical Node Problem) is the CNDP variant that has attracted more attention concerning the development of new algorithms. However, as stated by Lalou et al. (2018) few studies have been made on critical nodes in weighted graphs, with nodes having different weights.

2.3.1. Critical Node Problem

The Critical Node Problem (CNP) was introduced by Arulselvan et al. (2007). Given a graph and an integer k, the objective is to find a set of k nodes whose deletion results in the maximum network fragmentation. The formal definition of CNP is:

Definition 3 *INPUT: Undirected graph $G = (V, E)$ and an integer k.*
OUTPUT: $A = argmin \{\sum_{i,j \in V \setminus A} u_{ij}(G(V \setminus A)) : |A| \leq k\}$, where

$$u_{ij} = \begin{cases} 1, & \textit{if } i \textit{ and } j \textit{ are in the same component of } G(V \setminus A) \\ 0, & \textit{otherwise} \end{cases}$$

According to the authors, the CNP objective function can be rewritten as:

$$f(A) = \sum_{h \in M} \frac{\sigma_h(\sigma_h - 1)}{2},$$

where M is the set of all maximal connected components and σ_h is the size of the h^{th} component. The objective of the CNP is to find the subset of nodes $A \subseteq V$, such that $|A| \leq k$, whose deletion minimizes the pairwise connectivity the nodes in the induced subgraph $G(V \setminus A)$. Minimizing pair-wise connectivity simultaneously maximizes the number of connected components and minimizes cardinality variance among components in $G(V \setminus A)$(Arulselvan et al. (2009)). An explanation for the choice of the objective function for the CNP is given by Arulselvan et al. (2007): "for a fixed number of components the variance in the sizes of the components is the sum of the squares of deviation of sizes of the components from the mean size of a component, which is constant. Thus minimizing the variance of the size of the components reduces to minimizing the sum of squares of the sizes of the components, which is the objective function. Also, when the sizes of the components are equal the objective function is the minimum when the number of components is the maximum". In the same work, the authors prove that CNP is $\mathcal{NP}$-complete, showing a reduction from the Independent Set Problem (ISP). The *ISP* is known to be $\mathcal{NP}$-complete (Garey (1979)).

Chapter 3

Stochastic Weighted Graph Fragmentation Problem

In this chapter, we begin with the definition of one of the milestones in mathematical modeling: the SIR epidemic modeling (Section 3.1). In Sections 3.2 and 3.3 we describe the Node Immunization Problem and the Graph Fragmentation Problem respectively. The equivalence between CNP and GFP is proved. The NIP and GFP are combinatorial optimization problems that provide the theoretical foundations for the Stochastic Weighted Graph Fragmentation Problem addressed in this tesis and described in Section 3.4. In Subsections 3.3.1 and 3.3.2 we present further details about the formulation and the computational complexity of the Graph Fragmentation Problem.

3.1. SIR Epidemic Model

The SIR model is one of the simplest models of contagion of directly-transmitted diseases. It was initially proposed by Kermark and Mckendrick (1927). Given a fixed population, it is partitioned into three classes: Susceptible(S), Infected(I) and Recovered (or Removed) (R). According to this model, the Susceptible class includes individuals that have never been infected and therefore, can acquire the disease. The Infected class includes those individuals that have acquired the disease and can transmit it to susceptible individuals. The Recovered class referrs to individuals that have already been infected but became immune for life. The transitions between each class are governed by a system of three differential equations:

$$\begin{aligned}\frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I,\end{aligned}$$

where $S(t)$,$I(t)$ and $R(t)$ represent the number of susceptible, infected and recovered individuals at time t; $\beta > 0$ is the disease transmission rate and $\gamma > 0$ is the recovery rate.
The SIR model makes the following assumptions:

- Population is large enough to consider that the size of each class is a continous variable
- Population size ($S+I+R = N$) is constant and the population is closed: there is migration
- There are no natural births or deaths
- The outbreak is short-lived
- There is no latency period: an individual that leaves the Susceptible class enters inmediately into the Infected class
- Recovery implies lifetime immunity
- Homogenous mixing: individuals interact with equal probability with everyone else

The SIR model (as well as other epidemic models) were developed with the objective of a better understanding of disease dynamics that could result in efficient disease management measures. However, similarities have been found between the behaviour of biological epidemics and the behaviour of computer malware (Kephart et al. (1993); Lloyd and May (2001); Data and Wang (2005)). For this reason, as reviewed in del Rey (2015), much effort was put into the study of malware propagation using epidemic models.

3.2. Node Immunization Problem

The Node Immunization Problem (NIP) described in Piccini et al. (2018) is a combinatorial optimization problem where the goal is to minimize the impact of an outbreak by means of node-immunization. We describe the NIP in the following paragraph.
The population is represented by a simple graph $G = (V, E)$ is simple graph where nodes represent individuals and links the relations between them. Time is slotted in the natural domain $T = \mathbb{N}$, and at $t = 0$ we have:

- Set $V^* \subset V$ is removed. An epidemic process takes place in the subgraph G' induced by $V' = V - V^*$.
- A singleton $x_0 \in V'$ is chosen uniformly at random.
- The remaining nodes in $V' - \{x_0\}$ are susceptible.

Each infected node $v \in V$ disseminates the disease to every susceptible neighbor w at time t, with a probability ruled by a profile $p(t, \mu, v, w)$. The profile p depends on the carrier v, the susceptible node w, the time t and a level of virulence μ as well. It is assumed that the probability profile tends to the unit when μ tends to infinity, this is:

$$\lim_{\mu \to \infty} p(t, \mu, v, w) = 1. \tag{3.1}$$

Once a susceptible node v becomes infected, it remains infected for a random time governed by a random variable X_v with finite mean and variance. Then, it becomes susceptible again. Let $\{I_t\}_{t \in \mathbb{N}}$ be the stochastic process that counts the number of infected individuals. The node-immunization problem is the following combinatorial optimization problem:

Definition 4 *Given a simple graph $G = (V, E)$, probability profile p and random vector $(X_v)_{v \in V}$, the goal of the General Node Immunization Problem (GNIP) is to choose a node-immunization set V^* that minimizes the peak for the process $\{I_t\}_{t \in \mathbb{N}}$, subject to a budget constraint B:*

$$\min_{V^*} \max_{t \in \mathbb{N}} E(I_t)$$
$$s.t. \ |V^*| \leq B.$$

In this work, we will focus on the Node Immunization Problem under highly virulent scenarios: μ is infinitely large and the probability profile $p = 1$ so all contacts surely disseminate the disease.

3.3. Graph Fragmentation Problem

The Graph Fragmentation Problem (GFP) was first introduced in Piccini et al. (2015). The GFP can be defined as an extremal case of the NIP when the virulence rate tends to infinity(μ is infinitely large and the probability profile is $p = 1$). The GFP focuses in finding nodes to be targeted for vaccination in order to minimize the contagion propagation through the network. An accurate identification of these nodes will lead to an optimal fragmentation of the graph while minimizing the expected loss. Although the GFP was originally thought in an epidemiological context it can be noted that it models other catastrophic events, such as fire-fighting and electric shocks. In GFP, we are given a graph $G = (V, E)$ and a budget B. We pick B nodes for immunization. A non-immunized singleton $v \in V$ is randomly affected by nature and general probabilistic rules or *profiles* determine the propagation in a discrete-time fashion. Finally, a stochastic process counts the number of affected individuals. But in highly virulent scenarios, as considered, the connected component that includes v is completely affected, and the notion of *fragmentation* is essential.

3.3.1. Mathematical Formulation

We are given a population represented by a graph $G = (V, E)$, and a budget constraint B, where B is a natural number such that $0 \leq B \leq |V|$. We can choose B nodes and immunize them: we delete the nodes from G obtaining a subgraph G', so that the chosen nodes cannot be affected by the disaster. The nature picks a node v uniformly at random from G'. The disaster kills all the members of the same connected component as v.

The goal is to minimize the expected number of deaths. Mathematically, if the subgraph G' has $V' = n$ nodes and k connected components with orders $n_1, \ldots, n_k$, the probability to choose component i is n_i/n. Therefore, the expected number of deaths is $E(G') = \sum_{i=1}^{k} n_i p_i$, with $p_i = n_i/n$. The goal of the Graph Fragmentation Problem (GFP) is to choose the node-immunization set in order to minimize the expected number of deaths:

$$\min_{U \subseteq V} \sum_{i=1}^{k} \frac{n_i^2}{n}$$
$$s.t. |U| = B.$$

Curiously enough, the GFP and the CNP are completely equivalent as combinatorial problems, since the globally optimum solutions of both problems are identical for the same instances. However, both problems were independently presented by different researchers, with apparently distinguishable scenarios, either focused on connectivity or epidemic modeling. As stated before, the CNP and the GFP are equivalent combinatorial optimization problems. Two combinatorial optimization problems (COP) f and g are equivalent if there exist $a, b \in \mathbb{R}$ such that $g(S) = af(S) + b$ and

$$\min_S f(S) = \min_S g(S)$$

A demonstration of the equivalence between CNP and GFP is presented below.

Proof 1 (Equivalence between CNP and GFP)
Be the formulation of CNP:

$$\min_{V^* \subseteq V} = \sum_k \frac{n_k(n_k - 1)}{2} = g(V^*)$$
$$s.t\ |V^*| = B$$

Be the formulation of GFP:

$$\min_{V^* \subseteq V} = \sum_k \frac{n_k^2}{\sum n_k} = f(V^*)$$
$$s.t\ |V^*| = B$$

Then,

$$\begin{cases} 2g(V^*) + (\sum n_k) & = f(V^*)(\sum n_k) \\ \sum n_k = n - B \end{cases} \implies \quad g(V^*) = f(V^*)(\frac{n-B}{2}) - (\frac{n-B}{2}),$$

where $a = \frac{(n-B)}{2}$ *and* $b = \frac{-(n-B)}{2}$

3.3.2. Computational Complexity

The computational complexity of the Graph Fragmentation Problem is here established. In Piccini et al. (2016), it is proved that a large set of Node Immunization Problems are at least as hard as the GFP. In Aprile et al. (2018), the hardness of the GFP is proved in a more simple way.

The following problem will be used to characterize the computational complexity of the GFP:

Definition 5 (Minimum Cardinality Vertex Cover)
Instance: simple graph $G = (V, E)$ *and positive integer* k*.*
Does there exist a node-set U *such that* $|U| \leq k$ *and every link is incident to some node from* U*?*

The Minimum Cardinality Vertex Cover belongs to Karp list of 21 $\mathcal{NP}$-Complete decision problems (Karp (1972)). If a vertex cover U is found, there is only one dead in $G' = G - U$ for the GFP. This is clearly a globally optimum solution. The optimality for the GFP is strictly related to the determination of a vertex cover.

Theorem 1 *The GFP belongs to the class of* $\mathcal{NP}$*-Hard problems.*

Proof 2 *The graph* $G' = G - U$ *has isolated nodes if and only if* U *is a vertex cover, where* $|U| \leq B$*. Thus, the GFP is at least as hard as Minimum Cardinality Vertex Cover.*

Besides, in Aprile et al. (2018), it has been proven that there is no approximation algorithm for GFP with factor $\alpha < 5/3$, unless $\mathcal{NP} = \mathcal{P}$.

3.4. Stochastic Weighted Graph Fragementation Problem

The Stochastic Weighted Graph Fragmentation Problem (SWGFP) is a generalization of the GFP. While the GFP considers an unweighted graph, or a graph with nodes having the same weight, the SWGFP introduces weights to every node. The weight w can either be a measure of importance of that node, a measure of its influence on the network, the cost of losing it or any value that is considered appropiate depending on the problem that is being modeled. The SWGFP also assumes that the attack occurs with a certain probability that depends on whether the target infected node is chosen at random or not by the attacker.

3.4.1. Formulation

Given a graph $G = (V, E)$ and a positive integer B called budget. Let $G_i = (V_i, E_i)$, $i = 1, ..., k$ be the connected components where $\sum_{i=1}^{k} |V_i| = |V| = n$. Let $v_{ij} \in V_i$ be the j^{th} node in V_i. Let w_{ij} be the weight of that node and $W_i = \sum_{j=1}^{n_i} w_{ij}$ the total weight of the connected component V_i, where $n_i = |V_i|$. Let p_i be the probability that the infected component is V_i. the score function to minimize is:

$$L(G) = \sum_{i=1}^{k} p_i W_i$$
$$s.t\ |U| = B,$$

where $U \subseteq V$ is the subset of nodes that minimizes the function.

3.4.2. Probability of Infection

Three types of attack should be considered:

(I) *Random Attack*

(II) *Weighted Attack*

(III) *Best Attack*

In the *Random Attack*, a node v is uniformly chosen at random. It assumes that an attacker does not have any information about the targeted network. Under this attack, the expected losses are:

$$L(G)^R = \sum_{i=1}^{k} W_i p_i = \sum_{i=1}^{k} W_i \frac{n_i}{n},$$

where $p_i = \frac{n_i}{n}$.
In the *Weighted Attack*, the model assumes that the attack is deliberate and the attacker has partial information about the network. Under these circumstances, it is reasonable to think that the highest weighted component are more at risk of being attacked. Recalling that the weight is a measure of importance, if the attack is not deliberate, the highest weighted components are those that matter most to protect. Thus, if the attack depends on the components weight, the expected losses after this attack are:

$$L(G)^W = \sum_{i=1}^{k} W_i p_i = \sum_{i=1}^{k} \frac{W_i^2}{W},$$

where a node $v \in V_i$ is attacked with probability $p = \frac{W_i}{W}$, being $W = \sum_{i=1}^{k} W_i$ the total weight of the graph.
In the *Best Attack*, the model assumes that a node is chosen from the component with maximum weight. The attacker is supposed to have full information about the network. The expected loss equals the weight of the maximum weighted component:

$$L(G)^B = W_{max}$$

Chapter 4

Heuristics

In this Chapter, we review the heuristics designed for the Graph Fragmentation Problem and summarize the results of the comparison of these heuristics (Section 4.1). In Section 4.2, Greedy algorithm for SWGFP is explained.

4.1. Heuristics for GFP

In Piccini et al. (2018), three algorithms are presented: Balance, Greedy and GRASP. For completeness, we include a succint description of them. The reader can consult details in the corresponding paper.

4.1.1. Greedy for GFP

The best step is chosen whenever possible. Greedy tries to build the global optimum by means of the best local steps. It iteratively picks the best single node for protection, until the bound $|V^*| = B$ is met. Function $ChooseBestNode$ finds v such that $v = arg\min_w\{Sc(G-w)\}$. A linear search among all nodes $w \in V$ is developed in order to find the best node protection in Greedy.

Algorithm 1 $G_{out} = Greedy(G, B)$

```
1: for $i = 1 : B$ do
2:     $v \leftarrow ChooseBestNode(G)$
3:     $G \leftarrow G - v$
4: end for
5: $G_{out} \leftarrow G$
6: return $G_{out}$
```

4.1.2. Balance for GFP

An alternative algorithm called *Balance* always improves the score in each step as well. *Balance* iteratively picks nodes at random from the largest connected component. Since no score evaluation is required, its computational effort is dominated by *Greedy*.

Algorithm 2 $G_{out} = Balance(G, B)$

```
1: for $i = 1 : B$ do
2:     $V_{max} \leftarrow LargestComponent(G_{out})$
3:     $v \leftarrow ChooseRandom(V_{max})$
4:     $G \leftarrow G - \{v\}$
5: end for
6: $G_{out} \leftarrow G$
7: return $G_{out}$
```

4.1.3. GRASP for GFP

The main algorithm presented in Piccini et al. (2018) is inspired by GRASP methodology. GRASP (Greedy Randomized Adaptive Search Procedure) is a powerful multistart metaheuristic, where feasible solutions are produced in a construction phase and neighbor solutions are explored in a second phase. The key ingredient for diversification is randomization, which determines a pool of different solutions as a result. GRASP involves only two parameters: a parameter $\alpha \in [0, 1]$ in order to build a list of candidate solutions to be included, and the number of iterations $MaxIter$. Then, the algorithm *SelectBest* selects the best k solutions for a post-optimization Path-Relinking process (Glover (1997)). The general GRASP template is throughly explained in (Resende and Ribeiro (2016)). Lines 1-3 jointly combine GRASP, while Lines 4-6 represents our Path-Relinking post-optimization phase. Specifically, Line 2 is a single-run of GRASP,called Alg_GRASP. This function receives an instance (G, B) for the GFP and the level of randomization $\alpha \in [0, 1]$, and produces a solution $G_i = G - U$ for some node-set U with cardinality $|U| = B$. Then, *SelectBest* is introduced, there only the k best solutions of the list $G_1, \ldots, G_{MaxIter}$ are considered for the post-optimization process. This *Pool* of solutions are processed by *Path_Relinking*, and the output is called $Pool_{out}$ (see Line 5). The best solution is selected (Line 6) and returned (Line 7). In the following subsections *Alg_GRASP* and *Path_Relinking* are fully detailed.

Algorithm 3 $G_{out} = Main(G, B, \alpha, MaxIter, k)$

```
1: for $i = 1$ TO $MaxIter$ do
2:     $G_i \leftarrow Alg\_GRASP(G, B, \alpha)$
3: end for
4: $Pool \leftarrow SelectBest(k, G_1, \ldots, G_{MaxIter})$
5: $Pool_{out} \leftarrow Path\_Relinking(Pool)$
6: $G_{out} \leftarrow SelectBest(1, Pool)$
7: return $G_{out}$
```

In this GRASP implementation, a construction phase is first applied (Lines 1-7) and then a Local Search phase takes place (Lines 8-10). The best idea is to immunize exactly B nodes so the **for** loop has precisely B iterations. During each **for** loop (Lines 1-7), a single node is picked and removed. The lowest and highest feasible score reductions are found in Lines 2 and 3. A Restricted Candidate List (RCL) is built in Line 4. Parameter α represents the fraction of all feasible candidates for node-immunization. Since the highest score reduction is desirable, $\alpha = 0$ is *Greedy* strategy while $\alpha = 1$ is a *Random* strategy. In Line 5, a singleton v in chosen uniformly at random among the nodes from the RCL. The node v is deleted from G in Line 6. Line 5 represents the randomization of the GRASP. Finally, the block of Lines 8-10 represents the local search phase. This phase is composed by an elementary *Swap* operation (Line 9), until a local optima is met. *Swap* picks iteratively a single (protected,non-protected) pair, and their roles are exchanged only if the solution is better. The result is a local optima, which is returned in Line 11.

Algorithm 4 $G_{out} = Alg\_GRASP(G, B, \alpha)$

```
1: for $i = 1$ TO $B$ do
2:     $SC_l \leftarrow LowestReduction(G)$
3:     $SC_h \leftarrow HighestReduction(G)$
4:     $RCL \leftarrow \{v : Sc(G - v) \leq SC_l + \alpha(SC_h - SC_l)\}$
5:     $v \leftarrow ChooseRandom(RCL)$
6:     $G \leftarrow G - \{v\}$
7: end for
8: while $Improve(G) = True$ do
9:     $(G, LocalImprove) \leftarrow Swap(G)$
10: end while
11: return $G_{out}$
```

To perform a Path-Relinking post-optimization proccess, a neighborhood structure was defined. Two solutions S_1 and S_2 are neighbors if their symmetric difference is a singleton, i.e. $S_1 \vartriangle S_2 = \{v\}$ for some node v. The movement from S_1 to S_2 is called a *swap*. They define $\mathcal{V}$ as the set of all feasible solutions with identical cardinality $|S| = n - B$. The graph of feasible solutions with minimum cardinality is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set $\mathcal{E}$ consists of all pairs of solutions (S, T) such that $|S \vartriangle T| = 1$. The paths are built using lexicographic order inherited by the natural order $v_1 < v_2 < \ldots < v_n$.

Algorithm 5 $Pool = Path\_Relinking(S_1, S_2, \ldots, S_k)$

```
1: $Pool \leftarrow (S_1, S_2, \ldots, S_k)$
2: $\mathcal{S} \leftarrow \emptyset$
3: for $i = 1$ TO $k - 1$ do
4:     for $j = i + 1$ TO $k$ do
5:         $Path_{(i,j)} \leftarrow Lexicographic(S_i, S_j)$
6:         $S_{(i,j)} \leftarrow SelectBest(1, Path_{(i,j)})$
7:         $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_{(i,j)}\}$
8:     end for
9: end for
10: $Pool \leftarrow SelectBest(k, \mathcal{S})$
11: return $Pool$
```

Relinking receives a pool of k elite solutions and returns another pool of k solutions with better score. New candidate solutions $S_{(i,j)}$ are found for every pair of elite solutions S_i and S_j. The best k solutions are returned.

4.1.4. Heuristics comparison

In Piccini et al. (2018) it was shown that the *Main* heuristic, based on GRASP, outperformed both *Greedy* and *Balance* solutions for all feasible budgets. Even though *Balance* had a reduced computational cost, its performance presented a large gap with respect to *Greedy* heuristic.

4.2. Greedy Heuristic for SWGFP

Here, we present a Greedy algorithm that has been developed to solve the SWGFP. This heuristic choose the best local improvement in each step with the objective of finding good overall solutions. In SWGFP, a Greedy Heuristic was presented for every type of attack.

a. Greedy under Best Attack
   Under this attack, a node is chosen from the component with the highest weight. Given the components of the graph C_1,...,C_k and their weights such as $W_1 \geq ... \geq W_k$, the expected loss is W_1. So the Greedy heuristic picks a node from V_1. If the remaining weight $W_1 - w$ exceeds W_2, a node from V_1 is picked. On the other hand, if the remaining weight W_2 exceeds $W_1 - w$, the best strategy picks a node from V_2. The expected loss is $L(G_v)^B = max\{W_1 - w, W_2\}$.

b. Greedy under Random Attack
   If a node v is uniformly chosen at random with probability $p_i = \frac{n_i}{n}$, where n_i is the component size v belongs to and n is the order of the graph, and assuming $v \in V_1$, the expected loss is:

$$\begin{aligned} L(G_v)^R &= (\frac{n}{n-1}p_1 - \frac{1}{n-1})(W_1 - w) + \sum_{i=2}^{k} \frac{n}{n-1} p_i W_i \\ &= \frac{n}{n-1} L(G)^R - \frac{1}{n-1}(W_1 + w(n_1 - 1)) \end{aligned}$$

   Knowing that the Greedy heuristic iteratively chooses the best node, under the Random Attack, in each step j, v_j is chosen from the remaining graph $G^{(j)}$ until the budget is met:
   $v_j = arg\ max_{u \in G^{(j)}} = \{W_i + (n_i - 1)w(u)\}$
   This is, the chosen node is the one with maximum weight who belongs

to the component whose contribution in terms of weight and size is maximized.

c. Greedy under Weighted Attack
Under this attack, some node is infected with probability $p_i = \frac{W_i}{W}$. Assuming that $v \in V_1$, the expected loss in G^v is:

$$L(G_v)^W = \frac{1}{W-w}(W_1 - w)^2 + \sum_{i=2}^{k} \frac{W_i^2}{W-w},$$

where $W = \sum_{i=1}^{k} W_i$.
In each iteration, the protected node is the one with the maximum weight from the component with the largest weight.

Chapter 5

Contribution

In this chapter, we present two GRASP algorithms for solving SWGFP. We begin by describing the generalities of the GRASP procedures (Section 5.1). The Path-Relinking process as a post-optimization strategy is described in Section 5.2. The first GRASP is described in Section 5.3. The GRASP based on Independent Sets (GRASP-MIS) is described in Section 5.4.

5.1. GRASP

Greedy randomized adaptive search procedures, also known as GRASP, were introduced by Feo and Resende (1989). GRASP is a multi-start metaheuristic for finding approximate solution to diverse combinatorial optimization problems Festa and Resende (2009b). GRASP is based on a construction phase performed in a greedy randomized fashion. The construction phase is also iterative and adaptive, since it builds the solution choosing one element at each iteration and each newly chosen element depends on the previously chosen. Usually, the construction phase is followed by a local improvement phase, where a local optimum is found in the neighbourhood of the incumbent solutions (Feo and Resende (1995); Festa and Resende (2009a,b)). Local search procedures take a feasible solution and improve it by successive modifications until the solution cannot be further improved (Resende and Ribeiro (2016)).

5.2. Path Relinking

Path relinking is an enhancement to the GRASP procedure, leading to significant improvements in solution quality. Path relinking was first introduced in the context of tabu search by Glover and Laguna in 1997 (Glover (1997)). It was suggested as an approach to integrate intensification and diversification in the search for solutions. In the context of GRASP, path relinking was first introduced by Laguna and Martí(1999) (Laguna and Marti (1999)). This approach generates new solution by exploring trajectories between high-quality solutions: it starts from one of these solutions, called the *initiating solution* and generates a path in the neighbourhood space that leads towards other solutions, the *guiding solutions*. The path is created by selecting moves that incorporate attributes of the guiding solution into the initial solution. At each step, the best move (which is the one that best improves or least deteriorates the initial solution), is chosen among the restricted set of moves Aiex et al. (2005). When integrating path relinking to GRASP, two basic strategies are used (Resende and Ribeiro (2005)):

- path relinking is used as a post-optimization step, being applied to all pairs of k solutions in an elite set and,
- path-relinking is applied as an intensification strategy to each local optimum obtained after the local search phase

In this work, we implemented two GRASP consisting of a construction phase followed by a local search and a path relinking procedure applied as a post-optimization strategy.

5.3. GRASP for SWGFP

The first GRASP for solving the SWGFP is based on the GRASP algorithm designed for GFP and introduced in Piccini et al. (2018). The GRASP algorithm for GFP was adapted to solve the SWGFP: nodes weight was incorporated and the score function were replaced with the corresponding SWGFP function, depending on the type of attack being considered. As previously described, in lines 1-3 GRASP (Alg_GRASP) is executed $MaxIter$ times and Lines 4-6 represent the Path-Relinking phase. The function Alg_GRASP receives an instance (G, B) the level of randomization $\alpha \in [0, 1]$, the weights of each node (w), and produces a solution $G_i = G - U$ for some node-set U with cardinality $|U| = B$. Then, the function $SelectBest$ builds a $Pool$ with the k best solutions of the list $G_1, \ldots, G_{MaxIter}$ for the post-optimization process with $Path\_Relinking$. The output is called $Pool_{out}$ (see Line 5). The best solution is selected (Line 6) and returned (Line 7) (Figure 6). In the following subsections Alg_GRASP and $Path\_Relinking$ are fully detailed.

Algorithm 6 $G_{out} = Main(G, B, \alpha, MaxIter, k, w)$

```
1: for $i = 1$ TO $MaxIter$ do
2:     $G_i \leftarrow Alg\_GRASP(G, B, \alpha, w)$
3: end for
4: $Pool \leftarrow SelectBest(k, G_1, \ldots, G_{MaxIter})$
5: $Pool_{out} \leftarrow Path\_Relinking(Pool)$
6: $G_{out} \leftarrow SelectBest(1, Pool)$
7: return $G_{out}$
```

In GRASP, a construction phase is first applied (Lines 1-7) and then a Local Search phase takes place (Lines 8-10). The **for** loop has B iterations. During each **for** loop (Lines 1-7), a single node is picked and removed. The lowest and highest score reductions are determined in Lines 2 and 3. In Line 4 a Restricted Candidate List (RCL) is built. Parameter $\alpha = 0$ is *Greedy* strategy while $\alpha = 1$ is a *Random* strategy. In Line 5, a node v in chosen uniformly at random among the nodes from the RCL. The node v is deleted from G in Line 6. The Lines 8-10 represent the local search phase. This local search is composed by a *Swap* operation (Line 9), until a local optima is met. *Swap* picks iteratively a single (protected,non-protected) pair, and their roles are exchanged only if the solution is better. The result is a local optima, which is returned in Line 11 (Figure 7).

Algorithm 7 $G_{out} = Alg\_GRASP(G, B, \alpha, w)$

```
1: for $i = 1$ TO $B$ do
2:     $SC_l \leftarrow LowestReduction(G)$
3:     $SC_h \leftarrow HighestReduction(G)$
4:     $RCL \leftarrow \{v : Sc(G - v) \leq SC_l + \alpha(SC_h - SC_l)\}$
5:     $v \leftarrow ChooseRandom(RCL)$
6:     $G \leftarrow G - \{v\}$
7: end for
8: while $Improve(G) = True$ do
9:     $(G, LocalImprove) \leftarrow Swap(G)$
10: end while
11: return $G_{out}$
```

5.3.1. Path Relinking

Path relinking was used in this work as a post-optimization strategy. Once the Construction Phase together with the Local Search find $Maxiter$ solutions, the solutions with the lowest scores are chosen to integrate an elite set. Path relinking is performed for every pair of solutions in the pool. In this work, a mixed path relinking process was applied. In mixed path relinking, two paths are simultaneously explored, the first emanating from x_s and the second from x_t, until they meet at an intermediary solution equidistant from x_s and x_t Laguna et al. (2004); Resende and Ribeiro (2005); Ribeiro et al. (2012). It is faster than back-and-forward path relinking and is usually less than twice as long as the backward or forward variants. It was suggested by Glover in Glover (1997) and was first implemented by Resende and Ribeiro (2010), where it was shown to outperform forward, backward, and back-and-forward path-relinking. A template for the mixed path relinking is presented in Figure 5.1. In line 1 and 2, the best solution in the elite set and its score are saved as x_b and $f(x_b)$. The path relinking process is applied to every pair of solutions in the elite set while the stopping criteria are not satisfied. The symetric difference between the pair of solutions is computed in line 5. The current solutions is initialized as x_s in line 6. The loop between line 7 and line 18 determines the best solution at each step. Be $x \oplus l$ the solution x that incorporates attribute l. In line 8, the element l from Δ that minimizes $f(x \oplus l)$ is chosen and the symmetric difference Δ between x and x_t is updated in line 9. The chosen attribute l^* is incorporated into the current solution in line 10. From line 11 to line 14, the test verifies if the new solution x improves the best solution x^b , then x^b

Algorithm 8 *Mixed_path_relinking(eliteset)*

```
1: $x_b \leftarrow$best_sol($eliteset$)
2: $f(x_b) \leftarrow$best_score($eliteset$)
3: while stopping criteria not met do
4:    for (every pair of solutions$(x_s, x_t)$ in eliteset) do
5:       $\Delta \leftarrow \{j = 1, ..., n : x^s_j \neq x^t_j\}$ ;
6:       $x \leftarrow x_s$;
7:       while $|\Delta| > 1$ do
8:          $l^* \leftarrow \text{argmin}\{f(x \oplus l) : l \in \Delta\}$
9:          $\Delta \leftarrow \Delta \backslash \{l^*\}$
10:         $x \leftarrow x \oplus l^*$;
11:         if $f(x) < f(x_b)$ then
12:            $x_b \leftarrow x$;
13:            $f(x_b) \leftarrow f(x)$;
14:         end if
15:         $x' \leftarrow x_t$
16:         $x_t \leftarrow x$
17:         $x \leftarrow x'$
18:      end while
19:   end for
20:   $eliteset \leftarrow$UpdateElite($eliteset, x_b$)
21: end while
22: return $x^b$
```

Figure 5.1. Mixed path relinking

and $f(x^b)$ are updated. The roles of the initiating solution and the guiding solution are swapped in lines 15,16 and 17 and a new iteration starts. If a new best solution is found during the relinking process, the elite set is updated replacing the worst solution in the pool with the new best solution (line 20). The relinking process starts again with the updated elite set, as long as the stopping criteria are not met: either a fixed number of iterations has been reached or no improvement has been found. The path relinking terminates and returns the best overall solution (x_b).

5.4. GRASP-MIS for SWGFP

In this section, we will show how the GRASP and Path-Relinking procedures are combined to obtain an algorithm based on Maximal

Algorithm 9 $GRASP - MIS(G, B, w, Maxiter, \alpha)$

```
1: $eliteset \leftarrow \emptyset$
2: for $i = 1$ to $Maxiter$ do
3:     $\hat{s} \leftarrow$ Construction_Phase($G$,$B$,$w$,$\alpha$);
4:     $s \leftarrow$ Local_Search($\hat{s}$);
5:     $solutions \leftarrow solutions \cup \{s\}$;
6: end for
7: $eliteset \leftarrow$ build_elite_set($solutions$)
8: $best\_sol \leftarrow$ PathRelinking($eliteset$)
9: return $best\_sol$
```

Figure 5.2. Pseudocode of GRASP with path relinking.

Independent Sets (MIS) (Figure 5.2). In order to disrupt graph connectivity and minimize SWGFP score, we propose a GRASP combined with Path Relinking procedure based on the construction of maximal independent sets as a starting point. Graph connectivity can be disrupted through the removal of a vertex cover of that graph, leaving only an independent set(Arulselvan et al. (2009); Addis et al. (2016)). Recalling from Chapter 3, the optimality of the Graph Fragmentation Problem and, for instance, the Stochastic Weighted Graph Fragmentation Problem is strongly related with the finding of a vertex cover. For this reason, we started by finding a maximal independent set using a greedy randomized approach and then, some nodes in the vertex cover were added back to the graph in a greedy fashion, until the vertex cover had exactly B nodes. Unlike previous works, we considered a graph with weighted nodes. In summary, an adaptive, randomized and greedy construction phase was used to generate $Maxiter$ solutions and to perform a local search in each one of them(lines 2 to 6). From this set, the best solutions (the solutions with lower score) were selected to build an elite set (line 7). Then, path relinking was applied between all pair of solutions form this set to obtain the best overall solution (line 8).

5.4.1. Construction phase

A pseudocode for the construction phase is presented in Figure 5.3. Following Feo et al. (1994), this phase starts by finding a maximal independent set(MIS) for a given graph G (lines 1 to 8). This is done in a greedy randomized

Algorithm 10 *Construction Phase*$(G(V, E), w, B, \alpha)$

```
1:  $MIS \leftarrow \emptyset$
2:  while $\mathcal{B} \neq \emptyset$ do
3:      $minweight \leftarrow \min\{\text{weight}(v) : v \in \mathcal{B}\}$
4:      $\mathcal{RCL} = \{v \in \mathcal{B} : \text{weight}(v) \leq min\_weight + \alpha \times (max\_weight - min\_weight)\}$
5:      Select $v* \in \mathcal{RCL}$ at random
6:      $MIS \leftarrow MIS \cup \{v*\}$
7:      $\mathcal{B} = \mathcal{B} \backslash \{v*\} \backslash \{u \in \mathcal{B} : (u, v*) \in E\}$
8:  end while
9:  $S_0 \leftarrow V \backslash MIS$
10: while $|S_0| > B$ do
11:     $x \leftarrow \text{argmin}(\sum \frac{W_i^2}{W}$ : i are connected components in $MIS \cup chosen$ )
12:     $chosen \leftarrow chosen \cup \{x\}$
13:     $S_0 \leftarrow S_0 \backslash \{x\}$
14: end while
15: return $S_0$
```

Figure 5.3. Pseudocode of Construction Phase.

fashion. The minimum weight (min_weight) of the available nodes in each step is computed in line 3. A Restricted Candidate List (RCL) is built with every node whose weight (w) does not exceed $min\_weight + \alpha \times (max\_weight - min\_weight)$ and is not adjacent to the previously chosen nodes (line 4). A node is chosen randomly from the RCL and it is included in the MIS (lines 5 and 6). The set of available nodes is updated by deleting the chosen node and its adjacent nodes (line 7). These steps are repeated until it cannot be found other nodes to be included ($\mathcal{B} = \emptyset$)and the maximal independent set is built. Nodes in the vertex cover obtained after building the MIS, are the initial solution S_0(line 9). In general, S_0 has more than B nodes so it represents an infeasible solution to the SWGFP. However, a feasible solution can be built by removing nodes from S_0 and greedily adding them back to the MIS, until S_0 remains with exactly B nodes. The node chosen at each step (x) is the one that minimizes the SWGFP function when added to the subgraph formed by the MIS and previously chosen nodes (line 11 and 12). The chosen node x is deleted from S_0. It was shown that this strategy is effective for finding critical nodes and outperforms other methods (Addis et al. (2016)).

5.4.2. Local Search

Once a solution S_0 is built in the construction phase, a local search phase is performed on this solution (Figure 5.4). A first-improvement local search strategy was chosen. In a first-improvement strategy the algorithm chooses any neighbour with a better value for objective function. During the neighbourhood search, the first-improving step is performed. At each search position s, the procedure evaluates the neighbouring candidate solutions $s' \in N(s)$ and the first s' that presents a lower value for the objetive function is selected. This kind of search has the advantage of avoiding the time complexity of evaluating all possible neighbours (Hoos and Stützle (2004)). A *2-node-exchange* (or *swap*) neighbourhood is used in this algorithm: a node $v \in S_0$ is exchanged with a node $u \in V \setminus S_0$ and a new score is computed. If the score is lower, the new solution is kept as the best solution and the local search is restarted from this new solution. The local search continues until no further improvement can be found. While improvements can be found, the local search is performed(lines 2 to 16). A node v is removed from S_0, the symetric difference between S_0 and V is computed and stored in *nodes* and the *improve* flag is set as $FALSE$ (lines 3, 4 and 5). A node (from the *nodes* set)is added to S_0 and the score is computed. If the score is lower than the original solution, this is replaced by the new solution; the *improve* flag is set to $TRUE$(lines 7 to 10). If the solution does not improve the current one, the inspected node is removed from *nodes* (line 12) and the search continues through the neighbourhood until it has been completely inspected and no improvement can be done. The best solution S_0 is returned.

5.4.3. Path Relinking

We combined the GRASP based on Independent Sets with the Path Relinking strategy already described in 5.3.1, as a post-optimization process. As previously described, we applied a Mixed Path Relinking on an elite set containing the best solutions selected among the $Maxiter$ solutions delivered by the GRASP. This process if performed for every pair of solutions in the pool. If a new best solution is found, the elite set is updated replacing the worst solution in the set with the new best solution. The relinking process continues as long as the stopping criteria are not met: a number of iterations have been done or no improvement has been found. The path relinking terminates

Algorithm 11 *Local Search*$(G(V,E), w, S_0, iter)$

```
1: $improve \leftarrow TRUE$
2: while (improve) do
3:   $\hat{S}_0 \leftarrow S_0 \setminus \{v\}$
4:   $nodes \leftarrow \Delta(S_0, V)$
5:   $improve \leftarrow FALSE$
6:   while (not improve) and ($nodes \neq \emptyset$) do
7:     $\hat{S}_0 \leftarrow S_0 \cup \{u \in nodes\}$
8:     if $f(\hat{S}_0) < f(S_0)$ then
9:       $S_0 \leftarrow \hat{S}_0$
10:      $improve \leftarrow TRUE$
11:    else
12:      $nodes \leftarrow nodes \setminus \{u\}$
13:    end if
14:  end while
15: end while
16: return $S_0$
```

Figure 5.4. Pseudocode of First-Improvement Local Search Phase.

and returns the best overall solution. The pseudo-code for this Mixed Path Relinking was introduced in Figure 5.1.

Chapter 6

Results

6.1. Computational Results

All computational experiments were conducted using R® language (R Core Team (2013)), on a Home-PC with Intel® Core™ i5-3470U, 3.20GHz, 64-bit OS.

6.1.1. Test problems

We used five graphs as test cases in our experiments. These are:

- Arpanet: The precursor of the Internet, with 20 nodes and 25 links
- Dodecahedron: A famous graph from Graph theory, with 20 nodes and 30 links
- EON: The European Optical Network, with 19 nodes and 36 links
- FON: A real-life Fiber Optic Network with 62 nodes and 126 links (Risso (2014))
- IEEE300: Flow test case from Power Electric Grid (Hines et al. (2010)) with 265 nodes and 373 links.

6.1.2. Experiments

Several experiments were conducted. In the first place, a Brute Force procedure was performed on the following graphs: Arpanet, Dodecahedron, EON and FON. Brute Force lists all the possible combinations of B nodes for each graph ($C_B^n = \frac{n!}{B!(n-B)!}$ combinations) and calculates the score by removing each of those sets. The solution producing the lowest score is considered the global optimum for the SWGFP and the score is recorded. This procedure was repeated with parameter B varying from 1 to 6. This was done for the four mentioned graphs, under the three types of attack: *Random, Weighted* and *Best.* Since this procedure is prohibitive for large instances, Brute Force was not applied on the IEEE300 graph. Next, we ran the Random Algorithm, which picks uniformly at random a solution of B nodes from the graph and calculates the score when the selected set is removed from the graph. This algorithm was ran 10 times for the five graphs (Arpanet, Dodecahedron, EON, FON and IEEE) under the three types of attack: *Random, Weighted* and *Best*, with parameter B varying from 1 to 6. The best solution found among the 10 runs was recorded for each instance. Besides, the average run-time was computed. For graphs Arpanet, Dodecahedron, EON and FON, the average percent deviation from the optimal solution was calculated. For IEEE300 graph, since the optimal score was not known, we calculated the average percent deviation from the best-known solution among all the solutions found by the algorithms. Finally, 10 runs of each GRASP algorithm (GRASP and GRASP-MIS) were performed for the five graphs, under the three types of attacks, varying B from 1 to 6. Average run-time per iteration was calculated. The best score found by each algorithm was recorded after each test. For the first four graphs (Arpanet, Dodecahedron, EON and FON) the average percent deviation from the optimal solution was determined, while for the IEEE300 graph, the average percent deviation from the best-known solution was calculated. Both for GRASP and GRASP-MIS algorithms, $Maxiter = 50$ and the elite set size $k = 5$. After some preliminary experiments, parameter α was set to 0.05 for GRASP and 0.95 for GRASP-MIS. The results are described below.

6.1.3. Random Attack

The optimal scores obtained by Brute Force for Arpanet, Dodecahedron, EON and FON are listed in Table 6.1. Also, Brute Force running-time is detailed in this table. In Table 6.2, the average percent deviation from the optimal solution, the best score found and average run-time per iterarion is presented for algorithms: Random, GRASP and GRASP-MIS. The Random Algorithm, although the faster, performs poorly when compared to GRASP and GRASP-MIS. The Random algorithm found the optimal solution in only 5

out of the 24 tests. Besides, it presented large deviation values for almost all the tests. On the other hand, GRASP found the optimal solution in 20 out of the 24 tests while GRASP-MIS, although employing slight higher computation times, found 23 optimal solutions and presented null or small percent deviation from the optimal scores. GRASP and GRASP-MIS showed null deviation values in 17 cases, GRASP had only 1 deviation value lower than GRASP-MIS while GRASP-MIS had 4 lower deviation values than GRASP. This is more evident in large instances like the FON graph, where nearly all the deviation values obtained by GRASP were higher in comparison with those of GRASP-MIS, indicating that GRASP-MIS is more robust than GRASP.
In Table 6.3, the best-known scores are detailed for each B value. As it can be noted in table 6.4, the best solutions were found by GRASP-MIS. GRASP only found the best value in 1 out of the 6 cases. Again, the Random algorithm is the algorithm that performs worst, with higher scores and larger deviation values. Although GRASP-MIS running times are higher than GRASP, they are still moderate: less than half a minute. GRASP-MIS clearly outperforms GRASP, with lower score and small deviation values under the Random Attack.

Table 6.1. Brute Force solution score for Arpanet, Dodecahedron, EON and FON graphs under Random Attack

Graph	**Budget**	**Optimal Score**	**Time**
Arpanet	1	46	0.07s
	2	22.055	0.22s
	3	12.294	1.49s
	4	7.750	6.54s
	5	6.133	21.91s
	6	4.285	57.22s
Dodecahedron	1	57	0.06s
	2	54.000	0.30s
	3	45.352	1.72s
	4	37.500	7.39s
	5	25.000	20.63s
	6	19.285	51.23s
EON	1	66	0.05s
	2	45.058	0.26s
	3	35.250	1.39s
	4	20.400	5.43s
	5	14.285	14.79s
	6	9.384	38.85s
FON	1	241	0.11s
	2	231.166	2.45s
	3	214.779	50.8s
	4	206.000	11m55s
	5	190.508	2h19m57s
	6	177.714	22h55m06s

Table 6.2. Average % deviation from optimal score, best found score and average run-time for Arpanet, Dodecahedron, EON and FON graphs under Random Attack

Graph	**Budget**	**Random**			**Grasp**			**GraspMIS**		
		%Dev	Best Sc.	Avg Time	%Dev	Best Sc.	Avg Time/It	%Dev	Best Sc.	Avg Time/It
Arpanet	1	3.04	46	0.008s	0	46	0.05s	0	46	0.12s
	2	101.63	40.722	0s	0	22.055	0.07s	0	22.055	0.14s
	3	233.87	34.882	0.005s	0	12.294	0.12s	0	12.294	0.18s
	4	306.04	14.500	0s	0	7.750	0.16s	0	7.750	0.21s
	5	366.30	15.666	0.005s	0	6.133	0.25s	0	6.133	0.22s
	6	350.00	11.357	0.001s	0	4.285	0.22s	0	4.285	0.31s
Dodecahedron	1	0	57	0.009s	0	57	0.05s	0	57	0.15s
	2	0	54	0.003s	0	54	0.11s	0	54	0.18s
	3	11.82	45.352	0.003s	0	45.352	0.15s	0	45.352	0.22s
	4	26.50	42.375	0.002s	0	37.500	0.16s	0	37.500	0.27s
	5	70.56	30.600	0.002s	0	25	0.18s	0	25	0.28s
	6	93.00	21.428	0.003s	0	19.285	0.21s	0	19.285	0.29s
EON	1	2.57	66	0.006s	0	66	0.05s	0	66	0.13s
	2	44.03	61	0.005s	0	45.058	0.11s	2.22	46.058	0.17s
	3	66.41	51.687	0.001s	0	35.250	0.13s	0	35.250	0.18s
	4	156.99	39.400	0s	0	20.400	0.19s	0	20.400	0.20s
	5	252.70	31.285	0.002s	0	14.285	0.16s	0	14.285	0.27s
	6	363.19	36.461	0.003s	0	9.384	0.19s	0	9.384	0.29s
FON	1	2.82	245	0.006s	0	241	0.18s	0	241	1.17s
	2	5.59	239	0.004s	0.37	232.033	0.36s	0	231.166	1.35s
	3	11.22	232.016	0.002s	3.45	222.186	0.46s	0	214.779	1.56s
	4	13.69	227.034	0.004s	0	206	0.53s	0	206	1.72s
	5	20.78	216.193	0.002s	4.13	198.368	0.72s	0	190.508	2.08s
	6	27.05	217.071	0.003s	6.32	188.9464	0.77s	0.30	177.714	2.22s

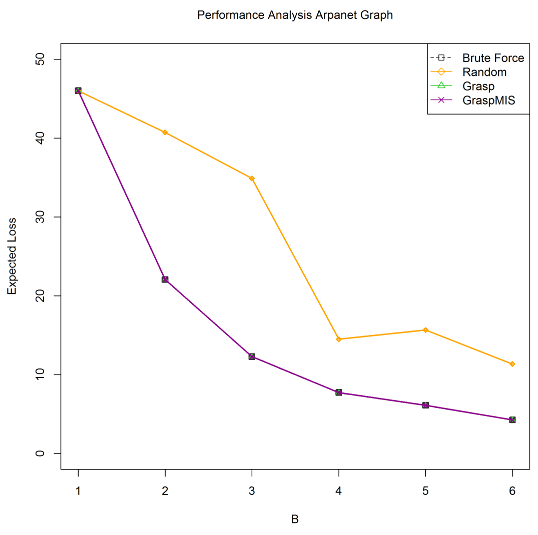
Table 6.3. Best-known score for IEEE graphs

Graph	Budget	Best Score
IEEE	1	646.295
	2	423.570
	3	388.3664
	4	327.639
	5	279.046
	6	232.432

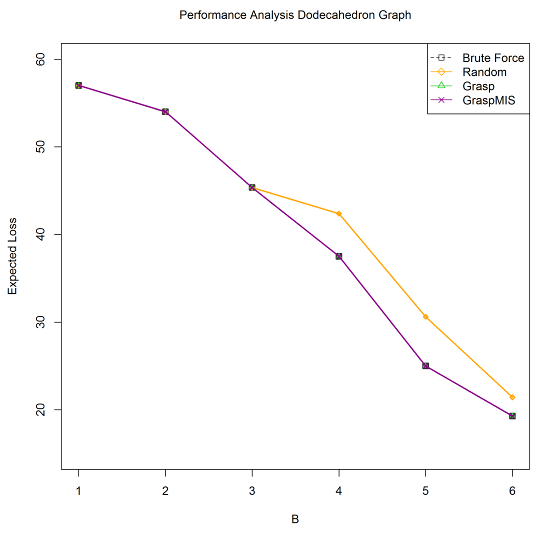
Table 6.4. Average % deviation from best-known score, best found score and average run-time for IEEE graphs under Random Attack

Graph	Budget	Random			Grasp			GraspMIS		
		%Dev	Best Sc.	Avg Time	%Dev	Best Sc.	Avg Time	%Dev	Best Sc.	Avg Time
IEEE	1	14.82	733.409	0.0065s	0	646.295	1.62s	0	646.295	19.15s
	2	74.62	727.616	0.007s	16.04	491.494	1.85s	0	423.570	20.76s
	3	87.37	647.278	0.0045s	0.84	391.633	2.58s	0	388.366	23.92s
	4	122.27	711.199	0.0055s	8.71	356.172	3.50s	0	327.639	23.68s
	5	159.67	635.384	0.004s	23.10	343.507	4.41s	1.63	279.046	23.52s
	6	207.12	624.447	0.0065s	36.89	317.961	5.23s	0.03	232.432	29.08s

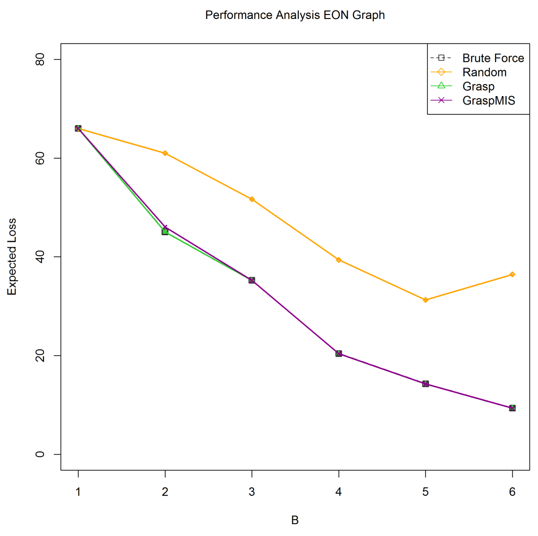
The score evolution for each one of the graphs considered is presented in Figure 6.1. For the three algorithms we represented the best score found by each one of them. A higher score reduction is achieved by both GRASP algorithms when compared with the Random algorithm. From the 30 cases that were analyzed, GRASP-MIS achieves the highest score reduction 29 times; GRASP only achieves the highest score reduction in 21 cases.



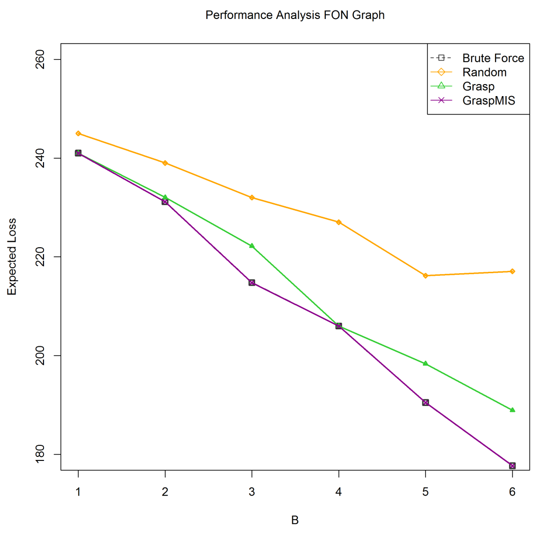
(a) Arpanet Graph



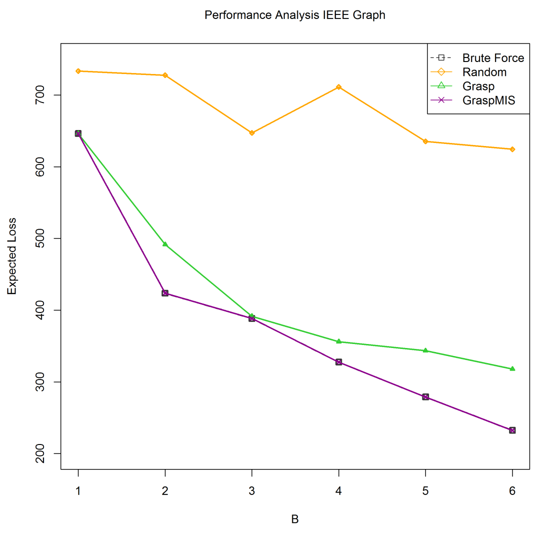
(b) Dodecahedron Graph



(c) EON Graph



(d) FON Graph



(e) IEEE Graph

Figure 6.1. Score evolution under Random Attack

6.1.4. Weighted Attack

The optimal scores obtained by Brute Force for Arpanet, Dodecahedron, EON and FON together with Brute Force running-time are listed in Table 6.5. In Table 6.6, the average percent deviation from the optimal solution, the best score found and average run-time per iterarion is presented for Random, GRASP and GRASP-MIS algorithms. Here, the *Weighted Attack* score function is being considered. The Random Algorithm present fast running-times but is the algorithm that performs worst. This algorithm found only 4 optimal solutions throughout the 24 tests. It also showed large deviation values for almost all the tests. GRASP found the optimal solution in 19 out of the 24 tests while GRASP-MIS, although requiring slight higher computation times, found the optimal solution in all of the 24 tests. GRASP showed null deviation values in 19 tests while GRASP-MIS showed null deviation values in 21 tests with the remaining three deviation values being smaller than GRASPs'. Again, this is more evident in large instances like the FON graph, indicating that GRASP-MIS has greater robustness.
In Table 6.7, the best-known scores for graph IEEE300 are detailed for each B value. In table 6.8, it can be appreciated that the 6 best solutions were found by GRASP-MIS while GRASP only found 2 solutions with the lowest score. The Random algorithm performs worse than the GRASPs, obtaining solutions with higher scores and larger deviation values. As expected, GRASP-MIS running times were higher than GRASP running times, however, they are still moderate and do not surpass the 30 seconds per iteration. Like under the *Random Attack*, GRASP-MIS shows a better performance than GRASP, with lower scores and higher robustness.

Table 6.5. Brute Force solution score for Arpanet, Dodecahedron, EON and FON graphs under Weighted Attack

Graph	**Budget**	**Optimal Score**	**Time**
Arpanet	1	46	0.06s
	2	21.790	0.47s
	3	13.050	3.23s
	4	8.055	16.33s
	5	6.187	67.07s
	6	4.333	251.85s
Dodecahedron	1	57	0.11s
	2	54	0.56s
	3	45.352	3.59s
	4	37.500	10.92s
	5	25.000	40.05s
	6	19.285	138.21s
EON	1	66	0.09s
	2	47.571	0.55s
	3	36.236	3.21s
	4	22.959	14.44
	5	15.311	34.16s
	6	9.789	101.72s
FON	1	244	0.19s
	2	235.033	5.76s
	3	222.479	108s
	4	214.495	56m54s
	5	198.859	14h59m04s
	6	185.663	32h40m47s

Table 6.6. Mean % deviation from optimal score, best found score and average run-time for Arpanet, Dodecahedron, EON and FON graphs under Weighted Attack

Graph	**Budget**	**Random**			**Grasp**			**GraspMIS**		
		%Dev	Best Sc.	Avg Time	%Dev	Best Sc.	Avg Time/It	%Dev	Best Sc.	Avg Time/It
Arpanet	1	2.94	46	0.004s	0	46	0.05s	0	46	0.08s
	2	105.86	41.177	0.002s	0	21.790	0.08s	0	21.790	0.14s
	3	197.87	22.714	0.001s	0	13.050	0.10s	0	13.050	0.17s
	4	274.10	10.722	0.003s	0	8.055	0.16s	0	8.055	0.21s
	5	342.22	7.705	0.003s	0	6.187	0.22s	0	6.187	0.22s
	6	394.81	11.400	0.003s	0	4.333	0.19s	0	4.333	0.31s
Dodecahedron	1	0	57	0.004s	0	57	0.06s	0	57	0.12s
	2	0	54	0.004s	0	54	0.10s	0	54	0.18s
	3	12.45	51	0.002s	0	45.352	0.14s	0	45.352	0.22s
	4	24.25	42.375	0.004s	0	37.500	0.18s	0	37.500	0.26s
	5	75.68	34.600	0.001s	0	25	0.17s	0	25	0.28s
	6	98.56	27	0.003s	0	19.285	0.20s	0	19.285	0.29s
EON	1	3.71	66	0.003s	0	66	0.05s	0	66	0.10s
	2	36.22	61	0.003s	0	47.571	0.10s	0	47.571	0.17s
	3	63.80	45.655	0.003s	0	36.236	0.11s	0	36.236	0.18s
	4	142.83	40.892	0.001s	0	22.959	0.16s	0	22.959	0.20s
	5	224.42	38.063	0.002s	0	15.311	0.12s	0	15.311	0.25s
	6	366.65	20.136	0.002s	0	9.789	0.16s	0	9.789	0.28s
FON	1	1.31	244.008	0.003s	0	244	0.15s	0	244	1.01s
	2	3.73	238.024	0.003s	0.41	236.008	0.35s	0	235.033	1.26s
	3	7.25	231.034	0.002s	2.05	227.034	0.42s	0.74	222.479	1.45s
	4	9.02	226	0.003s	2.12	219.062	0.65s	0.21	214.495	1.61s
	5	16.27	225	0.003s	4.24	207.298	0.69s	1.00	198.859	1.26s
	6	21.92	215.036	0.002s	6.90	198.472	0.68s	0	185.663	1.70s

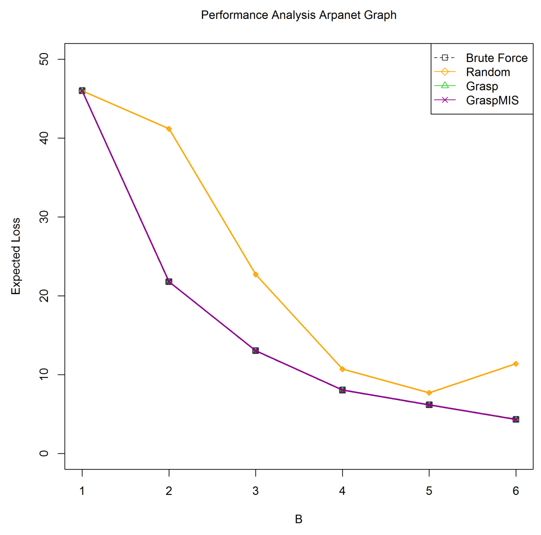
Table 6.7. Best-known score for IEEE graphs under Weighted Attack

Graph	Budget	Best Score
IEEE	1	647.498
	2	423.8525
	3	392.233
	4	333.171
	5	283.016
	6	234.720

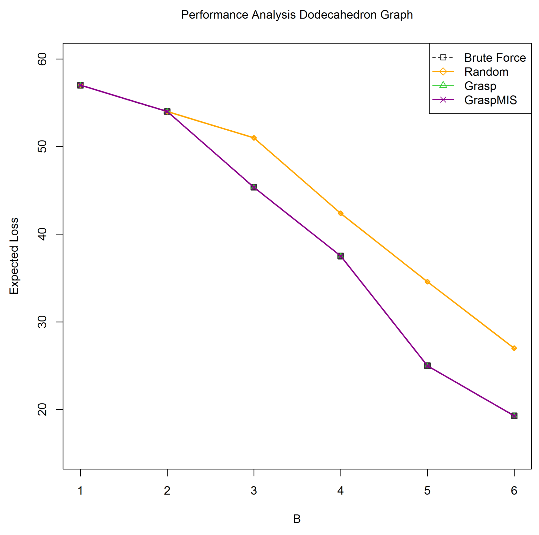
Table 6.8. Average % deviation from best-known score, best found score and average run-time for IEEE graphs under Weighted Attack

Graph	Budget	Random			Grasp			GraspMIS		
		%Dev	Best Sc.	Avg Time	%Dev	Best Sc.	Avg Time/It	%Dev	Best Sc.	Avg Time/It
IEEE	1	14.70	734.035	0.0065s	0	647.498	1.63s	0	647.498	16.24s
	2	74.39	724.116	0.007s	17.39	497.561	1.82s	0	423.852	18.02s
	3	87.52	722.092	0.0045s	0	392.233	2.76s	0	392.233	19.32s
	4	118.32	632.752	0.0055s	8.52	361.557	3.43s	0	333.171	19.90s
	5	157.87	719.133	0.004s	23.54	349.654	4.27s	0	283.016	20.14s
	6	204.31	631.024	0.0065s	43.44	336.677	5.27s	1.41	234.720	24.12s

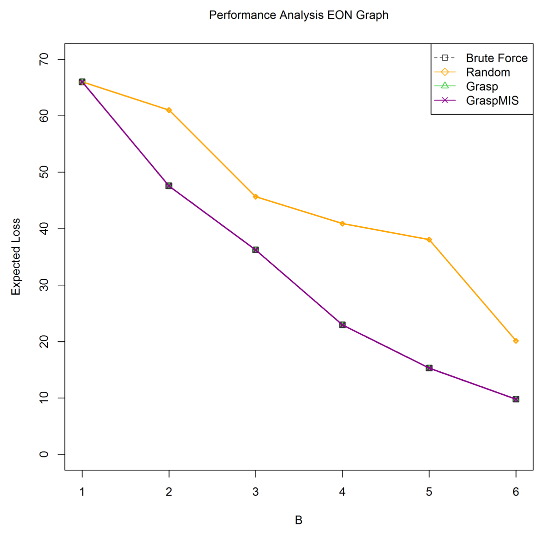
The score evolution for all the graphs is presented in Figure 6.2. The best scores obtained by each algorithm, for each one of the test cases, is represented here. A higher score reduction is achieved by both GRASP algorithms when compared with the Random algorithm. Both GRASP and GRASP-MIS achieve an equal score reduction in 21 cases, with GRASP-MIS outperforming GRASP in the remaining 9 cases by showing higher score reductions.



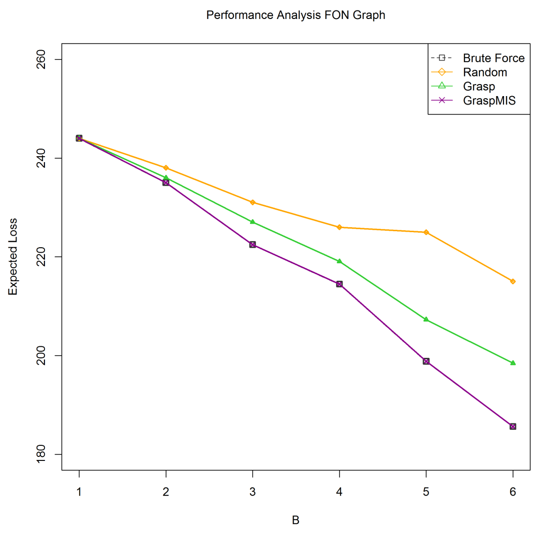
(a) Arpanet Graph



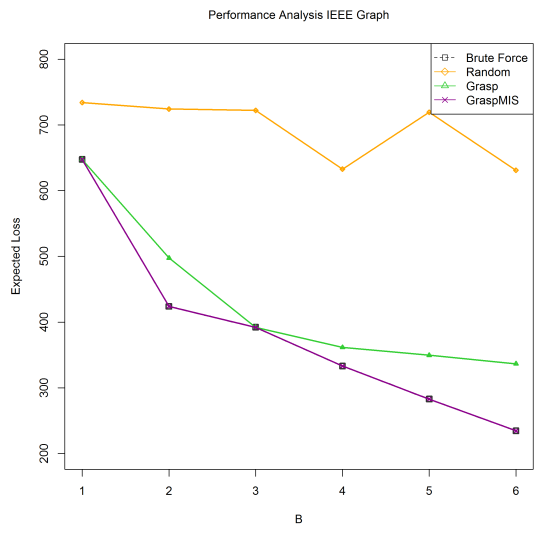
(b) Dodecahedron Graph



(c) EON Graph



(d) FON Graph



(e) IEEE Graph

Figure 6.2. Score evolution under Weighted Attack

6.1.5. Best Attack

Brute Force solutions for Arpanet, Dodecahedron, EON and FON are listed in Table 6.9. Also, Brute Force execution-time is listed in this table. In Table 6.10, the average percent deviation from the optimal solution, the best score found and average run-time per iterarion is presented for: Random, GRASP and GRASP-MIS. The Random Algorithm presents faster execution times. Nevertheless, its performance is poor when compared with both GRASP algorithms. The Random algorithm found only 5 of the the optimal solutions, presenting large deviation values for nearly all the tests. GRASP found the optimal solution in 22 tests while GRASP-MIS found the optimal solution in the 24 tests listed on this table. Both GRASP algorithms perform well for small instances. However, like in the other types of attacks, GRASP-MIS, although requiring slight longer execution times, has a better performance when dealing with larger instances: it is more efficient and more robust. GRASP-MIS presented null deviation values in 22 tests while GRASP presented null values in 20 tests. The remaining 2 deviation values obtained by GRASP-MIS were smaller than the corresponding GRASP deviation values.
In Table 6.11, we present the best-known scores for graph IEEE300 for each B value. In Table 6.12, it can be observed that GRASP-MIS found the 6 best solutions among all the solutions found by the algorithms. GRASP only found 1 solution with the lowest score (for $B = 1$). It must be noted that the Random algorithm found solutions with higher scores and large deviation values from the best-known solutions. Like under the *Random Attack and Weighted Attack*, GRASP-MIS shows a better performance than GRASP, with lower scores and smaller deviation values. Its running-time, although higher than GRASP, is still moderate.

Table 6.9. Brute Force solution score for Arpanet, Dodecahedron, EON and FON graphs under Best Attack

Graph	Budget	Optimal Score	Time
Arpanet	1	46	0.13s
	2	24	1.09s
	3	19	5.64s
	4	11	24.39s
	5	9	77.08s
	6	6	234.47s
Dodecahedron	1	57	0.14s
	2	54	1.63s
	3	48	8.16s
	4	42	30.03s
	5	30	88.18s
	6	21	226.54s
EON	1	66	0.11s
	2	54	1.05s
	3	43	5.75s
	4	27	20.90s
	5	17	60.24s
	6	12	149.68s
FON	1	244	0.09s
	2	237	2.50s
	3	229	51.89s
	4	221	12m40s
	5	212	4h15m07s
	6	200	23h51m12s

Table 6.10. Average % deviation from optimal score and average run-time for Arpanet, Dodecahedron, EON and FON graphs under Best Attack

Graph	Budget	Random			Grasp			GraspMIS		
		%Dev	Best Sc.	Avg Time	%Dev	Best Sc.	Avg Time/It	%Dev	Best Sc.	Avg Time/It
Arpanet	1	3.69	46	0.004s	0	46	0.06s	0	46	0.12s
	2	77.50	27	0.003s	0	24	0.08s	0	24	0.15s
	3	119.47	32	0.001s	0	19	0.12s	0	19	0.17s
	4	190.00	15	0.002s	0	11	0.18s	0	11	0.21s
	5	210.00	15	0.003s	0	9	0.23s	0	9	0.19s
	6	316.66	14	0.003s	5.56	6	0.28s	0	6	0.26s
Dodecahedron	1	0	57	0.005s	0	57	0.07s	0	57	0.15s
	2	0	54	0s	0	54	0.12s	0	54	0.18s
	3	5.93	48	0.002s	0	48	0.16s	0	48	0.22s
	4	14.28	48	0.002s	0	42	0.19s	0	42	0.27s
	5	48.00	42	0.002s	0	30	0.20s	0	30	0.35s
	6	85.71	30	0.002s	0	21	0.25s	0	21	0.32s
EON	1	3.03	66	0.004s	0	66	0.06s	0	66	0.13s
	2	20.74	63	0s	0	54	0.10s	0	54	0.19s
	3	40.23	54	0.003s	0	43	0.15s	0	43	0.19s
	4	105.92	52	0.002s	0	27	0.16s	0	27	0.25s
	5	202.35	42	0.003s	0	17	0.14s	0	17	0.23s
	6	291.66	41	0.003s	0	12	0.16s	0	12	0.24s
FON	1	1.47	245	0.001s	0	244	0.15s	0	244	1.16s
	2	2.78	240	0.003s	0	237	0.43s	0	237	1.26s
	3	4.71	235	0.003s	0	229	0.59s	0	229	1.40s
	4	6.47	231	0.001s	0.27	221	0.81s	0	221	1.47s
	5	9.85	230	0.004s	0.47	213	0.97s	0.37	212	1.08s
	6	13.75	218	0.003s	2.00	204	0.83s	0.60	200	1.78s

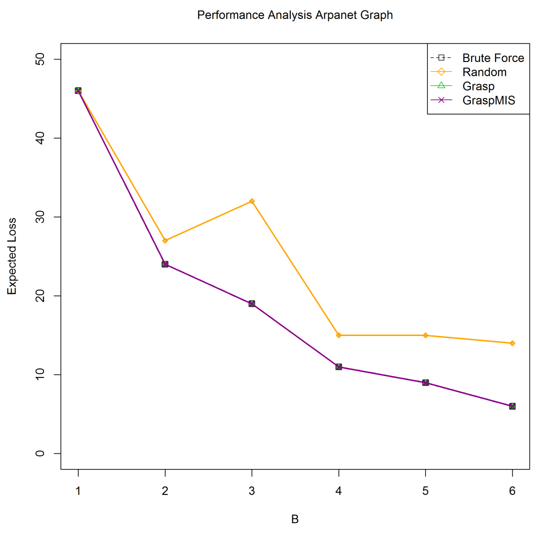
Table 6.11. Best-known score for IEEE graphs under Best Attack

Graph	Budget	Best Score
IEEE	1	690
	2	513
	3	483
	4	403
	5	367
	6	239

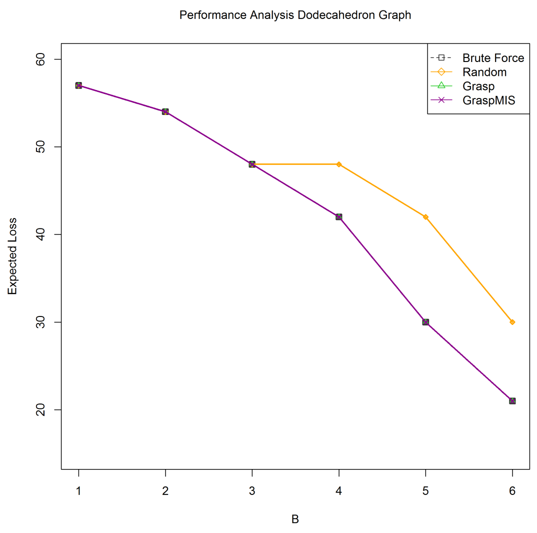
Table 6.12. Average % deviation from best-known score and average run-time for IEEE graphs under Best Attack

Graph	Budget	Random			Grasp			GraspMIS		
		%Dev	Best Sc.	Avg Time	%Dev	Best Sc.	Avg Time	%Dev	Best Sc.	Avg Time
IEEE	1	7.72	737	0.004s	0	690	1.57s	0	690	18.96s
	2	44.35	735	0.003s	14.81	589	1.79s	0	513	20.87s
	3	52.18	728	0.001s	15.07	519	2.49s	0	483	21.79s
	4	82.17	728	0.002s	23.97	489	3.35s	0	403	21.86s
	5	98.44	702	0.003s	30.84	479	5.42s	0	367	22.00s
	6	202.65	701	0.003s	95.23	466	4.79s	1.33	239	25.24s

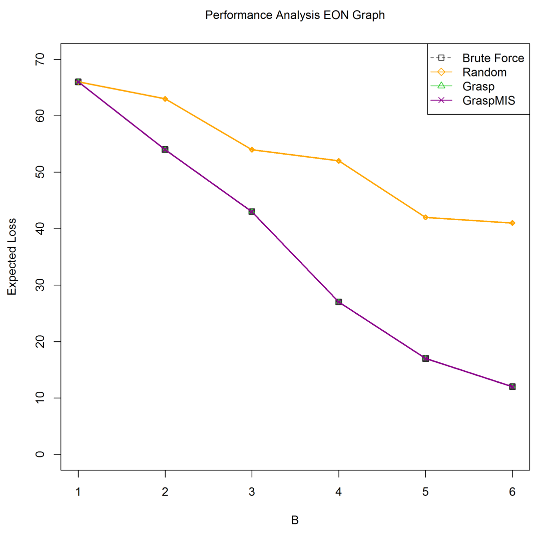
The score evolution for all the graphs is presented in Figure 6.3. The best scores from each algorithm are represented here. The higher score reduction is achieved by GRASP-MIS, achieving the optimum(or the best-known score) 30 times, followed by GRASP, which achieves the best values in 23 cases. Random algorithm is the one that performs worse.



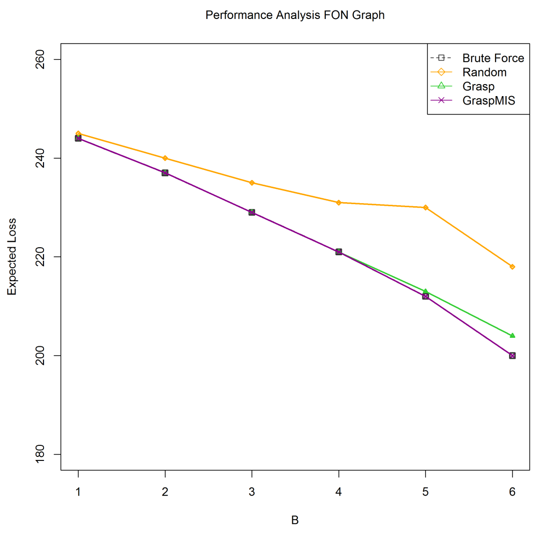
(a) Arpanet Graph



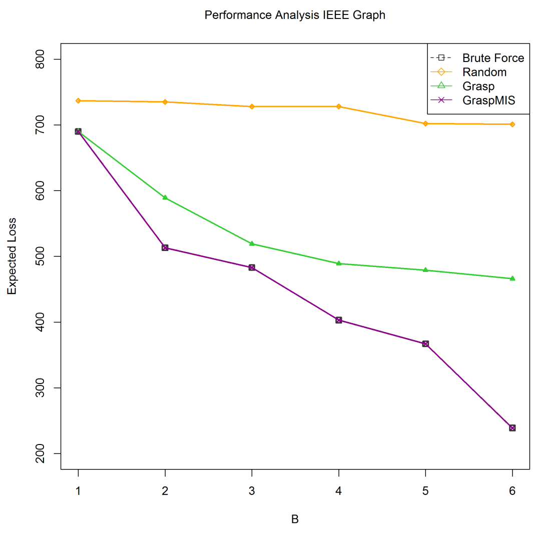
(b) Dodecahedron Graph



(c) EON Graph



(d) FON Graph

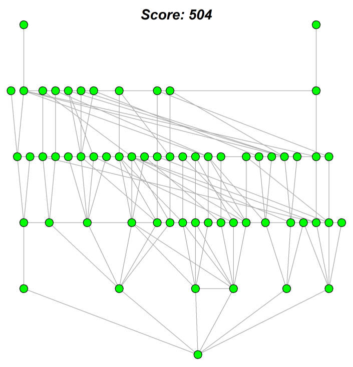


(e) IEEE Graph

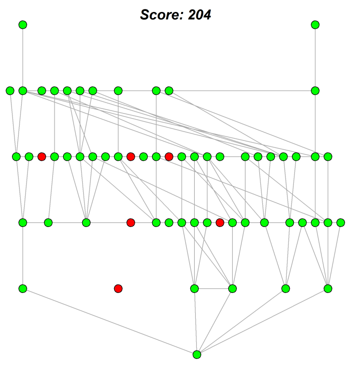
Figure 6.3. Score evolution under Best Attack

6.1.6. Graph Topology

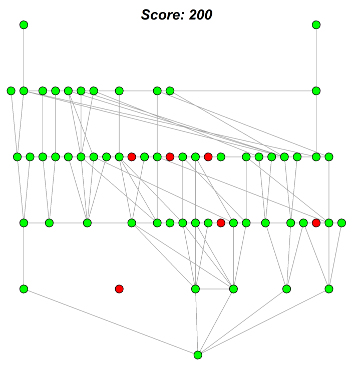
In the context of SWGFP, a higher score reduction implies a greater graph fragmentation. This translates into a lower expected loss when a singleton is infected. To illustrate this idea, Figure 6.4 shows the FON graph topology before node immunization (6.4a); FON topology after immunization with nodes selected using GRASP (6.4b) and FON topology after immunization with nodes selected using GRASP-MIS (6.4c). Immunized nodes are represented in red. This example corresponds to the *Best Attack* for $B = 6$.



(a) Before node immunization



(b) After node immunization with GRASP



(c) After node immunization with GRASP-MIS

Figure 6.4. FON graph topology: before and after node immunization

Chapter 7

Conclusions

In this thesis we introduce for the first time the Stochastic Weighted Graph Fragmentation Problem (SWGFP). This combinatorial optimization problem is a generalization of the Graph Fragmentation Problem (GFP). The GFP aims to identify a set of non-weighted nodes whose immunization, or removal, might help to fragment the graph, therefore, mitigating contagion spread in a network and minimizing the expected loss when a singleton is infected. The Critical Node Problem (CNP) is also mentioned in this thesis. The main goal of this optimization problem is to identify a set of nodes whose removal would minimize pairwise connectivity in the residual graph. This problem has great importance in several fields and has been widely studied in literature. Even though GFP was developed by a local team of researchers and the CNP was developed by a foreign group of researchers, an equivalence between both combinatorial optimization problems is established in this thesis. A demonstration for this equivalence is given in Chapter 3. This implies that knowledge of CNP and knowledge of GFP can be used together in order to fully understand connectivity properties in a graph. This is essential to take adequate measures to ensure network survivability and to minimize negative consequences when a disaster occurs.

The SWGFP introduced in this thesis incorporates, as a novelty, weights to the nodes. These weights reflect the node importance or the cost of losing it. It must be highlighted that several studies have been conducted on critical nodes in vertex-non-weighted graphs, however, research on critical nodes in vertex-weighted graphs is scarce. Considering weighted nodes is important since different nodes usually do not have the same relevance in a network.

Another novelty introduced by the SWGFP is the existence of three probability laws under which an attacker performs its attack on the network with the defender having full knowledge of the probability law. The three probability laws differ in the amount of knowledge about the network the attacker relies on. Under the *Random Attack* the attacker is assumed to have null information, under the *Weighted Attack* the attacker is assumed to have partial information and under the *Best Attack* the attacker is assumed to have full information. Taking into account nodes' weights and anticipating the type of attack that will be performed on the network contributes to achieve a better protection of the network prioritizing the highest-weighted nodes whether they should be removed or maintained.

From a methodological point of view, two GRASP enriched with Path-relinking have been proposed to solve the SWGFP in this thesis. The first GRASP was based on a previously designed GRASP that had been used to solve the Graph Fragmentation Problem; this was adapted to solve the SWGFP under the three possible types of attacks. A second hybrid GRASP was developed based on the construction of Independent Sets (GRASP-MIS). This concept that had been proposed by Arulselvan et al. (2009) to find critical nodes, however, it has not been explored in the context of graphs with weighted nodes and, in particular, in the SWGFP. As it can be appreciated in Chapter 6, computational experiments showed that the latter outperforms the former in terms of the quality of solutions and the average percent deviation from the scores of the best-known solutions. In terms of the running-times, before performing the computational experiments, the difference between both algorithms was expected: GRASP performs a Greedy phase iterating it B times to find the best B nodes to immunize while GRASP-MIS iterates the Greedy phase $n-m-B$ times, with n being the order of the graph, m the size of the maximal independent set and B the budget, in order to obtain a vertex cover of size B. Although the running-times required by the GRASP based on Independent Sets (GRASP-MIS) are slightly higher than those required by the simple GRASP, the average times are still moderate. For all these reasons, the newly proposed GRASP-MIS based on Independent Sets has proved to be an effective tool to solve the SWGFP under the three types of attacks: *Random, Weighted* and *Best*. This is helpful to the design of efficient protection strategies for the network in face of disasters. Our hypothesis about the reason why the algorithm based on Independent Sets outperforms the classical GRASP is that, since the SWGFP aims to maximally fragment the graph while minimizing the expected loss, a set of low-weighted isolated nodes (or Independent Set) provides the maximum fragmentation and minimum expected loss for that graph leaving the remaining nodes (a Vertex Cover) as an optimal solution for the problem. Nevertheless, the SWGFP imposes a restriction on the solution size which must be exactly B. If the Vertex Cover size equals B, the optimal solution has been found. If not, the exceeding nodes must be removed from the Vertex Cover and returned to the graph in a way that the SWGFP score function is minimized.

Finally, it must be highlighted that accurately identifying nodes which play a key role in graph connectivity is essential to prevent catastrophic consequences after an infection or an attack on a network and thus ensuring network survivability. As future work, we would like to further explore the importance of Independent Sets in the context of critical node identification, and in particular, critical weighted nodes. The need to protect nodes which have a great influence on network connectivity at the same time that they have high weight so as to minimize expected loss, presents us the challenge of identifying an appropriate initial Independent Set for the algorithm to be applied on. We believe that additional studies should be made in order to address this issue.

Bibliography

Addis, B., Aringhieri, R., Grosso, A., and Hosteins, P. (2016). Hybrid constructive heuristics for the critical node problem. *Annals of Operations Research*, 238(1-2):637–649.

Aiex, R. M., Resende, M. G. C., Pardalos, P. M., and Toraldo, G. (2005). GRASP with Path Relinking for Three-Index Assignment. *INFORMS Journal on Computing*, 17(2):224–247.

Aprile, M., Castro, N., Ferreira, G., Piccini, J., Robledo, F., and Romero, P. (2018). Graph fragmentation problem: analysis and synthesis. *International Transactions in Operational Research*, 26(1):41–53.

Aprile, M., Castro, N., Robledo, F., and Romero, P. (2017). Analysis of node-resilience strategies under natural disasters. In *International Conference on Design of Reliable Communication Networks*, pages 93–100.

Aringhieri, R., Grosso, A., Hosteins, P., and Scatamacchia, R. (2016a). A general evolutionary framework for different classes of critical node problems. *Engineering Applications of Artificial Intelligence*, 55:128–145.

Aringhieri, R., Grosso, A., Hosteins, P., and Scatamacchia, R. (2016b). Local search metaheuristics for the critical node problem. *Networks*, 67(3):209–221.

Arulselvan, A., Commander, C. W., Elefteriadou, L., and Pardalos, P. M. (2009). Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200.

Arulselvan, A., Commander, C. W., Pardalos, P. M., and Shylo, O. (2007). Managing network risk via critical node identification. In Gulpinar, N.

and Rustem, B., editors, *Risk Management in Telecommunication Networks*. Springer.

Arulselvan, A., Commander, C. W., Shylo, O., and Pardalos, P. M. (2011). Cardinality-constrained critical node detection problem. In Gülpınar, N., Harrison, P., and Rüstem, B., editors, *Performance models and risk management in communications systems*, pages 79–91. Springer New York.

Barabási, A.-L. and Bonabeau, E. (2003). Scale-free networks. *Scientific american*, 288(5):60–69.

Bavelas, A. (1948). A mathematical model for group structures. *Applied anthropology*, 7(3):16–30.

Bavelas, A. (1950). Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730.

Billur Engin, H., Gursoy, A., Nussinov, R., and Keskin, O. (2014). Network-based strategies can help mono-and poly-pharmacology drug discovery: a systems biology view. *Current pharmaceutical design*, 20(8):1201–1207.

Boginski, V. and Commander, C. W. (2009). Identifying critical nodes in protein-protein interaction networks. In *Clustering challenges in biological networks*, pages 153–167. World Scientific.

Borgatti, S. P. (2005). Centrality and network flow. *Social networks*, 27(1):55–71.

Borgatti, S. P. (2006). Identifying sets of key players in a social network. *Computational & Mathematical Organization Theory*, 12(1):21–34.

Borgatti, S. P. and Everett, M. G. (2006). A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484.

Castro, N. (2018). Análisis y estudio de complejidad del problema de fragmentación de grafos. Master's thesis, Universidad de la República, Montevideo, Uruguay.

Chen, D., Lü, L., Shang, M.-S., Zhang, Y.-C., and Zhou, T. (2012). Identifying influential nodes in complex networks. *Physica a: Statistical mechanics and its applications*, 391(4):1777–1787.

Chen, D.-B., Gao, H., Lü, L., and Zhou, T. (2013). Identifying influential nodes in large-scale directed networks: the role of clustering. *PloS one*, 8(10):1–10.

Crucitti, P., Latora, V., Marchiori, M., and Rapisarda, A. (2004). Error and attack tolerance of complex networks. *Physica A: Statistical mechanics and its applications*, 340(1-3):388–394.

Dasgupta, S., Papadimitriou, C. H., and Vazirani, U. (2008). *Algorithms.* McGraw-Hill, Inc.

Data, S. and Wang, H. (2005). The effectiveness of vaccinations on the spread of email-borne computer viruses. In *Proceedings of the 18th Canadian Conference on Electrical and Computer Engineering*, pages 219–223. IEEE.

del Rey, A. M. (2015). Mathematical modeling of the propagation of malware: a review. *Security and Communication Networks*, 8(15):2561–2579.

Di Summa, M., Grosso, A., and Locatelli, M. (2011). Complexity of the critical node problem over trees. *Computers & Operations Research*, 38(12):1766–1774.

Dinh, T. N., Xuan, Y., Thai, M. T., Pardalos, P. M., and Znati, T. (2012). On new approaches of assessing network vulnerability: hardness and approximation. *IEEE/ACM Transactions on Networking*, 20(2):609–619.

Dinh, T. N., Xuan, Y., Thai, M. T., Park, E., and Znati, T. (2010). On approximation of new optimization methods for assessing network vulnerability. In *INFOCOM*, volume 2010, pages 1–9.

Estrada, E. and Bodin, Ö. (2008). Using network centrality measures to manage landscape connectivity. *Ecological Applications*, 18(7):1810–1825.

Feo, T. A. and Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

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

Ferreira, G. (2018). *Analysis and Optimization of Highly Reliable Systems.* PhD thesis, Universidad de la República, Montevideo, Uruguay.

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

Festa, P. and Resende, M. G. (2009b). An annotated bibliography of GRASP–Part II: Applications. *International Transactions in Operational Research*, 16(2):131–172.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

Garey, M. R. (1979). A Guide to the Theory of NP-Completeness. *Computers and intractability.*

Glover, F. (1997). Tabu search and adaptive memory programming—advances, applications and challenges. In *Interfaces in computer science and operations research*, pages 1–75. Springer.

Grubesic, T. H., Matisziw, T. C., Murray, A. T., and Snediker, D. (2008). Comparative approaches for assessing network vulnerability. *International Regional Science Review*, 31(1):88–112.

Guimera, R., Mossa, S., Turtschi, A., and Amaral, L. N. (2005). The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. *Proceedings of the National Academy of Sciences*, 102(22):7794–7799.

Gupta, N., Singh, A., and Cherifi, H. (2015). Community-based immunization strategies for epidemic control. In *Proceedings of the 7th International Conference onCommunication Systems and Networks (COMSNETS 2015)*, pages 1–6. IEEE.

Harary, F. (1969). *Graph theory. 1969.* Addison-Wesley, Reading, MA.

Hines, P., Blumsack, S., Sanchez, E. C., and Barrows, C. (2010). The topological and electrical structure of power grids. In *Proceedings of the 43rd Hawaii International Conference onSystem Sciences (HICSS 2010)*, pages 1–10.

Hoos, H. H. and Stützle, T. (2004). *Stochastic local search: Foundations and applications*. Elsevier.

Iyer, S., Killingback, T., Sundaram, B., and Wang, Z. (2013). Attack robustness and centrality of complex networks. *PloS one*, 8(4):1–17.

Joyce, K. E., Laurienti, P. J., Burdette, J. H., and Hayasaka, S. (2010). A new measure of centrality for brain networks. *PLoS One*, 5(8):1–13.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Kempe, D., Kleinberg, J., and Tardos, É. (2005). Influential nodes in a diffusion model for social networks. In *International Colloquium on Automata, Languages, and Programming*, pages 1127–1138. Springer.

Kephart, J. O., White, S. R., and Chess, D. M. (1993). Computers and epidemiology. *IEEE Spectrum*, 30(5):20–26.

Kermark, M. and Mckendrick, A. (1927). Contributions to the mathematical theory of epidemics. part i. *Proceedings of the Royal Society of London*, 115(772):700–721.

Laguna, M., Glover, F., and Mart, R. (2004). Scatter search and path relinking: Foundations and advanced designs. *New Optimization Techniques in Engineering, editor G. Onwubolu, Springer-Verlag.*

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

Lalou, M., Tahraoui, M. A., and Kheddouci, H. (2018). The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92–117.

Latora, V. and Marchiori, M. (2004). How the science of complex networks can help developing strategies against terrorism. *Chaos, solitons & fractals*, 20(1):69–75.

Lloyd, A. L. and May, R. M. (2001). How viruses spread among computers and people. *Science*, 292(5520):1316–1317.

Malliaros, F. D., Rossi, M.-E. G., and Vazirgiannis, M. (2016). Locating influential nodes in complex networks. *Scientific reports*, 6:19307.

Newman, M. E. (2005). A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54.

Newman, M. E. J. (2016). *Mathematics of Networks*, pages 1–8. Palgrave Macmillan UK, London.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity.* Courier Corporation.

Piccini, J. (2006). *Static Reliability and Resilience in Dynamic Systems.* PhD thesis, Universidad de la República, Montevideo, Uruguay.

Piccini, J., Robledo, F., and Romero, P. (2015). Node-immunization strategies in a stochastic epidemic model. In *International Workshop on Machine Learning, Optimization and Big Data*, pages 222–232. Springer.

Piccini, J., Robledo, F., and Romero, P. (2016). Graph fragmentation problem. In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems*, pages 137–144.

Piccini, J., Robledo, F., and Romero, P. (2018). Complexity among combinatorial problems from epidemics. *International Transactions in Operational Research*, 25(1):295–318.

Piraveenan, M., Prokopenko, M., and Hossain, L. (2013). Percolation centrality: Quantifying graph-theoretic impact of nodes during percolation in networks. *PloS one*, 8(1):1–14.

Power, J. D., Schlaggar, B. L., Lessov-Schlaggar, C. N., and Petersen, S. E. (2013). Evidence for hubs in human functional brain networks. *Neuron*, 79(4):798–813.

Purevsuren, D., Cui, G., Win, N. N. H., and Wang, X. (2016). Heuristic algorithm for identifying critical nodes in graphs. *Advances in Computer Science: an International Journal*, 5(3):1–4.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Resende, M. G. and Ribeiro, C. C. (2005). GRASP with path-relinking: Recent advances and applications. In *Metaheuristics: progress as real problem solvers*, pages 29–63. Springer.

Resende, M. G. and Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of metaheuristics*, pages 283–319. Springer.

Resende, M. G. and Ribeiro, C. C. (2016). *Optimization by GRASP*. Springer.

Restrepo, J. G., Ott, E., and Hunt, B. R. (2006). Characterizing the dynamical importance of network nodes and links. *Physical review letters*, 97(9):094102.

Ribeiro, C. C., Rosseti, I., and Vallejos, R. (2012). Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms. *Journal of Global Optimization*, 54(2):405–429.

Risso, C. (2014). *Análisis de Propiedades en Redes Multi-overlay*. PhD thesis, Engineering School, University of the Republic, Uruguay.

Shams, B. and Khansari, M. (2014). Using network properties to evaluate targeted immunization algorithms. *Network Biology*, 4(3):74–95.

Shen, S. and Smith, J. C. (2012). Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. *Networks*, 60(2):103–119.

Sipser, M. (1996). *Introduction to the Theory of Computation*. International Thomson Publishing.

Soria, M., Lordan, O., and SALLAN, J. M. (2017). Heuristics of node selection criteria to assess robustness of world airport network. *Chinese Journal of Aeronautics*, 30(4):1473–1480.

Ventresca, M. and Aleman, D. (2013). Evaluation of strategies to mitigate contagion spread using social network characteristics. *Social Networks*, 35(1):75–88.

Ventresca, M. and Aleman, D. (2015). Efficiently identifying critical nodes in large complex networks. *Computational Social Networks*, 2(1):6.

Wei, D., Deng, X., Zhang, X., Deng, Y., and Mahadevan, S. (2013). Identifying influential nodes in weighted networks based on evidence theory. *Physica A: Statistical Mechanics and its Applications*, 392(10):2564–2575.

Zhao, L., Park, K., Lai, Y.-C., and Ye, N. (2005). Tolerance of scale-free networks against attack-induced cascades. *Physical Review E*, 72(2):025104.

Zhou, Y. and Hao, J.-K. (2017). A fast heuristic algorithm for the critical node problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 121–122. ACM.