UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Optimizing Reliable Networks with Node-Connectivity Constraints and Failure Conditions on Nodes and Edges

TO BE DEFENDED

Nelson Sebastián Laborde Castillo

Programa de Posgrado en Informática
Facultad de Ingeniería, PEDECIBA - Informática
Universidad de la República

Montevideo – Uruguay
Mayo de 2020

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Optimizing Reliable Networks with Node-Connectivity Constraints and Failure Conditions on Nodes and Edges

TO BE DEFENDED

Nelson Sebastián Laborde Castillo

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

Director de tesis:
Dr. Ing. Franco Robledo

Director académico:
Prof. Ing. Omar Viera

Montevideo – Uruguay
Mayo de 2020

Laborde Castillo, Nelson Sebastián

Optimizing Reliable Networks with Node-Connectivity Constraints and Failure Conditions on Nodes and Edges / Nelson Sebastián Laborde Castillo. - Montevideo: Universidad de la República, Facultad de Ingeniería, PEDECIBA - Informática, 2020.

XIII, 130 p.: il.; 29,7cm.

Director de tesis:

Franco Robledo

Director académico:

Omar Viera

Tesis de Maestría – Universidad de la República, Programa de Informática, 2020.

Referencias bibliográficas: p. 84 – 95.

1. Topological Network Design, 2. Network Reliability, 3. Simulation, 4. Network Optimization, 5. Backbone, 6. RVR, 7. Metaheuristics, 8. VNS. I. Robledo, Franco. II. Universidad de la República, Programa de Posgrado en Informática. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Dr. Prof. Víctor Albornoz

Dr. Prof. Leslie Murray

Dr. Prof. Sebastián Basterrech

Montevideo – Uruguay
Mayo de 2020

(Dedicatoria) A mis Padres
Ricardo y Rosario, a mi
Hermano Santiago y a mi
Familia.

Agradecimentos

Quisiera agradecer: A mis tutores, Dr. Franco Robledo, por el apoyo, guía y gran paciencia durante todo el proceso de desarrollo de esta Tesis. Al Dr. Pablo Romero por su ayuda, intercambio de ideas y revisión de este documento. A mi amigo y colega Ing. Julio Cesano por alentarme a inscribirme junto a él, en el programa de Maestría PEDECIBA. Un agradecimiento muy especial a mi familia, por todo el apoyo, comprensión y motivación que me dieron durante todo el desarrollo de este trabajo, el cuál está enteramente dedicado a ellos.

RESUMEN

En una red las entidades relevantes son nodos y conexiones entre nodos, y en general el principal objetivo buscado es lograr una comunicación segura entre nodos de esta red, ya sea para redes telefónicas y de comunicación de datos, de transporte, arquitectura de computadores, redes de energía eléctrica o sistemas de comando y control. La optimización relativa al costo de una red y la confiabilidad de la misma, relacionada con la supervivencia de esta, son los criterios predominantes en la selección de una solución para la mayor parte de los contextos. Un tema interesante que ha atraído un gran esfuerzo es cómo diseñar topologías de red, con un uso mínimo de recursos de red en términos de costo que brinde una garantía de confiabilidad. A pesar que por años el costo ha sido el factor primario, la confiabilidad ha ganado rápidamente en relevancia. Con sistemas de transmisión de fibra óptica de alta capacidad formando los backbones de la mayoría de las redes actuales y junto con el rápido desarrollo de la tecnología de comunicación de redes y el crecimiento explosivo de las aplicaciones de Internet, la confiabilidad de la red parece cada vez más importante, tanto para áreas tradicionales como la industria de defensa, finanzas y energía, y áreas emergentes como la computación confiable, la computación en la nube, internet de las cosas (IoT) y la próxima generación de Internet, la supervivencia del tráfico por sobre los fallos de red se ha convertido aún en más crítica. En ese sentido podemos diferenciar, a grandes rasgos, dos de los principales problemas a resolver en el análisis y diseño de topologías de red. Primeramente la obtención de una red óptima en algún sentido, siendo este definido por ejemplo mediante la obtención de la máxima cantidad posible de caminos disjuntos entre pares de nodos, esto sujeto a determinadas restricciones definidas según el contexto. El segundo problema es la evaluación de la confiabilidad de la red en función de las confiabilidades elementales de los nodos y conexiones entre nodos que componen la red. Estas confiabilidades elementales son probabilidades de operación asociadas a los nodos y conexiones entre nodos. Ambos problemas están fuertemente relacionados, pudiendo tener que comparar en el proceso de búsqueda de redes óptimas la confiabilidad entre soluciones candidatas, o

luego de obtener una solución candidata tener que evaluar la confiabilidad de la misma y de esta forma descartarla o no. El presente trabajo se centra en la resolución del problema enfocado en ambos puntos planteados. Para ello modelamos el problema de diseño de la topología de red sobre la base de un modelo definido como GSP-NCHR (Generalized Steiner Problem with Node-Connectivity Constraints and Hostile Reliability) extensión del más conocido GSP (Generalized Steiner Problem). El presente problema es NP-duro, se dedicó un capítulo para presentar resultados teóricos que lo demuestran. Nuestro objetivo es atacar de forma aproximada el modelo GSP-NCHR de tal modo de poder resolver la optimización de la red y luego medir la confiabilidad de la solución obtenida. Para ello optamos por desarrollar la metaherística VNS (Variable Neighborhood Search). VNS es un método potente que combina el uso de búsquedas locales basadas en distintas definiciones de vecindad, el cual ha sido utilizado para obtener soluciones de buena calidad en distintos problemas de optimización combinatoria. En lo referente al cálculo de confiabilidad de la red, nuestro modelo hostil pertenece a la clase $\mathcal{NP}$-duro, por eso desarrollamos RVR (Recursive Variance Reduction) como método de simulación, ya que la evaluación exacta de esta medida para redes de tamaño considerable es impracticable. Las pruebas experimentales fueron realizadas utilizando un conjunto amplio de casos de prueba adaptados de la TSPLIB, de heterogéneas topologías con diferentes características, incluyendo instancias de hasta 400 nodos. Los resultados obtenidos indican tiempos de cómputo altamente aceptables acompañados de óptimos locales de buena calidad.

Palabras Clave: Diseño Topológico de Redes, Confiabilidad de Redes, Simulación, Optimización de Redes, Red Dorsal, RVR, Metaheurísticas, VNS.

ABSTRACT

The relevant entities in a network are its nodes, and the links between them. In general, the goal is to achieve a reliable communication between different pairs of nodes. Examples of applications are telephonic services, data communication, transportation systems, computer systems, electric networks and control systems.

The predominant criterion for the design of a reliable and survivable system is the minimum-cost in most contexts. An attractive topic for research is to consider a minimum-cost topological optimization design meeting a reliability threshold. Even though the cost has been the primary factor in the network design, recently, the network reliability has grown in relevance. With the progress of Fiber-To-the-Home (FTTH) services for the backbone design in most current networks, combined with the rapid development of network communication technologies, and the explosive increase of applications over the Internet infrastructure, the network reliability has supreme importance, for traditional communication systems but for the defense, business and energy, and emergent fields such as trusted computing, cloud computing, Internet of Things (IoT) and Next Generation Networks (NGN), the fault tolerance is critical.

We can distinguish two main problems to address in the analysis and design of network topologies. First, the robustness is usually met under multi-path generation. Therefore, we require certain number of node-disjoint paths between distinguished nodes, called terminals. The second problem is to meet a minimum-reliability requirement in a hostile environment, using the fact that both nodes and links may fail. Both problems are strongly related, where sometimes the minimum-cost topology already meets the reliability threshold, or it should be discarded, and the design is challenging.

This thesis deals with a topological optimization problem meeting reliability constraints. The Generalized Steiner Problem with Node-Connectivity Constraints and Hostile Reliability (GSP-NCHR) is introduced, and it is an extension of the well-known Generalized Steiner Problem (GSP). Since GSP-NCHR subsumes the GSP, it belongs to the class of $\mathcal{NP}$-Hard problems. A full chapter is dedicated to the hardness of the GSP-NCHR, and an analysis of particular sub-problems. Here, we address the GSP-NCHR approximately. Our goal is to meet the topological

requirements intrinsically considered in the GSP-NCHR, and then test if the resulting topology meets a minimum reliability constraint.

As a consequence a hybrid heuristic is proposed, that considers a Greedy Randomized construction phase followed by a Variable Neighborhood Search (VNS) in a second phase. VNS is a powerful method that combines local searches that consider different neighborhood structures, and it was used to provide good solutions in several hard combinatorial optimization problems. Since the reliability evaluation in the hostile model belongs to the class of $\mathcal{NP}$-Hard problems, a pointwise reliability estimation was adopted. Here we considered Recursive Variance Reduction method (RVR), since an exact reliability evaluation is prohibitive for large-sized networks.

The experimental analysis was carried out on a wide family of instances adapted from TSPLIB, for heterogeneous networks with different characteristics and topologies, including up to 400 nodes. The numerical results show acceptable CPU-times and locally-optimum solutions with good quality, meeting network reliability constraints as well.

Keywords: Topological Network Design, Network Reliability, Simulation, Network Optimization, Backbone, RVR, Metaheuristics, VNS.

Contents

Glossary

- **GSP**: Generalized Steiner Problem.
- **GSP-NC**: GSP with Node-Connectivity Constraints.
- **GSP-EC**: GSP with Edge-Connectivity Constraints.
- **GSP-NCHR**: GSP with Node-Connectivity Constraints and Hostile Reliability.
- **GSP-ECHR**: GSP with Edge-Connectivity Constraints and Hostile Reliability.
- **SNDP**: Survivable Network Design Problem.
- **GSNDP**: Generalized Survivable Network Design Problem.
- **GNDP**: Generalized Network Design Problem.
- **VCSNDP**: Vertex Connectivity Survivable Network Design Problem.
- **SN-MSP**: Survivable Network with Minimal Steiner nodes Problem.
- **SMT**: Steiner Minimal Tree.
- **MCSP**: Minimal Connected Sub Graph Problem.
- **MST**: Minimum Spanning Tree.
- **TSP**: Travel Salesman Problem.
- **BNDP**: Backbone Network Design Problem.
- **KSP**: K Shortest Paths.
- **WAN**: Wide Area Network.
- **IP**: Internet Protocol.
- **MPLS**: Multiprotocol Label Switching.
- **ZDD**: Zero Decision Diagram.
- **CMC**: Crude Monte Carlo.
- **RVR**: Recursive Variance Reduction.
- **RNN**: Random Neural Network.
- **SMBS**: Stochastic Monotone Binary System.

- **GRASP**: Greedy Randomized Adaptive Search Procedure.
- **IoT**: Internet of Things.
- **GA**: Genetic Algorithm.
- **RL**: Relocation Heuristics.
- **VNS**: Variable Neighborhood Search.
- **ILS**: Iterated Local Search.
- **TS**: Tabu Search.
- **VND**: Variable Neighborhood Descent.
- **VNDS**: Variable Neighborhood Decomposition Search.
- **BVNS**: Biased Variable Neighborhood Search.
- **PVNS**: Parallel Variable Neighborhood Search.

Chapter 1

Introduction

1.1 Context

This thesis is developed for the Master in Informatics, under the Program for the Development of Basic Sciences (PEDECIBA), and Universidad de la República (UdelaR). This work is developed under the framework of a more general network planning project for modern communication networks. This is generally a complex and demanding task, which is accomplished by optimization as a main tool, and combines a quantitative analysis and evaluation as a primary element in the cycle of optimization. In this thesis, we wish to develop a research activity that includes the design of highly-reliable massive telecommunication networks. Given the previous experience in this field, it is essential to assist on decision-making, which is extremely useful for the design of fiber-optics communications.

The information revolution shocked the world during the XX and XXI centuries, and it represents one of the most relevant revolutions in history. The impact was even greater when sharing digital information, allowing cooperation and convergence between both technologies and people. At the beginning, telephonic networks[1] were considered to satisfy the data communication needs. Currently, the situation is much different, and data networks were adapted to pursue different goals, normally by means of service integration and traffic needs. The data networks allow the multiple convergence of different communication technologies permanently, even when the original deployment has more than a century. The interconnection allows

[1] Network: it can be considered as a set of nodes and a set of links between them.

hub, storage and centrality of information that is sparse among distinct continents. As a result, jobs, e-commerce, business and other routine activities are speed-up, and a great variety of on-line services are available anytime and anywhere, with anything at hand (i.e., a cell-phone). The network design task, combining different traffic and services, among many other factors, is not easy at all, but the contrary. This task is complex, and the design, network dimensioning and optimization[2] represents hard decisions to make. This complex task must be simultaneously accomplished with the development of a network topology[3] that meets a specific reliability threshold[4] suitable for the context. A large number of sites with different characteristics are interconnected during the network design, in order to meet a pre-established reliability bound at the minimum cost.

The goal in every topological design[5] is to adapt the technological requirements from the context as much as possible, meeting the budget constraints imposed by the project (which implies the cost of infrastructure but also factors related with an expected quality of service). In this work we address a topological design of highly-reliable networks[6], adding different optimization phases by means of quantitative evaluations in order to determine if the desired reliability parameter is achieved.

1.2 Problem

During the first phase of this thesis, a literature review was performed. As a result, we defined the problem under study with the following two items:

- Given a network where the potential link-costs are known, design a minimum-cost network meeting predetermined connectivity and reliability constraints (inputs of the problem).
- Perform a quantitative analysis of the results, in terms of cost.

[2]Optimization: a field of mathematics that assists on decision making, by means of a minimization/maximization of a quantity, using a specific criterion.

[3]Network Topology: physical configuration in which nodes are interconnected in a network.

[4]Reliability: the probability of correct operation of a system on given conditions during a specific period of time.

[5]Topological design: stage of the network planning process which consists in the physical location of the network components and their interconnections.

[6]Structural Reliability: probability of correct operation of a system, given the occurrence of failures on the network components.

In this context, it is relevant to dispose of methods for the topological network design meeting certain connectivity requirements (i.e., two node-disjoint paths between nodes) and simultaneously, some network reliability requirement (measured in probabilistic terms) exceeding a predefined threshold (problem data). The problem involves a mixture of structural reliability and topological survivability[7] of a network. In a first phase, a literature review is performed and, in a second phase, different solutions to the problem are proposed. The third phase is the implementation of the selected methodology. Finally, in the fourth phase, an experimental analysis is carried out to measure quantitatively the quality of the solution obtained following the designed methodology, and to determine, if possible, *how good* are the returned solutions.

1.3 Goals

1.3.1 General Goal

Develop a heuristic[8] whose result is the design of a network topology (i.e., associated graph) meeting connectivity requirements between pairs of nodes (problem data) and a minimum reliability threshold (problem data). Answer key-questions, in order to understand the interplay between topological survivability and structural network reliability.

1.3.2 Specific Goals

The author of this thesis is proposed to perform an in-depth study of the concepts of *Structural Reliability* and *Topological Survivability*. A specific goal is to get skills in network reliability and planning, particularly on the topological design of strategic complex networks with critical/relevant applications. Learn network planning tools and how to implement efficient algorithms to tackle $\mathcal{NP}$-Hard problems[9], such as the problem addressed in this thesis.

[7] Topological survivability: is to accomplish certain network connectivity levels.

[8] Heuristic: method and exploratory algorithms for the resolution of problems, where the solutions are discovered as a result of the progress achieved during a search.

[9] $\mathcal{NP}$-Hard: so far, these problems cannot be solved efficiently (in polynomial-time with respect to the size of the input).

1.4 Expected Results

Macroscopically, it is expected to offer a methodology that serves as a base-step for decision-making in the development of fault-tolerant telecommunication networks. This is typically the case of a backbone network design[10] of a Wide Area Network (WAN), (i.e., Internet). In order to meet these objectives, the following tasks should be performed:

- Understand the mathematical model associated to the problem to solve.
- Perform a literature review.
- Get a better insight of the following concepts:
  - Topological network survivability.
  - Structural network reliability.
- Explore different approaches to propose an approximate solution.
- Select and implement a solution.
- Measure the quality of the results obtained.

1.5 Methodology

In a previous stage to the development of a solution for the problem, the author performed a literature review, understanding the main concepts and related fields of knowledge. As far as we know, the object under study in this thesis is novel. We can find a scarce number of close problems from the scientific literature. In fact, the closest works from the literature either deal with network reliability, or network optimization independently, but not both. The first stage of this project is focused on understanding the problem and propose a formal (mathematical programming) definition. Given that, the problem under study belongs to the $\mathcal{NP}$-Hard class, an exact evaluation algorithm is prohibitive for large networks. As a consequence, a metaheuristic is adopted. In terms of the optimization problem, several metaheuristics were studied to potentially address the problem[11]. Among those metaheuristics we can find GRASP [24, 34, 76, 90, 93, 94, 101, 112, 95, 58, 92, 89, 71, 84, 38, 37, 32, 2, 86, 11, 68, 105, 80] and its particular version for GSPNC [24, 94],

[10] Backbone: is the skeleton or main core of a network.

[11] Metaheuristic: particular heuristics that serve as a template to solve a very large class of computational problems.

Genetic Algorithms [4, 76], Tabu Search [76, 75], Variable Neighborhood Search or VNS [76, 52, 49, 51, 12, 108, 25, 35, 30, 65, 105, 80, 114, 54] and Iterated Local Search, or ILS [76, 69, 68]. The selection of a metaheuristic provides opportunities to use a powerful and flexible tool, that can be easily combined with hybrid method or specific heuristics suitable for the problem. Once analyzed and understood a variety of potential metaheuristics for our network optimization problem (the construction phase of our topological network design), the decision was to adopt VNS. This metaheuristic is based on a simple principle: a systematic variation of neighborhood structures during the search. The accuracy to switch neighborhood structures is essential. VNS has shown its effectiveness by means of several experiments showing equal or better results than most metaheuristics for a great variety of combinatorial optimization problems, which makes this selection attractive.

Analogously, for the stage of network reliability analysis, different evaluation techniques were studied. An exact network reliability evaluation belongs to the class of $\mathcal{NP}$-Hard problems, for our hostile model of simultaneous links and node-failures. Therefore, simulation methods were considered[12], such as Crude Monte Carlo, or CMC [74] and Recursive Variance Reduction or RVR [74, 16, 21, 19, 63, 17, 18, 14, 13, 23, 22, 110, 115]. Even though Crude Monte Carlo proposed an unbiased reliability estimation, this technique is not suitable for highly reliable scenarios, since it does not satisfy the property of bounded relative error. An outstanding method for variance reduction is RVR, which was selected in this thesis. The implementation is not trivial, but there is both, practical and theoretical evidence that RVR presents much reduced variance than CMC. In practice, RVR is suitable for the reliability estimation on large networks, even under highly reliable scenarios.

1.6 Conclusions

In this thesis we study the topological design of highly-reliable networks, tackling two sub-problems clearly identified: the network optimization problem and the minimum network reliability requirement. The network optimization problem is here addressed using metaheuristics, since it is an

[12]Simulation: is to perform repetitions of a model under a fixed assumption.

$\mathcal{NP}$-Hard problem [46, 112, 95], and therefore, the application of exact methods is prohibitive in terms of computational time, even for networks with small and moderate size. For that reason, we decided to adopt Variable Neighborhood Search, VNS. The reasons to support this decision will be exposed in Chapter 3. In terms of the network reliability evaluation method, simulation methods are used, since exact reliability evaluation methods are also prohibitive. The Recursive Variance Reduction (RVR) method was selected for the pointwise reliability estimation in our hostile environment of simultaneous link/node failures. The reasons to select RVR are also discussed in Chapter 3. We do not have benchmark instances for our network optimization problem[13] in order to measure the quality of the results. Nevertheless, the CPU-times are acceptable, and the returned solutions are locally-optimal, with good quality in terms of costs. It is worth to note that the related literature from the scientific community is scarce, and the problem under study is novel.

1.7 Structure of this Thesis

This thesis is organized in the following manner. Chapter 1 serves as an introduction, and contains the motivation of this thesis, some elements of the problem under study and comments on the selected methodology for its resolution. Chapter 2 presents the terminology that will be used throughout this thesis. A description of the problem and reasons to select VNS and RVR as the building-blocks of our resolution is provided in Chapter 3. Chapter 4 formally presents the Generalized Steiner Problem with Node-Connectivity Constraints and Hostile Reliability (GSPNCHR) with a mathematical programming formulation. Its $\mathcal{NP}$-Hardness is established, and particular cases are also discussed. The related work for the selected resolution methods is covered in Chapter 5. Full details of the algorithmic resolution is presented in Chapter 6. The experimental tests together with a quantitative analysis of the results is included in Chapter 7. Chapter 8 presents Concluding remarks, and Chapter 9 points out trends for future work and possible research fields that extend or complement this thesis. An Appendix is devoted to validation tests, some special procedures involved in the algorithmic design and a

[13]Benchmark: technique used to measure the performance of a system or part of it, commonly in relation with a parameter of reference.

visualization tool for graphs that is also a product of this thesis.

Chapter 2

Background

This chapter includes the basic terminology from Network Optimization, Complexity and Graph Theory that will be used throughout this thesis. The reader is invited to consult the books [53, 34, 74, 93] for additional terminology.

2.1 Concepts on Network Optimization

1. *Graph*: a set of nodes and links between them. The links could be directed; in that case we have a *directed graph*.
2. *Network*: a weighted graph, where the weight is a function on the nodes and/or links that represent either costs, capacities or probabilities.
3. *Backbone*: is the skeleton or main core of a network. A fixed network could have more than one backbone (i.e., Internet).
4. *Reliability*: is the probability of correct operation of a system.
5. *Topological Design*: stage of the network planning process, which consists of the location of the network components and their interconnections.
6. *Survivability*: is the ability of a system, sub-system, equipment, process or procedure of its correct functioning during and after an alteration.
7. *Topological Survivability*: is to meet certain network connectivity levels of the network. It is precisely the existence of a pre-established number of node-disjoint (or link-disjoint) paths between every pair of terminal nodes.
8. *Heuristic*: exploration methods or algorithms to solve problems, where the solutions are discovered by the evaluation of the progress achieved

during the search of a final result. Even though the exploration is algorithmic, an evaluation is empirical. They are normally employed to address hard combinatorial optimization problems, and trade optimality for computational feasibility.

9. *Metaheuristic*: particular heuristics that serve as a template to solve a very large class of computational problems. The common target is a combinatorial optimization problem.
10. *Optimization*: maximization of an objective function (e.g., gains, velocity, efficiency, others), or minimization (e.g., cost, time, risk, error, others) subject to a feasible set of one or multiple constraints. The constraints mean that not every decision (solution) is feasible. In systems engineering, an optimization process implies the enhancement of a system, with the available resources (bandwidth, CPU, memory, etc.). In complex systems, a good approximative solution suffices.
11. *Combinatorial Optimization*: an optimization problem where the feasible set is finite.
12. Locally-optimum solution: best solution in a set of neighbor solutions.
13. Globally-optimum solution: best solution in the feasible set.
14. $\mathcal{NP}$: is the set of decision problems that can be solved in polynomial-time by a non-deterministic Turing machine.
15. $\mathcal{NP}$-Hard: the set of problems H such that every problem $L \in \mathcal{NP}$ can be reduced to H in polynomial-time. An $\mathcal{NP}$-Hard problem is at least as hard as any problem in the class $\mathcal{NP}$. In fact, if we solve a problem from the class $\mathcal{NP}$-Hard, then we can solve all the problems from the $\mathcal{NP}$ class.
16. $\mathcal{NP}$-Complete: the set of $\mathcal{NP}$ decision problems that belong to the $\mathcal{NP}$-Hard class. This class represents the hardest decision problems belonging to the $\mathcal{NP}$-class.
17. *Simulation*: repetitive experimentation with a model with a fixed hypothesis.
18. *Greedy Algorithm*: iteratively picks the cheapest item, in order to build the best solution of a combinatorial optimization problem. In most cases, Greedy does not find the globally-optimum solution, but a good approximation.
19. *Neighborhood*: a set of solutions that include a specific member x. We

can freely used these neighborhood, meeting the following clauses:

- x belongs to all its neighborhoods.
- A set that contains a neighborhood of x is also a neighborhood.
- The intersection of two neighborhoods of x is also a neighborhood.
- For every neighborhood V of x, there exists another neighborhood U of x such that V is a neighborhood of all the points of U.

2.2 Graph-Theoretic Terminology

In this section we present basic graph-theoretic terminology that will be used throughout this thesis [94].

1. *Adjacency*: two nodes u and v are adjacent if the link $\{u, v\}$ belongs to the graph. In directed graphs the order matters, and we denote (u, v) to the ordered pair. We also say that the link $\{u, v\}$ is adjacent to both nodes u and v.
2. *Degree*: the degree $d(v)$ of a node v is the number of adjacent links to v. A node is isolated if it has degree 0.
3. *Induced graph*: given a graph $G = (V, E)$ and a set $U \subseteq V$, the induced graph $G(U)$ denotes the graph in the node-set U, with those links from G whose extremes are included in U.
4. *Path*: non-empty graph $P = (V, E)$ such that $V = \{v_1, \ldots, v_k\}$ and $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)\}$. The nodes v_1 and v_k are connected by P, and v_1 and v_k are the extremes of P. The remaining nodes are internal nodes.
5. *Cycle*: given a path $P = \{v_1, \ldots, v_k\}$, the graph C obtained by the concatenation between P and $\{v_k, v_1\}$ is a cycle.
6. *Node-Disjoint Path*: two paths p and q are node-disjoint if $p \cap q = \{v_1, v_k\}$, being v_1 and v_k the extremes of both p and q. A generalization for multiple disjoint paths is straight.
7. *Independent Paths*: two paths p_1 and p_k are independent if $p_1 \cap p_k = \emptyset$, this is, p_1 and p_k do not share nodes in common.
8. *Subgraph*: given a graph $G = (V, E)$, $H = (V', E')$ is a subgraph of G if $V' \subseteq V$, $E' \subseteq E$ and $\forall (u, v) \in E'$, $u, v \in V'$.

9. *Connected Graph*: a graph $G = (V, E)$ is connected if for each pair of nodes $u, v \in V$ there exists a path that connects u and v in G.
10. *Tree*: a graph $G = (V, E)$ is a tree if it is connected and for all the links $e \in E$, the graph $G' = (V, E \setminus \{e\})$ is not connected.
11. *Spanning Tree*: given a connected graph $G = (V, E)$, a subgraph $H = (V, E')$ is a spanning tree if H is connected and for all the links $e \in E'$, $H' = (V, E' \setminus \{e\})$ is not connected.
12. *k-Node Connectivity*: a graph $G = (V, E)$ is k-node connected if for all $u, v \in V$, there exist at least k node-disjoint paths in G that connect them.
13. *Terminal nodes*: a distinguished node-set T that belongs to the backbone are called terminal-nodes or fixed nodes. These nodes generally correspond to access points in the local networks.
14. *Matrix with the connectivity requirements*: $R = \{r_{i,j}\}_{i,j \in T}$ is a matrix that stores, for every pair of terminal nodes $i, j \in T$, a non-negative integer $r_{i,j}$. The requirement $r_{i,j}$ means that we must construct $r_{i,j}$ node-disjoint paths between the terminal nodes i and j.
15. *Backbone Network Design Problem (BNDP)*: given a network G_B equipped with a terminal-set T, find the minimum-cost network $H_B \subseteq G_B$ that meets the connectivity requirements R for the terminal-nodes T.
16. *Key-node*: consider a feasible solution G_{sol} that meets the connectivity requirements R. A key-node is a non-terminal node $v \in V$, whose degree $d(v)$ is three or greater.
17. *Key-path*: consider a feasible solution G_{sol} that meets the connectivity requirements R. A key-path is a path belonging to G_{sol}, such that the internal nodes are non-terminal nodes with degree 2, and whose extremes are either terminals or key-nodes.
18. *Key-tree*: consider a feasible solution G_{sol} that meets the connectivity requirements R, and $v \in G_{sol}$ a key-node. The key-tree rooted at v is the tree composed by all the key-paths belonging to G_{sol} where v is one of the extremes. Topologically, this is a set of key-paths that share a key-node as a common extreme.

Chapter 3

Problem Definition

3.1 Motivation

Recently, the traditional design of copper lines, the redundancy and survivability [1] were not consider a relevant issue. This is due to the fact that multiple routes were mandatory, given the limited capacity of copper lines. For instance, several central sites were required, commonly called gateway [2]). As a consequence, the communication networks were not originally deployed in order to have enough robustness under single point of failures, or failures on the network sites. The arrival of fiber-optics communication and its high capacity brought sparse networks. The network design is more relevant, and requires a smart engineering. In particular, it must be fault tolerant and highly reliable. In telephonic services, we are only interested in the network topology. In this case the network is a set of nodes or offices and fiber-optics that interconnect them. The survivability is the existence of a pre-established number of node-disjoint paths. In practice, a low-cost network is first deployed, and an optimization process takes place, where the costs are considered (either routing or traffic costs). In a telephonic service, the offices are classified according to their importance in the following way:

- Special offices or *terminals*, meeting a high survivability level.
- Ordinary offices, that should be simply connected to the network, and
- Optional offices, that could be included or not in the network.

[1] Survivability: is the ability of a system, sub-system, equipment, process or procedure of its correct functioning during and after an alteration.

[2] Gateway: in a communication network, a gateway is a network element equipped in order to interact with other networks using different protocols.

It is known the pair of offices that accept a potential link, with a corresponding cost between them. The problem can be summarized in the selection of potential fiber-optics links that should be deployed in order to meet survivability aspects at the minimum cost, such that:

- The elimination of a single link does not disconnect two terminals.
- The elimination of a single office does not disconnect two special offices or terminals.

Topologically, this is to build two node-disjoint paths between the terminal nodes. A major refinement could establish three or more node-disjoint paths between some terminals, increasing the level of survivability under potential disasters or multiple node failures and/or link-cuts. This example can be easily extended to other context with similar characteristics, and summarizes the basis for the first phase of the problem under study in this thesis. A metaheuristic serves as a template or a generic framework [3] to solve a wide variety of hard combinatorial problems. A construction algorithm is needed to address a minimum-cost network design meeting connectivity requirements. Here, a metaheuristic is also considered for optimization, that will be discussed in Section 3.2.

In the previous example of a telephonic service, let us assume that each network component (nodes and links) have an associated elementary reliability (operational probability), which is known. We want to determine the network reliability for the topology that results from the first optimization phase. The goal in this second phase is that the resulting topology meets a certain reliability threshold established for the network operator or user. For that purpose, it is necessary to consider an algorithm to find the reliability measure for a given network. This topic is discussed in Section 3.3.

The final solution simultaneously solve both phases, using a multi-start optimization process followed by quantitative network reliability evaluations to determine if the networks meets the reliability threshold.

[3]Framework: in software development a framework is a structure in which another software projects can be organized and developed.

3.2 Choosing a Metaheuristic

There is a large class of potentially useful metaheuristic to address the problem at hand. After an analysis of possible metaheuristics, Variable Neighborhood Search (VNS) was selected. Why VNS?

The decision is not only based on a variety of metaheuristics that are potentially applicable to solve a specific problem, or if there is controversy for a particular context. This approach not only shortens the decision, but also makes it difficult, since commonly there is not available information to perform the correct decision. The first step is to consider the desirable qualities of a metaheuristic, and determine if these qualities are met. In this sense, VNS is based on a simple principle, not yet deeply explored, which is the systematic variation of neighborhood structures during the search. The accuracy to switch the different structures is crucial. Its effectiveness has been tested over different combinatorial problems and experiments, showing equal or better results than most metaheuristics, and faster. VNS has reached optimality or almost optimality in several datasets of a wide variety of problems, with moderate or reasonable CPU-times. A possibility is to find extensions to this metaheuristic [76, 52, 49, 51, 12, 108, 25, 35, 30, 65, 105, 80, 114, 54] or adding VNS to other metaheuristics, obtaining a hybrid proposal. Even though this is not the main goal of this thesis, a flexible algorithmic design is delivered in the search of a solution, and it could enrich the possibilities of future work. GRASP and VNS are powerful methodologies that were widely used. These metaheuristics are very efficient, being excellent methods to address $\mathcal{NP}$-Hard combinatorial problems related with telecommunications.

3.3 Choosing a Reliability Evaluation Method

An essential part of this thesis is to define a network reliability measure, given a topology and the elementary reliabilities of its components. Here we consider the hostile network reliability model, where both links and Steiner (optional) nodes fail independently. The exact reliability evaluation belongs to the class of $\mathcal{NP}$-Hard computational problems. As a consequence, there are exact methods that run in exponential time, or approximative methods. Given the hardness of the underlying model, an exact method is prohibitive for large-sized instances.

Several Monte Carlo based simulation methods are available in the literature. The simplest approach is Crude Monte Carlo (CMC), where the goal is to pick independent replicas of the system and take decisions on it, based on an averaging of observations. Even its simplicity, it is not suitable for highly-reliable systems, which is the target of this thesis. CMC is unbiased, but its mean square error (i.e., its variance) is large under rare-event scenarios. An alternative is Recursive Variance Reduction (RVR) method [74, 16, 21, 19, 63, 17, 18, 14, 13, 23, 22, 110, 115]. This method is selected since RVR is also unbiased, and presents smaller variance than CMC. This property has been proved experimentally and mathematically as well [74, 16, 21, 19, 63, 17, 18, 14, 13, 23, 22, 110, 115]. Furthermore, RVR is suitable for a large variety of models, such as Stochastic Monotone Binary Systems, and our hostile model belongs to this family.

3.4 Problem Formulation

The object under study in this thesis is a combinatorial optimization problem, that promotes an interplay between network reliability and topological network design. The problem is called *Generalized Steiner Problem with Node-Connectivity Constraints and Hostile Reliability*, and we will use the acronym GSPNCHR for short:

Definition 1 ((GSPNCHR)). *Given a simple undirected graph $G = (V, E)$, a set of distinguished nodes $T \subseteq V$ (called terminals), a matrix with link-costs $\{c_{i,j}\}_{(i,j)\in E}$ and a matrix with connectivity requirements $R = \{r_{i,j}\}_{i,j\in T}$. Further, we assume that the links may fail, and the elementary reliabilities are $P_E = \{p_e\}_{e\in E}$, and Steiner nodes belonging to $V - T$ also have an elementary reliability $P_{V-T} = \{p_v\}_{v\in V-T}$. Given a reliability threshold p_{min}, the goal is to build a minimum-cost topology $G_S \subseteq G$ meeting both the connectivity requirements R and the reliability threshold: $R_K(G_S) \geq p_{min}$, being $K = T$ the terminal-set.*

The following notation is used in the definition of the GSPNCHR:

- $\{c_{i,j}\}_{(i,j)\in E}$ is a matrix that returns the link-cost $c_{i,j}$ for all $(i, j) \in E$.
- $R = \{r_{i,j}\}_{i,j\in T}$ is a matrix with the connectivity requirement between different pairs of terminals. Specifically, the positive integer $r_{i,j}$ denotes

the number of node-disjoint paths between the terminals $i, j \in T$ that are required in the solution.

- $R_K(G_S)$ denotes the probability that the random graph G_S spans the terminal set $K = T$, where both links and Steiner nodes may fail with respective probabilities P_E and P_{V-T}. Throughout this thesis we will consider the terminal-set as $K = T$, unless stated otherwise. This model is known in the literature as the hostile network reliability model.

It is worth to note that node-disjoint paths are required in the GSPNCHR. If edge-disjoint paths are required instead, we consider the alternative GSPECHR. The main goals of this thesis is to answer the following key-questions:

1 How many feasible networks there exists given the full probabilistic model (p_{min}, P_E, P_{V-T})?
2 What is the sensibility of the model with respect to the elementary reliabilities? For instance, for any given threshold ($p_{min} = 0.98$), what happens if we fix $p_v = 0.99$ but we pick different values for the elementary link reliabilities $p_e \in \{0.99, 0.97, 0.95\}$? How many feasible networks survive? Analogously, if we fix $p_e = 0.99$ and $p_v \in \{0.99, 0.97, 0.95\}$.
3 How many networks survive on average, for any given probabilistic model? Understand the sensibility of the model with respect to the connectivity requirements $r_{i,j} \in \{2, 3, 4\}$.
4 Is it better to improve the elementary reliability of links, or the reliability of Steiner nodes, in order to meet a demanding reliability threshold?

Currently, there is no polynomial-time algorithm to test whether a given network meets a minimum reliability threshold. Given the hardness of this decision problem, we will consider a relaxation for the GSPNCHR without reliability constraint during the first phase of this thesis. In fact, in order to answer the key-questions, we will produce a full-algorithm to solve the relaxed GSPNCHR, that is called GSPNC. In a second phase of this thesis, we will count the number of feasible solutions returned by our algorithm for the general GSPNCHR. This phase considers a pointwise reliability estimation method called Recursive Variance Reduction (RVR). Chapter 4 provides a formal definition of both problems.

Chapter 4

Problem and Analysis

In this chapter, the problem under study is formalized by means of a combinatorial optimization problem. Its hardness is established. Particular sub-problems are briefly discussed.

4.1 Model

Given an instance $(G, C, R, T, P_E, P_{V-T}, p_{min})$ for the GSPNCHR, where G is the ground graph, $C = \{c_{i,j}\}_{(i,j)\in E}$ is the matrix with the link-costs, $R = \{r_{i,j}\}_{i,j\in T}$ is the matrix with connectivity requirements between terminal nodes $T \subseteq V$, P_E the elementary reliabilities of the links, P_{V-T} the reliability of Steiner (optional) nodes and p_{min} is the reliability threshold. The goal is to find the minimum-cost subgraph $G_S \subseteq G$ meeting both the connectivity requirements R and the reliability threshold $R_K(G) \geq p_{min}$, being $K = T$ the terminal-set. Consider three sets of decision variables:

$$y_{(i,j)}^{u,v} = \begin{cases} 1 & \textit{if} (i,j) \in E \textit{ is used in a path } u - i - j - v \\ 0 & \textit{otherwise} \end{cases}$$

$$x_{(i,j)} = \begin{cases} 1 & \textit{if} (i,j) \in E \textit{ is used in the solution} \\ 0 & \textit{otherwise} \end{cases}$$

$$\hat{x}_i = \begin{cases} 1 & \textit{if the Steiner node } i \in V - T \textit{ is used in the solution} \\ 0 & \textit{otherwise} \end{cases}$$

In this thesis, we introduce the GSPNCHR as the following combinatorial optimization problem:

$$\min \sum_{(i,j)\in E} c_{i,j}x_{i,j}$$

$$s.t.\ x_{ij} \geq y^{u,v}_{(i,j)} + y^{u,v}_{(j,i)}\ \forall\,(i,j) \in E,\ \forall u, v \in T, u \neq v \tag{4.1}$$

$$\sum_{(u,i)\in E} y^{u,v}_{(u,i)} \geq r_{u,v}\ \forall\, u, v \in T,\ u \neq v \tag{4.2}$$

$$\sum_{(j,v)\in E} y^{u,v}_{(j,v)} \geq r_{u,v}\ \forall\, u, v \in T,\ u \neq v \tag{4.3}$$

$$\sum_{(i,p)\in I(p)} y^{u,v}_{(i,p)} - \sum_{(p,j)\in I(p)} y^{u,v}_{(p,j)} \geq 0,\ \forall p \in V - \{u, v\},\ \forall u, v \in T,\ u \neq v \tag{4.4}$$

$$\sum_{(s,i)\in E} x_{s,i} \leq M\hat{x}_s,\ \forall s \in V - T \tag{4.5}$$

$$R_K(G) \geq p_{min} \tag{4.6}$$

$$x_{(i,j)} \in \{0, 1\}\,\forall (i,j) \in E \tag{4.7}$$

$$\hat{x}_i \in \{0, 1\}\,\forall i \in V - T \tag{4.8}$$

$$y^{u,v}_{(i,j)} \in \{0, 1\}\,\forall (i,j) \in E,\ \forall u, v \in T,\ u \neq v \tag{4.9}$$

The objective is to minimize the global cost of the solution. The set of Constraints 4.1 state that links are one-way. The connectivity requirements are expressed by means of Constraints 4.2 and 4.3. Constraints 4.4 represent Kirchhoff law, or flow conservation [41]. Constraints 4.5 state that an incident link to a Steiner node can be used only if the Steiner node is considered in the solution. Observe that M is a large real number; $M = |E|$ can be used in the model without loss of generality. The minimum reliability threshold is established with Constraint 4.6. Finally, the set of constraints 4.7-4.9 state that all the decision variables belong to the binary set $\{0, 1\}$.

4.2 Hardness

In this section, we show that the GSPNCHR belongs to the class of $\mathcal{NP}$-Complete problems. Recall that the Generalized Steiner Problem (GSP) already belongs to this class:

Definition 2 (Generalized Steiner Problem). *Given an undirected graph* $G =$

(V, E) and a matrix with link-costs $C = \{c_{i,j}\}_{(i,j)\in E}$, a terminal-set $T \subseteq V$ and a matrix with requirements $R = \{r_{i,j}\}_{i,j\in T}$, the goal is to find the minimum-cost subgraph $G_S \subseteq G$ such that every pair of terminals $i, j \in T$ is connected by at least $r_{i,j}$ disjoint paths.

If we consider the GSP with $T = V$ and $r_{i,j} = 2$ for all $i, j \in V$, the minimum-cost is not greater than $n = |V|$ if and only if G has a Hamiltonian Tour. Since Hamiltonian tour belongs to Karp list of $\mathcal{NP}$-Complete problems [57], then GSP is $\mathcal{NP}$-Hard. Further, the GSP also belongs to the $\mathcal{NP}$ set, since both the feasibility can be accomplished by Menger theorem, and the cost of a feasible solution is found in an additive manner. Therefore, the GSP belongs to the class of $\mathcal{NP}$-Complete problems.

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

Proof. Consider an arbitrary instance (G, C, R, T) of the GSP. Consider the instance $(G, C, R, T, P_E, P_{V-T}, p_{min})$, where the probabilities are trivially selected as $p_e = 0, \forall e \in E$, $p_v = 1, \forall v \in V - T$, and $p_{min} = 1$. It is straight to see that the mapping $\pi : (G, C, R, T) \rightarrow (G, C, R, T, P_E, P_{V-T}, p_{min})$ can be accomplished in polynomial-time, and, by inclusion, the GSPNCHR is at least as hard as the GSP. Therefore, the GSPNCHR is $\mathcal{NP}$-Hard. □

The same result holds if we consider edge-disjoint requirements instead, and the GSPECHR is also $\mathcal{NP}$-Hard. Theorem 1 can be strengthened considering strong inapproximability results of special sub-problems [42].

4.3 Special Sub-Problems

In the first phase of this thesis, we will tackle a relaxation of the GSPNCHR:

Definition 3 ((GSPNC)). *This is the relaxation of GSPNCHR, without Constraint 4.6. Specifically, given a simple undirected graph $G = (V, E)$, a set of distinguished nodes $T \subseteq V$ (called terminals), a matrix with link-costs $\{c_{i,j}\}_{(i,j)\in E}$ and a matrix with connectivity requirements $R = \{r_{i,j}\}_{i,j\in T}$, build a minimum-cost topology $G_S \subseteq G$ meeting the connectivity requirements.*

In this thesis we develop a full algorithm for the GSPNC. Then, we study the number of feasible solutions for the GSPNC that also meet the reliability threshold. The key-questions of this thesis are strictly related with the number

of feasible solutions for the GSPNC that are also feasible for the GSPNCHR. In this way, we study the interplay between topological network design and reliability analysis. Furthermore, a sensibility on the reliability parameters and connectivity constraints is also discussed.

Source-Terminal Reliability

Let $T = \{s, t\}$ be the terminal-set (known as *source-terminal* model) with $r_{s,t} = 1$. If l denotes the length of the shortest path $P_{s,t}$ between s and t:

Lemma 1. *The threshold p_{min} is met with the shortest path if and only if $p^l \geq p_{min}$.*

Observe that a globally optimum solution is met with the shortest path $G_S = P_{opt}$ under identical costs if l satisfies the previous inequality. The shortest path P_{opt} is found using Dijkstra algorithm [36].

Proposition 1. *If there exists a feasible single path $P_{s,t}$ for the source-terminal scenario with identical probabilities $p_{i,j} = p$, then the globally optimum solution for GSPNCHR can be found.*

Proof. By hypothesis, there exists a feasible path $P_{s,t}$ that meets the reliability threshold p_{min}. In particular, the threshold is met by the shortest path. By Lemma 1, $p^l \geq p_{min}$. Consider the greatest integer h such that $p^h \geq p_{min}$:

$$h = \lfloor \frac{log(p_{min})}{p} \rfloor \tag{4.10}$$

The globally optimum solution for GSPNCHR is obtained applying Cheng-Ansari algorithm [45], finding the minimum cost among all the paths with lengths $i \in \{l, \ldots, h\}$. □

All-Terminal Reliability

Under the all-terminal reliability model, all the nodes are terminal, i.e., $T = V$, and there are no Steiner (optional) nodes. Consider the cheapest or Minimum Spanning Tree $G_S = T$. If T respects the reliability threshold, then the globally optimum solution is met.

Proposition 2. *Under the all-terminal reliability model, a Minimum Spanning Tree T achieves the globally optimum solution if and only if $\prod_{e \in T} p_e \geq p_{min}$.*

Proposition 3. *For the case source-terminal:* $T = \{s, t\}$, $P_E = \{p_{ij}\}_{(i,j)\in E}$, $P_{V\setminus T} = \{1\}_{v\in V\setminus T}$, $r_{st} = 1$, $c_{ij} = c$, $\forall (i,j) \in E$, *and a given* p_{min}, *the global optimal solution of the GSPNCHR can be computed in polynomial time.*

Proof. Given a path p communicating s and t in G, the reliability condition for this path is:

$$\prod_{(i,j)\in p} p_{ij} \geq p_{min}.$$

This constraint can be established by the following equation by applying logarithm to both sides of the inequality.

$$\sum_{(i,j)\in p} (-\log(p_{ij})) \leq -\log(p_{min}).$$

Let us consider the matrix $\hat{P} = \{-\log(p_{ij})\}_{(i,j)\in E}$. The length-bounded Dijsktra Algorithm is applied until the first path that satisfies the reliability condition is found. The length-bounded Dijsktra Algorithm computes the shortest path between two nodes with the condition that this path has no more than l edges (hops), being l a pre-established parameter.

```
Algorithm GSPNCHR_Source_Terminal;
Input: G = (V, E), T = {s, t}, P̂, c, p_min;

1   l ← 1;
2   found_solution ← FALSE;
3   while (l ≤ |E| − 1) and not(found_solution) do
4       [p̂, cost] ← Restricted_Dijkstra(G, P̂, s, t, l);
    /* It is computed the bounded shortest path from s to t based in the matrix P̂ */
    /* the path found p̂ has no more than l hops */
5       if cost ≤ −log(p_min) then
6           found_solution ← TRUE;
7           optimal_solution ← p̂;
8           optimal_cost ← l · c;
9       else l ← l + 1;
11   end_while;
12   if (found_solution) return (optimal_solution,optimal_cost);
```

Figure 4.1: GSPNCHR Source-Terminal Pseudo-code.

If the GSPNCHR instance has a feasible solution, the algorithm 4.1 returns the global optimum for this particular case.

□

Lemma 2. *For the case All Terminal:* $T = V$, $C = \{c_{ij}\}_{(i,j)\in E}$, $P_E = \{p_{ij}\}_{(i,j)\in E}$, *and a given* p_{min}, *the GSPNCHR can be formulated as the following Integer Linear Programming Problem:*

$$\text{(GSPNCHR\_All\_Terminal)} \quad \min \sum_{i,j \in E} c_{ij} \cdot x_{ij} \tag{4.11}$$

s.a.

$$/* \ \textit{Connectivity constraints} \ */$$

$$\sum_{(i,j) \in E} x_{ij} = n - 1, \tag{4.12}$$

$$/* \ \textit{Reliability restriction} \ */$$

$$\sum_{(i,j) \in E} (-\log(p_{ij})) \cdot x_{ij} \leq -\log(p_{min}), \tag{4.13}$$

$$/* \ \textit{Binary decision variables} \ */$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in E. \tag{4.14}$$

Proof. Equation 4.11 minimizes the global connectivity cost. Equation 4.12 indicates that the spanning subgraph must have $n-1$ edges, where $n = |V|$. This condition forces that the topology to be a spanning tree of V. Equation 4.13 is the reliability condition for a feasible solution. Finally (equation 4.14), the decision variable $x_{ij} \in \{0,1\}$ indicates whether or not an edge $(i,j) \in E$ will be part of the solution. □

Let us consider the following Linear Programming Problem:

$$(\hat{\text{P}}): \ \min \sum_{i,j \in E} c_{ij} \cdot x_{ij}$$

s.a.

$$\sum_{(i,j) \in E} x_{ij} \geq n - 1,$$

$$\sum_{(i,j) \in E} (-\log(p_{ij})) \cdot x_{ij} \leq -\log(p_{min}),$$

$$x_{ij} \geq 0, \forall (i,j) \in E.$$

The $\hat{\mathrm{P}}$ formulation is a lineal relaxation of the problem GSPNCHR_All_Terminal. The Lagrangean relaxation of $\hat{\mathrm{P}}$ is given by the following formulation:

$$\hat{\mathrm{P}}_{\lambda_1,\lambda_2}: \min \sum_{i,j\in E} c_{ij}\cdot x_{ij} + \lambda_1 \cdot \left((n-1) - \sum_{(i,j)\in E} x_{ij}\right) + \lambda_2 \cdot \left(-\log(p_{min}) + \sum_{(i,j)\in E} (\log(p_{ij}))\right)$$
$$\text{s.a.}$$
$$x_{ij} \geq 0,\ \forall (i,j) \in E. \tag{4.15}$$

Regrouping terms we have:

$$\hat{\mathrm{P}}_{\lambda_1,\lambda_2}: \min \sum_{i,j\in E} (c_{ij} - \lambda_1 + \lambda_2 \cdot \log(p_{ij}))\cdot x_{ij} + \lambda_1 \cdot (n-1) - \lambda_2 \cdot \log(p_{min})$$
$$\text{s.a.}$$
$$x_{ij} \geq 0,\ \forall (i,j) \in E.$$

If $\exists (\hat{i},\hat{j}) \in E$ such that $(c_{\hat{i}\hat{j}} - \lambda_1 + \lambda_2 \cdot \log(p_{\hat{i}\hat{j}})) < 0$, then we set all variables x_{ij} to zero except $x_{\hat{i}\hat{j}}$. In this way, if $x_{\hat{i}\hat{j}} \to +\infty$ the feasibility is preserved and the objective function tends to $-\infty$. In order to avoid this, we impose that:

$$(c_{ij} - \lambda_1 + \lambda_2 \cdot \log(p_{ij})) \geq 0,\ \forall (i,j) \in E.$$

Since we are minimizing, the optimum of $\hat{\mathrm{P}}_{\lambda_1,\lambda_2}$ with positive coefficients is obtained with $x_{ij} = 0\ \forall (i,j) \in E$. The objective function of the Dual Problem D of the primal problem $\hat{\mathrm{P}}$ is given by:

$$\Phi(\vec{\lambda}) = \Phi(\lambda_1,\lambda_2) = \lambda_1 \cdot (n-1) - \lambda_2 \cdot \log(p_{min}).$$

The complete formulation of D dual problem of $\hat{\mathrm{P}}$ is:

$$\begin{aligned} \mathrm{D}: \ & \max \lambda_1 \cdot (n-1) - \lambda_2 \cdot \log(p_{min}) \\ & \text{s.a.} \\ & \quad c_{ij} \geq \lambda_1 - \lambda_2 \cdot \log(p_{ij}),\ \forall (i,j) \in E. \\ & \quad \lambda_1, \lambda_2 \geq 0. \end{aligned}$$

The feasible region of D is:

$$F_D = \{(\lambda_1, \lambda_2) | c_{ij} \geq \lambda_1 - \lambda_2 \cdot \log(p_{ij}),\ (i,j) \in E; \lambda_1, \lambda_2 \geq 0\}.$$

Let us consider a system of equations given by two lines r y $\hat{r}$:

$$\begin{aligned} r: \ & \lambda_1 - \lambda_2 \cdot \log(p_{ij}) = c_{ij}, \\ \hat{r}: \ & \lambda_1 - \lambda_2 \cdot \log(p_{uv}) = c_{uv}. \end{aligned}$$

Assuming that r and $\hat{r}$ are not parallel, the resolution of this system is the point:

$$\lambda_1 = c_{ij} + \frac{(c_{uv} - c_{ij})}{\log(\frac{p_{ij}}{p_{uv}})} \cdot \log(p_{ij}).$$

$$\lambda_2 = \frac{(c_{uv} - c_{ij})}{\log(\frac{p_{ij}}{p_{uv}})}.$$

Let $(\hat{\lambda_1}, \hat{\lambda_2})$ be the point given by:

$$(\hat{\lambda_1}, \hat{\lambda_2}) = \arg\max \left\{ \Phi(\lambda_1, \lambda_2) | \lambda_1 = c_{ij} + \frac{(c_{uv} - c_{ij})}{\log(\frac{p_{ij}}{p_{uv}})} \log(p_{ij}),\ \lambda_2 = \frac{(c_{uv} - c_{ij})}{\log(\frac{p_{ij}}{p_{uv}})}; \right\}$$

$$(u,v) \in E, (i,j) \in E.$$

Let us define $\bar{\lambda}_1$ and $\bar{\lambda}_2$ by:

$$\bar{\lambda}_1 = \min \{c_{ij} | (i,j) \in E\}.$$

$$\bar{\lambda}_2 = \min \left\{ \frac{c_{ij}}{-\log(p_{ij})} | (i,j) \in E \right\}.$$

We consider the following points: $A = (\bar{\lambda}_1, 0)$, $B = (0, \bar{\lambda}_2)$, and $C = (\hat{\lambda_1}, \hat{\lambda_2})$.

Let $\vec{\lambda}^{(opt)} = (\lambda_1^{(opt)}, \lambda_2^{(opt)})$ be given by:

$$(\lambda_1^{(opt)}, \lambda_2^{(opt)}) = \arg\max \{\Phi(A), \Phi(B), \Phi(C)\}.$$

$$(\lambda_1^{(opt)}, \lambda_2^{(opt)}) = \arg\max \left\{ \bar{\lambda}_1(n-1), -\bar{\lambda}_2 \log(p_{min}), \hat{\lambda}_1(n-1) - \hat{\lambda}_2 \log(p_{min}) \right\}.$$

The point $\vec{\lambda}^{(opt)}$ is the global optimum of D and the optimal value is:

$$\Phi(\vec{\lambda}^{(opt)}) = \lambda_1^{(opt)}(n-1) - \lambda_2^{(opt)} \log(p_{min}).$$

Since the duality gap is zero between $\hat{\mathrm{P}}$ and D, the value $\Phi(\vec{\lambda}^{(opt)})$ is also the global optimal value of $\hat{\mathrm{P}}$.

Lemma 3. *The Integer Linear Programming Problem associated with GSPNCHR for the All-Terminal case has as lower bound, the value:*

$$\lambda_1^{(opt)}(n-1) - \lambda_2^{(opt)} \log(p_{min}),$$

being $\lambda_1^{(opt)}$ *and* $\lambda_2^{(opt)}$ *the values computed above.*

Proof. We know that $\hat{\mathrm{P}}$ is a lineal relaxation of GSPNCHR_All_Terminal. Furthermore, $\Phi(\vec{\lambda}^{(opt)})$ is the optimal value of $\hat{\mathrm{P}}$, completing the proof. □

Lemma 4. *For the case All Terminal:* $T = V$, $C = \{c_{ij}\}_{(i,j)\in E}$, $P_E = \{p_{ij}\}_{(i,j)\in E}$, *and a given* p_{min}, *the following Integer Linear Programming Problem is a multi-objective formulation that provide an approximation to the optimal value of the GSPNCHR.*

$$(\bar{\mathrm{P}}): \ \min \sum_{i,j\in E} \alpha \cdot (c_{ij} \cdot x_{ij}) + \beta \cdot \left(\sum_{(i,j)\in E} (-\log(p_{ij})) \cdot x_{ij} \right)$$

s.a.

$$\sum_{(i,j)\in E} x_{ij} = n-1,$$

$$x_{ij} \in \{0,1\}, \ \forall (i,j) \in E.$$

Proof. Fixed $\alpha \geq 0$ and $\beta \geq 0$ the objective have two components which are minimized:

- $\sum_{i,j \in E} \alpha \cdot (c_{ij} \cdot x_{ij})$, influencing directly in the minimization of the cost associated with the spanning tree for V.
- $\beta \cdot (\sum_{(i,j) \in E} (-\log(p_{ij})) \cdot x_{ij})$, linked directly to the reliability maximization of the required network topology (a spanning tree covering the set V).

The constraint $\sum_{(i,j) \in E} x_{ij} = n - 1$ guarantees that the computed feasible solution is a spanning tree for V. □

By relaxing the constraint $x_{ij} \in \{0, 1\}$, $\forall (i,j) \in E$ by $x_{ij} \geq 0$, $\forall (i,j) \in E$, we have the problem:

$$
\begin{aligned}
(\bar{\mathrm{P}}_L): \ & \min \sum_{i,j \in E} (\alpha \cdot c_{ij} + \beta \cdot (-\log(p_{ij}))) \cdot x_{ij} \\
& \text{s.a.} \\
& \sum_{(i,j) \in E} x_{ij} = n - 1, \\
& x_{ij} \geq 0, \forall (i,j) \in E.
\end{aligned}
$$

The Lagrangean relaxation of $\bar{\mathrm{P}}_L$ is given by the following formulation:

$$
\begin{aligned}
(\bar{\mathrm{P}}_L^{(\lambda_1)}): \ & \min \sum_{i,j \in E} (\alpha \cdot c_{ij} + \beta \cdot (-\log(p_{ij})) - \lambda_1) \cdot x_{ij} + \lambda_1 \cdot (n-1) \\
& \text{s.a.} \\
& x_{ij} \geq 0, \forall (i,j) \in E.
\end{aligned}
$$

If $\exists (\hat{i}, \hat{j}) \in E$ such that $(\alpha \cdot c_{\hat{i}\hat{j}} + \beta \cdot (-\log(p_{\hat{i}\hat{j}})) - \lambda_1) < 0$, then we set all variables x_{ij} to zero except $x_{\hat{i}\hat{j}}$. In this way, if $x_{\hat{i}\hat{j}} \to +\infty$ the feasibility is preserved and the objective function tends to $-\infty$. In order to avoid this, we impose that:

$$\alpha \cdot c_{ij} + \beta \cdot (-\log(p_{ij})) - \lambda_1 \geq 0, \forall (i,j) \in E.$$

The global optimal solution for the problem $\bar{\mathrm{P}}_L^{(\lambda_1)}$ satisfying that constraints is accomplished by setting $x_{ij} = 0, \forall (i,j) \in E$. Under these conditions the value of the objective function is the objective function of the

Dual problem of $\bar{\mathrm{P}}_L$ (let us denote it by $\bar{D}$) which is given by: $\Phi(\lambda_1) = \lambda_1 \cdot (n-1)$.

The Dual Problem $\bar{D}$ can then be formulated as:

$$
\begin{aligned}
(\bar{D}): \ \max \ & \lambda_1 \cdot (n-1) \\
& \text{s.a.} \\
& \alpha \cdot c_{ij} + \beta \cdot (-\log(p_{ij})) \geq \lambda_1, \, \forall (i,j) \in E, \\
& \lambda_1 \geq 0
\end{aligned}
$$

Let us consider:

$$\gamma_M = \arg\max \left\{ \alpha \cdot c_{ij} - \beta \cdot \log(p_{ij}) | (i,j) \in E \right\}.$$

Thus, $\gamma_M = \alpha \cdot c_{\hat{i}\hat{j}} - \beta \cdot \log(p_{\hat{i}\hat{j}})$ for a certain $(\hat{i}, \hat{j}) \in E$. The global optimal solution of the Dual problem $\bar{D}$ is $\lambda_1 = \gamma_M$ and the optimum value is $\gamma_M \cdot (n-1)$.

By the Duality Theorem for Linear Programming Problems we have that the duality gap is zero, and the optimum value of $\bar{\mathrm{P}}_L$ is also:

$$\gamma_M \cdot (n-1) = (\alpha \cdot c_{\hat{i}\hat{j}} - \beta \cdot \log(p_{\hat{i}\hat{j}})) \cdot (n-1).$$

Theorem 2. *Fixed a p_{min} value there exist $\hat{\alpha}$ and $\hat{\beta}$ for the problem $\bar{\mathrm{P}}$ such that if $\hat{E} \subseteq E$ is the solution edge set, it is fulfilled that:*

$$-\sum_{(i,j) \in \hat{E}} \log(p_{ij}) \leq -\log(p_{min}).$$

Proof. Let us consider the following values for $\hat{\alpha}$ and $\hat{\beta}$: $\hat{\alpha} \geq 0$ arbitrary,

$$\hat{\beta} = \left(\frac{\log(p_{min})}{\sum_{(i,j) \in E} \log(p_{ij}) \cdot \hat{x}_{ij}} \right),$$

where $\{\hat{x}_{ij}\}_{(i,j) \in E}$ are the values of the decision variables for the globally optimal solution of the following Linear Programming Problem:

$$(\hat{\mathrm{M}}):\ \max\ -\sum_{i,j\in E}\log(p_{ij})\cdot x_{ij}$$

s.a.

$$-\sum_{i,j\in E}\log(p_{ij})\cdot x_{ij}\leq -\log(p_{min})$$

$$x_{ij}\geq 0,\ \forall (i,j)\in E.$$

Since when solving $(\hat{\mathrm{M}})$ the set of values $\{\hat{x}_{ij}\}_{(i,j)\in E}$ fulfills the first constraint of the $(\hat{\mathrm{M}})$ model, we have that:

$$-\sum_{i,j\in E}\log(p_{ij})\cdot \hat{x}_{ij}\leq -\log(p_{min}).$$

By dividing this inequality by the left side we have that the $\hat{\beta}$ value satisfies:

$$1\leq \hat{\beta}=\frac{log(p_{pmin})}{\sum_{(i,j)\in E}\log(p_{ij})\cdot \hat{x}_{ij}}.$$

Let us consider now:

$$\sum_{(i,j)\in \hat{E}}(-\log(p_{ij}))\leq \hat{\beta}\cdot\sum_{(i,j)\in \hat{E}}(-\log(p_{ij}))=\left(\frac{log(p_{pmin})}{\sum_{(i,j)\in E}\log(p_{ij})\cdot \hat{x}_{ij}}\right)\cdot\sum_{(i,j)\in \hat{E}}(-\log(p_{ij}))=$$

$$=(-\log(p_{min}))\cdot\left(\frac{\sum_{(i,j)\in \hat{E}}\log(p_{ij})}{\sum_{(i,j)\in E}\log(p_{ij})\cdot \hat{x}_{ij}}\right)\leq(-\log(p_{min})),$$

where in the last inequality we use that it is fulfilled: $\left(\frac{\sum_{(i,j)\in \hat{E}}\log(p_{ij})}{\sum_{(i,j)\in E}\log(p_{ij})\cdot \hat{x}_{ij}}\right)\leq 1$.

Then with $\hat{\beta}$ defined above it is satisfied:

$$-\sum_{(i,j)\in \hat{E}}\log(p_{ij})\leq -\log(p_{min}),$$

as required, and completing the proof. □

4.4 Minimum-Weight k-Connected Spanning Networks with Reliability Constraints

In the work [6] Bienstock et. al introduce important theorems related to the "Minimum-Weight k-Connected Spanning Networks" Problem under the hypothesis of triangular inequality in edge costs.

4.4.1 MWKECSNP ILP Formulation based on Bienstock Theorem

In this point we introduce an exact formulation for the MWKECSNP when the triangular inequality is satisfied by the costs of the edges.

Decision variables:

$$y_{(i,j)}^{u,v} = \begin{cases} 1, & \text{if the directed edge } (i,j) \text{ is used in a path communicationg } u \text{ with } v \\ & \text{in the sense } u - i - j - v;\ u, v \in V; \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if the edge } (i,j) \in E \text{ is used in the solution} \\ 0, & \text{otherwise.} \end{cases}$$

Binary-Integer Linear Programming Model for the MWKECSNP:

$$
\begin{aligned}
P_{MWKECSNP}: \ \min & \sum_{(i,j)\in E} c_{ij}\cdot x_{ij} \\
\text{s.a.} & \\
& \sum_{(u,i)\in E} y^{u,v}_{(u,i)} \geq k,\ \forall u,v\in V, \\
& \sum_{(j,v)\in E} y^{u,v}_{(j,v)} \geq k,\ \forall u,v\in V, \\
& \sum_{(i,p)\in I^-(p)} y^{u,v}_{(i,p)} - \sum_{(p,j)\in I^+(p)} y^{u,v}_{(p,j)} \geq 0,\ \forall u,v\in V; \forall p\in V\setminus\{u,v\}, \\
& y^{u,v}_{(i,j)} + y^{u,v}_{(j,i)} \leq x_{ij},\ \forall u,v\in V,\ \forall (i,j)\in E, \\
& \sum_{(v,i)\in E} x_{vi} \leq k+1,\ \forall v\in V, \\
& x_{ij}\in\{0,1\},\ \forall (i,j)\in E; y^{u,v}_{(i,j)}\in\{0,1\},\ \forall (i,j)\in E, \forall u,v\in V.
\end{aligned}
$$

Next, we introduce an exact model for solving the MWKECSNP with Reliability Constraints. The formulation is also a Binary-Integer Linear Programming Model:

$$
\begin{aligned}
P_{MWKECSNP\_RC}: \ \min & \sum_{(i,j)\in E} c_{ij}\cdot x_{ij} \\
\text{s.a.} & \\
& \sum_{(u,i)\in E} y^{u,v}_{(u,i)} \geq k,\ \forall u,v\in V, \\
& \sum_{(j,v)\in E} y^{u,v}_{(j,v)} \geq k,\ \forall u,v\in V, \\
& \sum_{(i,p)\in I^-(p)} y^{u,v}_{(i,p)} - \sum_{(p,j)\in I^+(p)} y^{u,v}_{(p,j)} \geq 0,\ \forall u,v\in V; \forall p\in V\setminus\{u,v\}, \\
& y^{u,v}_{(i,j)} + y^{u,v}_{(j,i)} \leq x_{ij},\ \forall u,v\in V,\ \forall (i,j)\in E, \\
& \sum_{(v,i)\in E} x_{vi} \leq k+1,\ \forall v\in V, \\
& \sum_{(i,j)\in E} -\log(p_{ij})\cdot x_{ij} \leq -\log(p_{min}),\ \forall (i,j)\in E, \\
& x_{ij}\in\{0,1\},\ \forall (i,j)\in E; y^{u,v}_{(i,j)}\in\{0,1\},\ \forall (i,j)\in E, \forall u,v\in V.
\end{aligned}
$$

Let $\hat{P}_{MWKECSNP\_RC}$ be the following model resulting of the linear relaxation in $[0,1]$ of the x_{ij} and $y^{u,v}_{(i,j)}$ variables:

$$\hat{P}_{MWKECSNP\_RC}: \min \sum_{(i,j)\in E} c_{ij} \cdot x_{ij}$$

s.a.

$$\sum_{(u,i)\in E} y^{u,v}_{(u,i)} \geq k,\ \forall u,v \in V,$$

$$\sum_{(j,v)\in E} y^{u,v}_{(j,v)} \geq k,\ \forall u,v \in V,$$

$$\sum_{(i,p)\in I^-(p)} y^{u,v}_{(i,p)} - \sum_{(p,j)\in I^+(p)} y^{u,v}_{(p,j)} \geq 0,\ \forall u,v \in V; \forall p \in V \setminus \{u,v\},$$

$$y^{u,v}_{(i,j)} + y^{u,v}_{(j,i)} \leq x_{ij},\ \forall u,v \in V,\ \forall (i,j) \in E,$$

$$\sum_{(v,i)\in E} x_{vi} \leq k+1,\ \forall v \in V,$$

$$\sum_{(i,j)\in E} -\log(p_{ij}) \cdot x_{ij} \leq -\log(p_{min}),\ \forall (i,j) \in E,$$

$$0 \leq x_{ij} \leq 1,\ \forall (i,j) \in E; 0 \leq y^{u,v}_{(i,j)} \leq 1,\ \forall (i,j) \in E, \forall u,v \in V.$$

Now, let $\hat{P}^{(L)}_{MWKECSNP\_RC}$ be a Lagrangian relaxation of $\hat{P}_{MWKECSNP\_RC}$

formulated as follows:

$$
\begin{aligned}
\hat{P}^{(L)}_{MWKECSNP\_RC}: \ \min & \sum_{(i,j)\in E} c_{ij}\cdot x_{ij} \\
& + \sum_{u,v\in V} \lambda_1^{u,v}\cdot\left(k - \sum_{(u,i)\in E} y^{u,v}_{(u,i)}\right) \\
& + \sum_{u,v\in V} \lambda_2^{u,v}\cdot\left(k - \sum_{(j,v)\in E} y^{u,v}_{(j,v)}\right) \\
& + \sum_{u,v\in V}\sum_{p\in V\setminus\{u,v\}} \lambda_3^{u,v,p}\cdot\left(\sum_{(p,j)\in I^+(p)} y^{u,v}_{(p,j)} - \sum_{(i,p)\in I^-(p)} y^{u,v}_{(i,p)}\right) \\
& + \sum_{u,v\in V}\sum_{(i,j)\in E} \lambda^{u,v}_{4,(i,j)}\cdot\left(y^{u,v}_{(i,j)} + y^{u,v}_{(j,i)} - x_{ij}\right) \\
& + \sum_{v\in V}\lambda_5^v\cdot\left(\sum_{(v,i)\in E} x_{vi} - k - 1\right) \\
& + \lambda_6\cdot\left(\sum_{(i,j)\in E} -\log(p_{ij})\cdot x_{ij} + \log(p_{min})\right) \\
& + \sum_{(i,j)\in E} \lambda_7^{(i,j)}\cdot(x_{ij}-1) \\
& x_{ij}\geq 0,\ \forall (i,j)\in E; 0\leq y^{u,v}_{(i,j)}\leq 1,\ \forall (i,j)\in E, \forall u,v\in V.
\end{aligned}
$$

Let us denote: $\lambda_4^{(i,j)} = \sum_{u,v\in V}\lambda^{u,v}_{4,(i,j)}$. Notice that the following points are satisfied for the $\hat{P}^{(L)}_{MWKECSNP\_RC}$ formulation:

- $\left(k - \sum_{(u,i)\in E} y^{u,v}_{(u,i)}\right)$ is minimum when $\sum_{(u,i)\in E} y^{u,v}_{(u,i)} = g^+(u)$, where $g^+(u)$ is the degree of node u.
- $\left(k - \sum_{(j,v)\in E} y^{u,v}_{(j,v)}\right)$ is minimum when $\sum_{(j,v)\in E} y^{u,v}_{(j,v)} = g^-(v)$, where $g^-(v)$ is the degree of node v.
- $(\sum_{(p,j)\in I^+(p)} y^{u,v}_{(p,j)} - \sum_{(i,p)\in I^-(p)} y^{u,v}_{(i,p)})$ is minimum when $\sum_{(p,j)\in I^+(p)} y^{u,v}_{(p,j)} = 0$ and $\sum_{(i,p)\in I^-(p)} y^{u,v}_{(i,p)}) = g^-(p)$ where $g^-(p)$ is the degree of node p.
- Fixed x_{ij}, the expression $(y^{u,v}_{(i,j)} + y^{u,v}_{(j,i)} - x_{ij})$ is minimum when $y^{u,v}_{(i,j)} = y^{u,v}_{(j,i)} = 0$.
- $\sum_{v\in V}\lambda_5^v\cdot\left(\sum_{(v,i)\in E} x_{vi} - k - 1\right) = \sum_{(i,j)\in E}\lambda_5^i\cdot x_{ij} - (k+1)\cdot\sum_{v\in V}\lambda_5^v$.

By considering these points we provide the following formulation for the $\hat{P}^{(L)}_{MWKECSNP\_RC}$ which considers only the $\{x_{ij}\}$ variables.

$$
\begin{aligned}
\hat{P}^{(L)}_{MWKECSNP\_RC}: \ \min & \sum_{(i,j)\in E} c_{ij}\cdot x_{ij} \\
& + \sum_{u,v\in V} \lambda_1^{u,v}\cdot(k-g^+(u)) \\
& + \sum_{u,v\in V} \lambda_2^{u,v}\cdot(k-g^-(v)) \\
& + \sum_{u,v\in V}\sum_{p\in V\setminus\{u,v\}} \lambda_3^{u,v,p}\cdot(-g^-(p)) \\
& + \sum_{(i,j)\in E} \lambda_4^{(i,j)}\cdot(-x_{ij}) \\
& + \sum_{(i,j)\in E} \lambda_5^{i}\cdot x_{ij} - (k+1)\cdot\sum_{v\in V}\lambda_5^v \\
& + \lambda_6\cdot\left(\sum_{(i,j)\in E} -\log(p_{ij})\cdot x_{ij} + \log(p_{min})\right) \\
& + \sum_{(i,j)\in E} \lambda_7^{(i,j)}\cdot(x_{ij}-1) \\
& x_{ij}\geq 0,\ \forall (i,j)\in E.
\end{aligned}
$$

Lemma 5. *If the lambda parameters satisfy:* $\lambda_1^{u,v}\geq 0$, $\lambda_2^{u,v}\geq 0$, $\lambda_3^{u,v,p}\geq 0$, $\lambda_4^{(i,j)}\geq 0$, $\lambda_5^v\geq 0$, $\lambda_6\geq 0$, *and* $\lambda_7^{(i,j)}\geq 0$, *then* $\hat{P}^{(L)}_{MWKECSNP\_RC}$ *is a relaxation of* $\hat{P}_{MWKECSNP\_RC}$.

By regrouping terms we can formulate $\hat{P}^{(L)}_{MWKECSNP\_RC}$ as follows:

$$\begin{aligned}\hat{P}^{(L)}_{MWKECSNP\_RC}: \min & \sum_{(i,j)\in E}\left(c_{ij}-\lambda_4^{(i,j)}+\lambda_5^{i}-\lambda_6\cdot\log(p_{ij})+\lambda_7^{(i,j)}\right)\cdot x_{ij}\\ &+\sum_{u,v\in V}\lambda_1^{u,v}\cdot(k-g^{+}(u))\\ &+\sum_{u,v\in V}\lambda_2^{u,v}\cdot(k-g^{-}(v))\\ &+\sum_{u,v\in V}\sum_{p\in V\setminus\{u,v\}}\lambda_3^{u,v,p}\cdot(-g^{-}(p))\\ &-(k+1)\cdot\sum_{v\in V}\lambda_5^{v}\\ &+\lambda_6\cdot\log(p_{min})\\ &-\sum_{(i,j)\in E}\lambda_7^{(i,j)}\\ &x_{ij}\geq 0,\ \forall(i,j)\in E.\end{aligned}$$

If $\exists(\hat{i},\hat{j})\in E$ such that $(c_{\hat{i}\hat{j}}-\lambda_4^{(\hat{i},\hat{j})}+\lambda_5^{\hat{i}}-\lambda_6\cdot\log(p_{\hat{i}\hat{j}})+\lambda_7^{(\hat{i},\hat{j})})<0$, then we set all variables x_{ij} to zero except $x_{\hat{i}\hat{j}}$. In this way, if $x_{\hat{i}\hat{j}}\to+\infty$ the feasibility is preserved and the objective function tends to $-\infty$. In order to avoid this, we impose that:

$$c_{ij}-\lambda_4^{(i,j)}+\lambda_5^{i}-\lambda_6\cdot\log(p_{ij})+\lambda_7^{(i,j)}\geq 0,\ \forall(i,j)\in E.$$

The global optimal solution for the problem $\hat{P}^{(L)}_{MWKECSNP\_RC}$ satisfying that constraints is accomplished by setting $x_{ij}=0,\ \forall(i,j)\in E$. Under these conditions the value of the objective function is the objective function of the Dual problem of $\hat{P}_{MWKECSNP\_RC}$ (let us denote it by $\hat{D}_{MWKECSNP\_RC}$) which is given by:

$$\hat{\Phi}(\vec{\lambda})=\sum_{u,v\in V}\left(\lambda_1^{u,v}\cdot(k-g^{+}(u))+\lambda_2^{u,v}\cdot(k-g^{-}(v))-\sum_{p\in V\setminus\{u,v\}}\lambda_3^{u,v,p}\cdot(g^{-}(p))\right)$$

$$-(k+1)\cdot\sum_{v\in V}\lambda_5^{v}+\lambda_6\cdot\log(p_{min})-\sum_{(i,j)\in E}\lambda_7^{(i,j)}.$$

The Dual Problem $\hat{D}_{MWKECSNP\_RC}$ can then be formulated as:

$$\hat{D}_{MWKECSNP\_RC}: \max \ \hat{\Phi}(\vec{\lambda})$$

s.a.

$$c_{ij} - \lambda_4^{(i,j)} + \lambda_5^i - \lambda_6 \cdot \log(p_{ij}) + \lambda_7^{(i,j)} \geq 0, \forall (i,j) \in E$$
$$\lambda_1^{u,v}, \lambda_2^{u,v} \geq 0, \forall u, v \in V$$
$$\lambda_3^{u,v,p} \geq 0, \forall u, v \in V, \forall p \in V \setminus \{u, v\}$$
$$\lambda_4^{(i,j)} \geq 0, \forall (i,j) \in E$$
$$\lambda_5^v \geq 0, \forall v \in V$$
$$\lambda_6 \geq 0$$
$$\lambda_7^{(i,j)} \geq 0, \forall (i,j) \in E.$$

Theorem 3. *Let $\hat{d}_{opt} = \hat{\Phi}(\vec{\lambda}_{opt})$ be the global optimal value of $\hat{D}_{MWKECSNP\_RC}$. Then, $\hat{d}_{opt}$ is a lower bound for the globally optimal value of $P_{MWKECSNP\_RC}$.*

Proof. We know that $\hat{P}_{MWKECSNP\_RC}$ is a lineal relaxation of $P_{MWKECSNP\_RC}$. Moreover, de duality gap between $\hat{P}_{MWKECSNP\_RC}$ and $\hat{D}_{MWKECSNP\_RC}$ is zero since they are Linear Programming Models, thus completing the proof. □

Chapter 5

Related Work

In this chapter we revisit the related work, as well as some works where GRASP and/or VNS methodologies were provided for specific combinatorial optimization problems. Previous works on the Recursive Variance Reduction (RVR) method under different contexts are also cited.

5.1 Topological Network Design

Robledo and Canale [95] develop a GRASP methodology for the backbone design of telecommunication networks. Robledo considers GRASP for the design of a WAN topology [94], using local searches based on Random Neural Networks (RNN). Risso also develops a GRASP combined with evolutionary algorithms for the design of IP/MPLS networks [92].

Resende covers a wide spectrum of GRASP algorithms for several hard combinatorial optimization problems [89, 91], combining Greedy notion with randomization, local searches and even post-optimization methods such as path-relinking as well as parallel continuous GRASP. In [71], different ways to implement GRASP for a multi-objective problem are suggested, specially using path-relinking. In [38], GRASP combined with external path-relinking is considered to minimize the differential dispersion. In [37], GRASP is developed to tackle the graph coloring of convex graphs. In [32], a GRASP algorithm is defined for solving a large-scale single row facility layout problems. In the p-next center problem, we must assign users in the centers in order to minimize the worst distance between a user and its closest center. A GRASP and VNS for solving the p-next center problem was proposed in [68].

In [109], a fully deterministic algorithm of time complexity $O(k^3 log(n))$ was presented for the Vertex Connectivity Survivable Network Design Problem (VCSNDP), being k the maximum connectivity requirement for the problem. This work outperforms previously randomized algorithms. In [43], an enhancement in the computational order with polynomial time for the first proposal authored by Williamson [42] is proposed. The key concept is a combinatorial characterization of *redundant links*. The order is reduced from $O(k^3 n^4)$ to $O(k^2 n^2 + kn^2 \sqrt{log(log(n))})$.

Agrawal, Klein and Ravi [1] developed an approximation algorithm with logarithmic factor for the Generalized Steiner Problem with Edge Connectivity (GSP-EC). More recently, Jain [55] presented a factor-2 approximation algorithm for the GSP-EC, where a feasible solution of a linear programming relaxation of the problem is first found, and the solution is iteratively built. Sartor and Robledo proposed a GRASP/VNS heuristic for the GSP-EC [101]. Kortsarz, Krauthgamer and Lee [60] introduced the first inapproximability result for the Generalized Steiner Problem with Node Connectivity (GSP-NC) when there are no Steiner nodes.

There are several works that develop approximation algorithms for the GSP and sub-problems. In [87], the authors develop approximations for the 2-node connectivity versions stated in [48]. Subsequent articles [47, 43, 111] extend those methods to give approximation algorithms for the GSP-EC without multiple links.

An important particular case for the GSP-NC occurs in the minimum-cost k-node connected spanning graph. In [27, 33, 50] the authors propose approximation algorithms. The reader is invited to consult the references [26, 28, 39, 44]. These articles offer different approximation algorithms and their respective approximation ratios. Some works study the particular case of identical costs, usually known as the minimum cardinality augmentation. Sartor and Robledo solved the GSP-EC [103] and GSP-NC [102, 104] with a GRASP proposal. In [82] E. Paolini explores a generalized formulation that extends the original GSP to infinite sets in metric spaces. In [113], the GSP is addressed in Halin graphs (obtained connecting the leaf-nodes of a tree in a cycle). In [70], Mahjoub and Pesneau study the particular 2-edge connected Steiner subgraph polytope. The authors find polynomial-time cutting-plane solutions for particular cases, where the terminals have special dispositions. As a consequence, they

generalize the previous analysis for Halin graphs presented in [113].

In [77, 78], S. Nesmachnow presents an empirical evaluation of several simple metaheuristics (VNS is among them) to address the GSP, with promising results. In [83], M. Pedemonte and H. Cancela developed an Ant Colony Optimization (ACO) to solve the GSP using parallel computing in order to reduce the CPU-time. In [79], another proposal of ACO for the GSP is presented to tackle the GSP in general graphs, outperforming previous heuristics.

In [97, 96], an integer linear programming formulation was solved using branch-and-cut for the Generalized Network Design Problem (GNDP), applied to two-types of survivability structured: rings and 2-edge connected topologies. In [64], the same formulation is considered in order to solve the $\{0, 1, 2\}$ Survivable Network Design Problem or $\{0, 1, 2\}$-GSNDP, that extends the GNDP and has a direct application for the design of backbone networks.

In [8], the Generalized SNDP or GSNDP with hop-constraints is discussed, studying the static problem (given link-reliabilities) and dynamic problem with an *upgrading*, where the elementary reliabilities can be increased, with an associated cost. In [106], a compilation of several techniques is performed, showing a comparison for the GSP, SNDP and the minimal connected subgraph problem (MCSP).

In [29], an experimental study is carried out using exact algorithms over 2-node connected graph with more than one-hundred nodes, showing the computational feasibility of this solution. In [56], the Survivable Network with Minimum Steiner nodes Problem (SN-MSP) is addressed, by means of a natural transformation from SN-MSP into SNDP, such that a factor α for SNDP implies a factor $\alpha O(k^2)$ for SN-MSP. In [67] the SNDP is tackled using stochastic models. Several algorithms are proposed, in particular a branch and cut to solve the SNDP with an acceptable optimization which shows to speed-up the CPU time. A fast and easy-to-implement technique to strengthen cuts is also suggested. In [107] the SNDP is addressed. In practice, the approximate solution is far from optimal. Then, in this work, an enumeration of optimal solutions is carried out with a compact data structure, called Zero-Suppressed Binary Decision Diagrams (ZDD). The authors show that this method works for several real-world instances.

In [100], the SNDP with mixed node and link requirements is considered.

The authors propose a cutting plane algorithm for an integer linear programming formulation. In [81], the GSNDP for Wireless networks is considered, where the link-activity depends on some parameters and the cost is a function of them. The model proposed is a generalization of several connectivity problems previously addressed in the literature, such as Node-Weighted Steiner Network, Power Optimization and Minimum Connected Dominating Set.

In [12] several heuristics such as VNS, Tabu Search and Relocation Heuristic (RL) are developed to solve the Partitioning signed networks problem. The authors shows that the combination of multi-step relocation heuristics with Tabu Search and VNS produce a fast node-partition algorithm for signed networks that is competitive with existent metaheuristics. VNS is considered in [108] to solved the well-known Steiner Minimal Tree Problem (SMT) in sparse graphs. The authors obtained better results than previous heuristics.

In [25], VNS is adapted to solve a network clustering problem with similar results obtained by machine learning approaches such as clustering k-Means. This work also solves the sum-square of the distance between all the node-pairs, using a novel VNS approach. A modified VNS heuristic is considered in [80] for a k-Means Clustering problem. There, the authors compare k-VNS versus traditional k-means and j-means algorithms. It is worth to remark that VNS outperformed the traditional approaches, specially in large datasets. A hybrid VNS/GA proposal is proposed in [35] to solve the Multicriteria route planning in public transit networks. The hybrid proposal outperforms pure VNS and GA solutions, in both quality and CPU-time. In [30], VNS is developed to solve the k-labelled spanning forest problem, with a strong impact in multi-modal transportation networks. The goal is to build a spanning forest of the ground graph, having the least number of connected components and an upper-bound in the number of labels to use. In [65], the authors consider on one hand VNS, and GRASP on the other, to solve the Capacitated Connected Facility Location problem, that combines locations with Steiner trees. This problem gains relevance for its applications in the last-mile in Fiber-To-The-Home (FTTH) services. Both heuristics obtain solutions with high quality in reduced times. In [105], the same heuristics were employed for the Three-Layer Hierarchical Ring Network Design problem, that is widely used in large telecommunication networks. Better results were obtained using VNS.

In [114], VNS is considered to solve a relay design problem. Given a set of products that will be routed through the network, the relay problem implies to select a route for each product and determine the locations for the relays were the product should be re-processed at certain distance intervals. A VNS is proposed, were different local searches look for the routes for each product and the optimal place for retransmission for a given set of routes, that are found with an implicit enumeration that by means of a dynamic programming algorithm. The experiments confirm that VNS with optimal retransmission assignment outperforms all the existent algorithms from the literature. Several implementations of VNS has been developed for the Traveling Salesman Problem as well, showing that VNS is competitive [54]. Exact solutions for the TSP and extensions can be found in [98]. The reader can appreciate that the CPU-times provided by the exact solutions are longer than the heuristics, and in particular VNS proposal for the TSP.

5.2 Network Reliability

It is worth to remark that there are scarce works that jointly deal with a topological network optimization under reliability constraints. Javiera Barrera et. al. proposed a topological network optimization, trying to minimize costs subject to K-terminal reliability constraints [5]. The authors consider Sample Average Approximation (SAA) method, which is a powerful tool for $\mathcal{NP}$-Hard combinatorial problems and stochastic optimization [59]. They conclude that suboptimal solutions could be found if dependent failures are ignored in the model. The scientific literature also offers topological optimization problems meeting reliability constraints, or reliability maximization under budget constraints, which is known as network synthesis. The reader can find a survey on the synthesis in network reliability in [7]. More recent works propose a reliability optimization in general stochastic binary systems [20], even under the introduction of Sample Average Approximation [85]. Building uniformly most-reliable graphs is an active and challenging research field, where the goal is to find graphs with fixed nodes and links with maximum reliability evaluation in a uniform sense, for the whole compact set of elementary reliabilities $p \in [0, 1]$. There are pairs of nodes and links where such uniformly most-reliable graphs do not exist [9]. The interested reader can consult [3] for conjectures in this field. A strictly

related problem to ours is to consider topological modifications (i.e., moving links, or path replacements, among many others) in order to increase the reliability measure. This problem is not mature, and a recent work propose a novel reliability-increasing network transformation [15]. There, E. Canale et. al. show that any graph with a cut-point can be transformed into a biconnected graph with greater reliability. The reader can find alternative measure such as the average reliability and its hardness in [10].

Most works in the field of network reliability analysis deal with its evaluation rather than its maximization. The literature on network reliability evaluation is abundant, and here we can mention distinguished works on this field. A trade-off between accuracy and computational feasibility is met by simulations, given the hardness of the classical network reliability models [99]. Macroscopically, Monte Carlo methods consider independent replications of a complex system, and by means of statistical laws find pointwise estimations, in order to make decisions on the system. The reader is invited to consult an excellent book on Monte Carlo methods authored by Fishman [40], which was inspirational for network reliability, numerical integration, statistics and other fields of knowledge. In our particular case we deal with the hostile network reliability model, where both links and non-terminal nodes fail independently. Its reliability evaluation belongs to the class of $\mathcal{NP}$-Hard problems as well [20].

H. Cancela y El Khadiri propose a Monte Carlo-based algorithm for a variance reduction, called Recursive Variance Reduction method, or RVR [17]. This formulation allows a meaningful reduction in variance, and the product between time and variance is also reduced when compared to Crude Monte Carlo. Furthermore, the variance is mathematically proved to be always better in RVR than in CMC. A novel Monte Carlo-based method is proposed in [63], based on a dynamic importance sampling. The goal is to recursively approximate a variance-zero importance sampling estimation, which is adequate for rare event scenarios (i.e., highly-reliable networks in particular). The approximation is based on properties of mincuts. It is worth to remark that this approximate zero-variance proposal achieves the bounded relative error property, meaning that asymptotically, when the rarity of the individual failures tends to zero, the relative error is bounded. Furthermore, it converges to zero under special conditions stated in the article.

In [19] and [23], RVR is combined with Importance Sampling (IS) for static network reliability models. The authors present two estimators: Balanced RVR (considers a uniform distribution to choose the first operational link of a cutset) and Zero-Variance Aproximation RVR tries to imitate the zero-variance estimator [63]. In [18, 13], RVR is extended to a large variety of models, and its variance is again smaller than the one obtained using CMC. In [14], the applicability of RVR is extended to Stochastic Monotone Binary Systems (SMBS), and approximative methods are discuss for the reliability evaluation of SMBS in general. First, two variants of Monte Carlo are presented, and RVR is finally generalized for SMBS. In [22] a novel method RVR-MonteCarlo is presented. This method is based on series-parallel reductions and partitions that consider both pathsets and cutsets to recursively reduce the original problem to an equivalent problem with smaller networks. Good results were obtained for rare events, with a meaningful improvement with respect to state-of-the-art variance reduction methods. In [110], the traditional RVR is combined with integer-programming algorithms to find better cutsets. The accuracy of RVR is improved using this special selection of cutsets.

Chapter 6

Algorithms

A literature review of the potential algorithms to implement was carried out, considering scientific literature and documents provided by the Computer Science institute (InCO) provided by the advisors. The proposal is driven by the following criteria:

1. Simplicity: it must be simple.
2. Efficiency: it must assume reasonable (non-prohibitive) time on a PC.
3. Reproducibility: it should be specified clearly.
4. Effectiveness: it must find globally optimum solutions in most of the instances under study.

It is hard to determine the correct metaheuristic, since the available information is scarce, and the problem addressed in this thesis is novel. A possible approach is to determine simple heuristics that comply the previous criteria in previous problems, related with topological network design. VNS [76, 52, 49, 51, 12, 108, 25, 35, 30, 65, 105, 80, 114, 54] has shown to be efficient in multiple problems (see Chapter 5 for further details). Its effectiveness has been tested over several datasets with optimal or near-optimal solutions, in moderate or at least reasonable times for small and medium-sized instances. A valuable element is its simplicity. This metaheuristics is modern, and it has a lower number of related work in contrast with other metaheuristics. Another valuable fact is that VNS is capable of hybridization or extensions, combining other metaheuristics.

Additionally, the algorithm should be selected in an ad-hoc manner specifically for our problem. The network reliability evaluation under the

hostile model belongs to the class of $\mathcal{NP}$-Hard problems. Consequently, an exact reliability evaluation for large networks is currently prohibitive. A valuable alternative is to consider Monte Carlo-based methods for simulation, such as RVR [74, 16, 21, 19, 63, 17, 18, 14, 13, 23, 22, 110, 115]. Usually, Crude Monte Carlo is the most simple estimation approach for network reliability. However, its variance is large for highly reliable systems, which is our focus. It presents large variance and, consequently, its accuracy does not meet our requirements. This is the reason why we selected RVR [74, 16, 21, 19, 63, 17, 18, 14, 13, 23, 22, 110, 115]. Furthermore, this celebrated method had success in terms of reliability estimation under a large variety of scenarios. Section 6.5 presents outstanding properties of RVR method (see Chapter 5 for further details on its potential applicability to other scenarios). It is worth to mention that RVR works in general for monotone systems, and our hostile network reliability model is monotone. Our main algorithm is presented in Section 6.1. A greedy construction is followed by multiple local searches in a VNS, and the reliability estimation is carried out using RVR.

6.1 Network-Design

NetworkDesign executes different phases that solve the problem of this thesis. Essentially, we can identify three phases:

1. Construction Phase
2. Survivability Phase
3. Reliability Phase

Construction Phase receives the ground graph and returns a feasible solution for the GSPNC[1], which will be introduced in a Survivability phase. This second phase considers a metaheuristic to address the problem, trying to provide an improvement of the received solution in terms of cost, and preserving feasibility. Finally, the Reliability phase is in charge of the reliability estimation. Figure 6.1 presents a pseudocode for *NetworkDesign*. It receives the ground graph G_B, a number of iterations *iter* and a positive integer k to find the k shortest paths during the Construction Phase, a

[1]During this chapter, the feasibility is always related to the relaxed problem with no reliability constraint, this is, the GSPNC.

reliability threshold p_{min} and number of iterations *simiter* during the simulations carried out in the Reliability Phase.

Algorithm 1 $sol = NetworkDesign(G_B, iter, k, p_{min}, simiter)$

```
1: i ← 0; P ← ∅; sol ← ∅
2: while i < iter do
3:    ḡ ← ConstructionPhase(G_B, P, k)
4:    ḡ ← SurvivabilityPhase(ḡ, P)
5:    rel ← ReliabilityPhase(g_sol, simiter)
6:    if rel > p_min then
7:       sol ← sol ∪ {g_sol}
8:    end if
9: end while
10: return sol
```

Figure 6.1: Pseudocode for the main algorithm: *NetworkDesign*.

NetworkDesign will collect all the feasible solutions found in a set *sol*. In Line 1, *sol* is initially the empty set, a counter of iterations is $i = 0$ and a set of paths between the different terminals is also empty. During the *while-loop* of Lines 2-9, the three phases are called in order (Lines 3-5), and a feasibility test takes effect (Lines 6-8). The algorithm returns *sol*, that contains all the feasible solutions obtained during the execution (Line 10).

6.2 Construction

In a trade-off between simplicity and effectiveness, a Greedy Randomized solution has been developed [31, 94]. This algorithm builds a feasible solution based on paths, trying to combine speed and optimality in terms of cost. It builds a graph meeting the connectivity constraints between the terminals $R = \{r_{i,j}\}_{i,j \in T}$. Specifically, given $i, j \in S_D^I$ there exist $r_{i,j}$ node-disjoint paths that connect i and j in the graph. From now on, $S_D^{(I)}$ is the set of terminal nodes, following the terminology of the backbone design from Wide Area Networks. Figure 6.2 receives the ground graph G_B, the matrix with link-costs C, the connectivity matrix R, and the parameter k.

Algorithm 2 $(sol, P) = Greedy(G_B, C, R, k)$

1: $g_{sol} \leftarrow (S_D^{(I)}, \emptyset)$; $m_{i,j} \leftarrow r_{i,j}$; $P_{i,j} \leftarrow \emptyset, \forall i,j \in S_D^{(I)}$; $A_{i,j} \leftarrow 0, \forall i,j \in S_D^{(I)}$
2: **while** $\exists\, m_{i,j} > 0 : A_{i,j} < MAX\_ATTEMPTS$ **do**
3: $(i,j) \leftarrow ChooseRandom(S_D^{(I)} : m_{i,j} > 0)$
4: $\overline{G} \leftarrow G_B \setminus P_{i,j}$
5: **for all** $(u,v) \in E(\overline{G})$ **do**
6: $\overline{c}_{u,v} \leftarrow c_{u,v} \times 1_{\{(u,v) \notin g_{sol}\}}$
7: **end for**
8: $L_p \leftarrow KSP(k, i, j, \overline{G}, \overline{C})$
9: **if** $L_p = \emptyset$ **then**
10: $A_{i,j} \leftarrow A_{i,j} + 1$; $P_{i,j} \leftarrow \emptyset$; $m_{i,j} \leftarrow r_{i,j}$
11: **else**
12: $p \leftarrow Select\_Random(L_p)$; $g_{sol} \leftarrow g_{sol} \cup \{p\}$
13: $P_{i,j} \leftarrow P_{i,j} \cup \{p\}$; $m_{i,j} \leftarrow m_{i,j} - 1$
14: $(P, M) \leftarrow General\_Update\_Matrix(g_{sol}, P, M, p, i, j)$
15: **end if**
16: **end while**
17: **return** (g_{sol}, P)

Figure 6.2: Pseudocode for the Construction Phase: *Greedy*.

In Line 1, the solution g_{sol} is initialized only with the terminal nodes S_D^I without links, $M = \{m_{i,j}\}_{i,j \in T}$ stores the unsatisfied requirements, so initially $m_{i,j} = r_{i,}$ for all $i,j \in S_D^{(I)}$, and the matrix $P = \{P_{i,j}\}_{i,j \in S_D^{(I)}}$ that represents the collection of node-disjoint paths is empty for all $P_{i,j}$. Additionally, the matrix $A = \{A_{i,j}\}_{i,j \in S_D^{(I)}}$ that controls the number of attempts that the algorithm fails to find $r_{i,j}$ node-disjoint paths between i,j is initialized correspondingly: $A_{i,j} = 0\, \forall i,j \in S_D^{(I)}$.

The purpose of the *while-loop* (Lines 2-13) is to fulfill all the connectivity requirements, or detect all the pair of terminals $i,j \in S_D^{(I)}$ that could not fulfill the connectivity requirements during $MAX\_ATTEMPTS$. Each iteration works as follows. A pair of terminals (i,j) is uniformly picked at random from the set $S_D^{(I)}$, provided that $m_{i,j} > 0$ (Line 3). The graph $\overline{G}$ is defined in Line 4 discards the nodes that were already visited in the previous paths. Therefore, if we find some path between i and j in $\overline{G}$, it will be included.

In the *for-loop* of Lines 5-7, an auxiliary matrix with the costs $\overline{C} = \overline{c_{i,j}}$ is defined, where the links belonging to g_{sol} have null cost. This allows to use already existent links from g_{sol} without additional cost, and add them to build

a new node-disjoint path. The k-Shortest Paths from i to j are computed in Line 8 using the costs from the matrix $\overline{C}$. These paths are stored in a list L_p. The function KSP is implemented efficiently using Yen algorithm [72], that finds the k-Shortest Paths between two fixed nodes in a graph. In Line 9, we test if the list L_p is empty. In this case we re-initialize $P_{i,j}$, $m_{i,j}$, and add a unit to $A_{i,j}$, since i and j belong to different connected components. If the list L_p is not empty, a path p is uniformly picked at random from the list L_p, and it is included in the solution (Line 12). The path p is added to $P_{i,j}$, and the requirement $m_{i,j}$ is decreased a unit (Line 13). The addition of the path p could build node-disjoint paths from different terminals. Consequently, the method $General\_Update\_Matrix$ finds these new paths. $Greedy$ returns a feasible solution g_{sol} equipped with all the sets $P = \{P_{i,j}\}_{i,j \in S_D^{(I)}}$ of node-disjoint pairs between the different terminals (Line 17).

The reader can find a proof of feasibility for $Greeedy$, as well as details of the auxiliary functions KSP and $General\_Update\_Matrix$ in the Appendix.

6.3 Local Search

The construction phase does not return even a locally-optimum solution. Therefore, VNS combines different local searches in order to systematically modify the neighborhoods and find a better solution [76, 52, 49, 51, 12, 108, 25, 35, 30, 65, 105, 80, 114, 54]. Here, three different local searches are proposed, each one based on different neighborhood structures. It is worth to mention that only one of them, called $SwapKeyPathLocalSearch$, updates the set of paths P. This seems a subtlety, but it means in fact a major implementation decision.

If we decide to update the set P during a local search, in successive iterations we can either use this set or not, but the risk is to loose the updated version of P in a different local search that considers P. As a consequence, the execution of $SwapKeyPathLocalSearch$ is independent of the iterations which $KeyPathLocalSearch$ and $KeyTreeLocalSearch$ are implied. This is properly illustrated in the following sections.

6.3.1 Local Search 1: KeyPathLocalSearch

Before the detailed description of this local search, some auxiliary concepts and a neighborhood structure are in order. First, recall the concepts of key-node and key-path:

Definition 4 (key-node). *A key-node in a feasible solution $v \in g_{sol}$ is a Steiner (non-terminal) node with degree three or greater.*

Definition 5 (key-path). *A key-path in a feasible solution $p \subseteq g_{sol}$ is an elementary path where all the intermediate nodes are non-terminal with degree 2 in g_{sol}, and the extremes are either terminals or key-nodes.*

Definition 6 (Neighborhood Structure for key-paths). *Given a key-path in a feasible solution $p \in g_{sol}$, a neighbor-solution is $\hat{g}_{sol} = \{g_{sol} \setminus p\} \cup \{\hat{p}\}$, where $\hat{p}$ is other path that connects the extremes from p, and preserves feasibility. The neighborhood of key-paths from g_{sol} is composed by the previous operation to the distinct key-paths belonging to $K_{g_{sol}} = \{p_1, \ldots, p_h\}$, the decomposition of g_{sol} into key-paths.*

KeyPathLocalSearch builds neighbor solutions with an iterative replacement of key-paths with the same key-nodes, preserving feasibility. The process is repeated until no additional improvements are feasible, and a locally-optimum solution is met for this neighborhood system. A pseudocode for *KeyPathLocalSearch* is presented in Figure 6.3. It receives the ground graph G_B, link-costs C and a feasible solution g_{sol}. The variable *improve* is set to TRUE in Line 1. This variable is useful to determine whether there exists an improvement or not during the replacement of key-paths of the algorithm. The *while-loop* of Lines 2-14 looks for neighbor solutions, studying each key-path from the solution g_{sol} and replacing by new key-paths in order to reduce the cost of the global solution, preserving feasibility. Each iteration works as follows. The variable *improve* is set to $FALSE$ in Line 3. The decomposition of g_{sol} into key-paths is found in Line 4. The internal *while-loop* of Lines 5-13 studies the key-paths from $K(g_{sol})$ one-by-one, looking for a cheaper and feasible replacement. A key-path that was not previously studied $p \in K(g_{sol})$ is uniformly picked at random in Line 6. The network $\hat{\mu}$ induced by the nodes $NODES(p) \cup S_D \setminus NODES(g_{sol})$ is computed in Line 7 (recall that S_D is the set of terminal nodes following the

traditional terminology from WAN network design). The set on the right, $\overline{S} = S_D \setminus NODES(g_{sol})$, is precisely the terminal nodes not belonging to g_{sol}.

Observe that $\hat{\mu}$ does not possess nodes from $g_{sol} \setminus p$, except for u and v. Then, all the paths connecting u and v belonging to $\hat{\mu}$ re-establish the feasibility of $g_{sol} \setminus p$. Consequently, the shortest path from u to v in $\hat{\mu}$ is found in Line 8. The cost between $\hat{p}$ and the original path p is compared in Line 9. If $\hat{p}$ is cheaper than p, the key-path p is correspondingly replaced by $\hat{p}$ in g_{sol} (Line 10), and the variable *improve* is set to $TRUE$ (Line 11), in order to re-start the local search from Line 2. On the other hand, if $\hat{p}$ is not cheaper than p, the *while-loop* of Lines 5-13 picks another key-path not studied before, until all key-paths are studied. The process is finished as soon as there are no possible improvements, or there are no key-paths to study. The best neighbor solution g_{sol} is returned in Line 15. A proof that $KeyPathLocalSearch$ preserved feasibility is provided in the Appendix.

Algorithm 3 $g_{sol} = KeyPathLocalSearch(G_B, C, g_{sol})$

```
 1: improve ← TRUE
 2: while improve do
 3:    improve ← FALSE
 4:    K(g_sol) ← {p_1, ..., p_h} {Key-path decomposition of g_sol}
 5:    while not improve and ∃ key-paths not analyzed do
 6:       p ← (K(g_sol)) {Path not analyzed yet, with extremes u and v}
 7:       μ̂ ←< NODES(p) ∪ S_D \ NODES(g_sol) > {Induced subgraph μ̂}
 8:       p̂ ← Dijkstra(u, v, μ̂)
 9:       if COST(p̂) < COST(p) then
10:          g_sol ← {g_sol \ p} ∪ {p̂}
11:          improve ← TRUE
12:       end if
13:    end while
14: end while
15: return g_sol
```

Figure 6.3: Pseudocode for Local Search 1: $KeyPathLocalSearch$.

6.3.2 Local Search 2: KeyTreeLocalSearch

First, a neighborhood structure is in order. Recall the concept of key-tree:

Definition 7 (Key-tree). *Let $v \in g_{sol}$ be a key-node belonging to a feasible solution g_{sol}. The key-tree associated to v, denoted by T_v, is the tree composed by all the key-paths that meet in the common end-point (i.e., the key-node v).*

Definition 8 (Neighborhood Structure for key-tree). *Consider the key-tree $T_v \in g_{sol}$ rooted at the key-node v, where g_{sol} is a feasible solution. A neighbor of g_{sol} is $\hat{g}_{sol} = \{g_{sol} \setminus T_v\} \cup \{T\}$, being T another tree that replaces T_v, with identical leaf-nodes, and preserving feasibility. The neighborhood of g_{sol} is composed by all the neighbor solutions obtained with an iterative application of the previous operations, for the different key-trees belonging to g_{sol}.*

Based on this neighborhood structure, we define a second local search that replaces key-trees (note that the previous local search was a replacement of key-paths).

Algorithm 4 $g_{sol} = KeyTreeLocalSearch(G_B, C, g_{sol})$

```
1:  improve ← TRUE
2:  while improve do
3:     improve ← FALSE
4:     X ← KeyNodes(g_sol) {Key-nodes from g_sol}
5:     S̄ ← S_D \ NODES(g_sol)
6:     while not improve and ∃ key-nodes not analyzed do
7:        v ← X {Key-node not analyzed yet}
8:        [g_sol, improve] ← General_RecConnect(G_B, C, g_sol, v, S̄)
9:     end while
10: end while
11: return g_sol
```

Figure 6.4: Pseudocode for Local Search 2: *KeyTreeLocalSearch*.

Figure 6.4 presents a pseudocode for *KeyTreeLocalSearch*. The rationale is to iteratively build neighbor solutions using a replacement of key-trees, preserving feasibility. The process is repeated until no possible improvement exists. It receives the ground graph G_B, link-costs C and the feasible solution g_{sol}. The variable *improve* is set to $TRUE$ (Line 1). The *while-loop* of

Lines 2-10 looks for better neighbors solutions, studying each key-node from the current solution g_{sol}, and a replacement takes place if corresponds. The *while-loop* is repeated whenever an improvement is found. Each iteration works in the following manner. The variable *improve* is set to $FALSE$ in Line 3. The set X of all the key-nodes from g_{sol} are obtained in Line 4. In Line 5, the set of Steiner nodes $\overline{S}$ not belonging to g_{sol} are computed. The internal *while-loop* of Lines 6-9 studies every key-node belonging to X, together with its associated key-tree, trying to find a cheaper key-tree for replacement. A key-node $v \in X$ is uniformly picked at random in Line 7. The algorithm called *General_RecConnect* is called in Line 8, to find a replacement that is both feasible and cheaper than T_v. The description of *General_RecConnect*, and a proof of feasibility are found in the Appendix.

If this search is successful, this algorithm returns an improved neighbor solution in Line 8, and the solution is updated in the same line. Additionally, the variable *improve* is set to $TRUE$, and the local searches proceeds in Line 2. If *General_RecConnect* fails to find a replacement, the internal *while-loop* considers an alternative key-node not previously studied, or this loop is finished if all the key-nodes were studied.

The process is finished as soon as there are no possible improvements, or there are no key-trees to study. The best neighbor solution g_{sol} is returned in Line 11.

6.3.3 Local Search 3: SwapKeyPathLocalSearch

The following neighborhood structured will be useful:

Definition 9 (Neighborhood Structure for key-path replacement)**.** *Given a key-path $p \subseteq g_{sol}$ from a feasible solution g_{sol}, a neighbor solution for g_{sol} is $\hat{g}_{sol} = \{g_{sol} \setminus p\} \cup \{m\}$, being m the set of nodes and links that will be added to preserve the feasibility of the solution $\hat{g}_{sol}$. The set m could be empty, if the deletion of a key-path from g_{sol} is already feasible. The neighbor of key-paths from g_{sol} is composed by the previous neighbor solutions to each of the different key-paths belonging to the $K(g_{sol}) = \{p_1, \ldots, p_h\}$, the decomposition of g_{sol} into key-paths.*

SwapKeyPathLocalSearch iteratively builds neighbor solutions, removing key-paths and reconstructing a feasible solution using the

information stored in the matrix P (of node-disjoint paths), generated by *Greedy* algorithm during the Construction phase. The process is finished only when no feasible improvements are possible. Figure 6.5 shows a pseudocode for $SwapKeyPathLocalSearch$. It receives the ground graph G_B, the link-costs C, a feasible solution g_{sol} and the matrix with the node-disjoint paths P obtained by the Greedy randomized construction.

The variable *improve* is set to TRUE in Line 1. This variable is useful to determine whether there exists an improvement or not during the replacement of key-paths by paths. The *while-loop* of Lines 2-9 looks for neighbor solutions, studying each key-path from the solution g_{sol} and replacing by nodes and links, or simply deleting the key-path improving the cost, whenever the resulting network is feasible. Each iteration works as follows. The variable *improve* is set to $FALSE$ in Line 3. The decomposition $K(g_{sol})$ of g_{sol} into key-paths is found in Line 4.

The internal *while-loop* of Lines 5-8 studies the key-paths from $K(g_{sol})$ one-by-one, looking for a cheaper and feasible replacement with nodes and links. A key-path that was not previously studied $p \in K(g_{sol})$ is uniformly picked at random in Line 6. The routine $FindSubstituteKeyPath$ is called in Line 7. It deletes the key-path from the current solution and tries to re-connect the extremes by nodes and links, preserving feasibility. If, in addition, the resulting solution is cheaper, the variable *improve* is set to $TRUE$, and the solution is effectively replaced. Otherwise, the following key-path is studied. The process is finished as soon as there are no possible improvements, by replacements from key-paths to nodes and links , or there are no key-paths to study. The best neighbor solution g_{sol} is returned in Line 10. A proof of feasibility for $SwapKeyPathLocalSearch$ and the auxiliary algorithm $FindSubstituteKeyPath$ are found in the Appendix. The diversity of the three local searches explains the effectiveness of our VNS, as we will see in the Results (Chapter 7).

Algorithm 5 $g_{sol} = SwapKeyPathLocalSearch(G_B, C, g_{sol}, P)$

```
1: improve ← TRUE
2: while improve do
3:    improve ← FALSE
4:    K(g_sol) ← {p_1, ..., p_h} {Key-path decomposition of g_sol}
5:    while not improve and ∃ key-paths not analyzed do
6:       p ← (K(g_sol)) {Path not analyzed yet}
7:       (g_sol, improve) ← FindSubstituteKeyPath(g_sol, p, P)
8:    end while
9: end while
10: return g_sol
```

Figure 6.5: Pseudocode for Local Search 3: $SwapKeyPathLocalSearch$.

6.4 Variable Neighborhood Search (VNS)

VNS [76, 52, 49, 51, 12, 108, 25, 35, 30, 65, 105, 80, 114, 54, 50] is supported by a systematic modification of neighborhood structures, hence it requires a finite set of predefined neighborhoods. This represents a major difference with respect to most local search algorithms that use a fixed neighborhood structure. VNS is based on three simple facts:

1. A locally-optimum solution for one neighborhood structure is not necessarily for another one.
2. A globally-optimum solution is locally-optimum under all neighborhood structures.
3. In most problems, a locally optimum solution with respect to one or many neighborhood structures are relatively close.

The last observation is empirical, and it implies that a locally-optima provides information about the globally-optimum solution. Sometimes they share common features. Nevertheless, those features are usually not known. Therefore, it is natural to perform an organized exploration of the vicinities of a local optima, until an improvement is met. The facts 1-3 suggest that several neighborhood structures should be employed to address a combinatorial optimization problem. The change of neighborhood structures can be performed either in a deterministic or a stochastic way, or even in a

hybrid manner. Here, we considered a purely deterministic change of local searches, called Variable Neighborhood Descent or VND [52], which consists of an iterative replacement to the current solution for a better one, whenever an improvement is feasible. If a change of neighborhood structure takes place whenever a locally-optimum solution is met for some structure, a VND is obtained [52].

A general template for VND is presented in Figure 6.6. It receives an objective function f for the combinatorial problem, a feasible solution x and a collection of neighborhood structures $\mathcal{NS}$. Observe that the resulting solution provided by VNS is a locally-optimum solution with respect to all the $k_{max} = |\mathcal{NS}|$ neighborhood structures, and the possibility to reach a globally optimum solution is greater than using a single structure [52]. In the following paragraphs, we detail our VNS implementation.

Algorithm 6 $x = VND(f, x, \mathcal{NS})$

```
1:  improve ← TRUE
2:  while improve do
3:    localsearch ← 1
4:    while localsearch ≤ |NS| do
5:      x' ← BestNeighbor(localsearch, x)
6:      if f(x') < f(x) then
7:        x ← x'; localsearch ← 1
8:      else
9:        localsearch ← localsearch + 1
10:     end if
11:     if localsearch > |NS| then
12:       improve ← FALSE
13:     end if
14:   end while
15: end while
16: return x
```

Figure 6.6: General Template for VND.

Our implementation follows a VNS with minor variations; see Figure 6.7 for a pseudocode. It receives the current solution G and the matrix P with node-disjoint paths from G, previously obtained by our Greedy Randomized construction phase, and a collection of local searches cls.

Algorithm 7 $G = VNS(G, cls, P)$

1: $k \leftarrow 0$; $k_{max} \leftarrow size(cls)$;
2: $G \leftarrow SwapKeyPathLocalSearch(G, P)$
3: $notimprove \leftarrow 0$
4: **while** $notimprove < k_{max}$ **do**
5: $\quad \overline{G} \leftarrow LocalSearch(cls[k], G)$
6: $\quad cost \leftarrow GetCost(G)$
7: $\quad newcost \leftarrow GetCost(\overline{G})$
8: $\quad$ **if** $newcost < cost$ **then**
9: $\quad\quad cost \leftarrow newcost$
10: $\quad\quad notimprove \leftarrow 0$
11: $\quad\quad G \leftarrow \overline{G}$
12: $\quad$ **else**
13: $\quad\quad notimprove \leftarrow notimprove + 1$
14: $\quad$ **end if**
15: $\quad k \leftarrow (k + 1) \, mod \, k_{max}$
16: **end while**
17: **return** G

Figure 6.7: Pseudocode for our specific VNS proposal.

In Line 1, the number of local searches k_{max} and a pointer of current local search are initialized. *SwapKeyPathLocalSearch* is introduced in Line 2, which considers the matrix P as an input. Recall that *SwapKeyPathLocalSearch* is not compatible with the other local searches. For that reason, it is executed at the beginning of the algorithm only once and it is not considered further as a part of the previously mentioned neighborhood structures. Nevertheless, it is included in our VNS implementation, and it is essential to achieve considerable improvements in the initial solutions coming from the Construction phase. In Line 3, the variable *notimprove* is set to 0. In the *while-loop* of Lines 4-15, the local search k takes effect, until there is no feasible improvement (Line 5). If an improvement is achieved, both the cost and the solution are updated (Lines 6-11). Otherwise, the variable *notimprove* is increased a unit (Line 13). In all cases, the following structure is visited in a cyclic-way (Line 15). Here there is a difference with respect to the general template from Figure 6.6, since we proceed with the following structure in a cyclic order, instead of returning to the first structure. This is to avoid the

hierarchy imposed by a traditional VNS scheme, since our local searches are equally relevant. Finally, the resulting solution is returned in Line 17. It is worth to remark that this algorithm can be configured for N arbitrary local searches, that can be parameterized by the collection cls.

6.5 Recursive Variance Reduction (RVR)

The rational behind RVR [74, 16, 21, 19, 63, 17, 18, 14, 13, 23, 22, 110, 115] is to reduce the original problem to a problem with a smaller network that is derived or built from the original. The method is recursive, building several (smaller) networks successively, and it stops when we find a network that is always connected or disconnected, no matter the states of its components. The target is a pointwise estimation for the unreliability Q_K for a given terminal-set K. However, it can be extended to a larger family of stochastic monotone binary systems. The definitions pseudocodes and properties presented in this section are extracted from [74]. The reader can consult [74, 16, 21, 19, 63, 17, 18, 14, 13, 23, 22, 110, 115] for further information. Consider a network $G = (V, E)$ equipped with a terminal set $K \subseteq V$, $v \in V$ and $e \in E$. The following terminology will be used:

- The network is K-connected when for every $u \in K$ and every $v \in K$, there exists a path that connects u and v (operational state).
- $R(G)$ denotes the reliability (the probability that the network is K-connected).
- $Q(G) = 1 - R(G)$ denotes the unreliability.
- $D \subseteq V \cup E$ is an extended K-cut if $G' = (V - D, E - D)$ is not K-connected.
- $G - \{e\}$ is the network whose node-set is V, link-set is $E - \{e\}$ and terminal-set K.
- $G - \{v\}$ is the network whose node-set is $V - \{v\}$, the link-set consists of E minus all the links incident to v, and the terminal-set is now $K - \{v\}$.
- If d is a component (node or link), $G|d$ denotes the derived network, setting the operational probability of d to 1 (d is a perfect component).
- $G * d$ denotes the reduced network, setting the operational probability of d to 1. If $d = e = \{u, v\}$, $G * e$ denotes a link-contraction. If the nodes u and v are identified with the node w, the terminal-set is $K' =$

$K - \{v_1, v_2\} \cup \{w\}$, if $v_1 \in K$ or $v_2 \in K$, or simply $K' = K$ otherwise.

The goal is to build an unbiased pointwise reliability estimator with smaller variance than Crude Monte Carlo [74]. For that purpose, the following properties are considered:

Property 1. *Let $G = (V, E)$ the network equipped with the terminal-set $K = \{v_1, \ldots, v_{|K|}\}$. The following relations hold for the reliability $R(G)$ and unreliability $Q(G)$:*

$$R(G) = \left(\prod_{v \in K} r_v\right) R(G|v_1|v_2| \ldots |v_{|K|}) \tag{6.1}$$

$$Q(G) = 1 - \left(\prod_{v \in K} r_v\right) + \left(\prod_{v \in K} r_v\right) Q(G|v_1|v_2| \ldots |v_{|K|}) \tag{6.2}$$

Property 1 means that the reliability $R(G)$ can be found setting perfect terminal nodes, and multiplying the result of the resulting network by the products of the elementary reliabilities for the terminal nodes.

Property 2. *Let $G = (V, E)$ the network equipped with the terminal-set K, and consider an arbitrary component d. Then: $R(G|d) = R(G * d)$.*

Property 2 means that we can contract perfect components, and the reliability is preserved. The RVR method considers this property in order to successively reduce the size of the network. Consider the structure function $\phi : G \to \{0, 1\}$. For every state of the system, the function ϕ equals 1 if and only if the system is K-operational, or 0 otherwise.

Consider N independent replicas of the system, and the averaging provided by CMC for the unreliability evaluation:

$$\overline{Y} = \frac{1}{N} \sum_{i=1}^{N} (1 - \phi(X - i)) \tag{6.3}$$

Clearly, if we denote $Y = 1 - \phi(G)$, then $E(\overline{Y}) = E(Y) = Q(G)$, and CMC is unbiased for the reliability. The goal of RVR is to build an unbiased random variable $Z(G)$ for the unreliability, with smaller variance than CMC (see Property 3). Such random variable is built using extended K-cuts, and it is expressed as a function of $|D|$ random variables $Y(G_i)$ that corresponds to the states of the original network.

Property 3. *Consider a network $G = (V, E)$ equipped with perfect terminal-set K ($r_v = 1$ for all $v \in K$). Further, consider the following notation:*

- $D = \{d_1, d_2, \ldots, d_{|D|}\}$ *an extended K-cut from G.*
- A_D *the event* all the components from D fail.
- $Q(D) = Pr(A_D) = \prod_{i=1}^{|D|}(1 - r_{d_i})$*: the probability of the event A_D.*
- B_i *the event:* the components from $D_i = \{d_1, \ldots, d_{i-1}\}$ fail but d_i works.
- $G_i = (G - d_1 - \ldots - d_{i-1}) * d_i$.
- V *a random variable independent of $Y(G_i)$ ruled by the probabilities:*

$$Pr(V = v) = Pr(\{B_v\})/(1 - Q_D) = r_{d_v} \prod_{i=1}^{v-1}(1 - r_{d_i})/(1 - Q_D),$$

for all $1 \leq v \leq |D|$.

Then, the following random variable:

$$Z(G) = Q_D + (1 - Q_D) \sum_{i=1}^{|D|} 1_{\{V=i\}} Y(G_i),$$

is unbiased for the unreliability, and presents smaller variance than CMC:

$$E(Z(G)) = Q(G)$$
$$Var(Z(G)) = (Q(G) - Q_D)R(G) \leq Q(G)R(G) = Var(Y(G))$$

The summation $\sum_{i=1}^{|D|} 1_{v=i} Y(G_i)$ corresponds to $Z(G_v)$, which is constructed using an extended K-cut D, and it is expressed in terms of $|D|$ random variables $Y(G_i)$ each corresponding to different states of the original network.

Property 3 states that the random variable Z is unbiased, and it presents smaller variance than the original Y. Therefore, RVR always have smaller variance than CMC. Observe that the collection of the events B_i and A_D is a partition of the possible network states. If the event A_D holds, a cutset is found; otherwise, we can study a derived smaller network. Intuitively, this partition of a random variable into indicator random variables reduce the variance.

Based on the Properties 1-3, the following recursive random variable F is considered:

$$F(G) = \begin{cases} 1 \ if \ \ G \ is \ not \ K-connected \\ 0 \ if \ \ K \ is \ a \ single \ node \\ Q_D + (1-Q_D)\sum_{i=1}^{|D|} 1_{\{V=i\}} F(G_i) \ \ otherwise \end{cases}$$

An independent sample of F for a network G is considered to develop RVR method. The algorithm is detailed in Figure 6.8

Algorithm 8 *RVR*

Input: (G, K, p_v, p_e)

1. **Test**:
   - If G always K-connected, return 0.
   - If G is never K-connected, return 1.
2. Find an extended K-cutset $D = \{d_1, \ldots, d_{|D|}\}$
3. Find Q_D (all the components from D fail)
4. Pick a sample v of the discrete random variable V
5. Build $G_v = G - \{d_1, \ldots, d_{v-1}\} * d_v$
6. Recursive step: return $Q_D + (1 - Q_D)RVR(G_v)$

Figure 6.8: Pseudocode for RVR method.

RVR can be implemented in a variety of network reliability models, such as *all-terminal* R_V, *source-terminal* $R_{s,t}$, and *K-terminal* R_K, among many others (the reader can find a validation of RVR in the Appendix). Further, it is suitable for our hostile network reliability model, where both links and non-terminal (Steiner) nodes may fail. The necessary functionalities for *Graph* class were implemented in order to apply RVR to the structure previously designed. Its implementation in blocks is described in Figure 6.9.

Algorithm 9 $R = Rel(G, seed, N)$

Input: (G, K, p_v, p_e)

1. $s = 0$ *counter for the mean value*
2. $ss = 0$ *counter for the variance*
3. $Set(seed)$
4. **for** $i = 1$ **to** N **do**
   - $G' = G$
   - $x = 1 - RVR(G')$
   - $s = s + x$
   - $ss = ss + x^2$
5. **end for**
6. $esp = s/N$
7. $var = 1/N(N-1)(ss - s^2/N)$
8. **return** (esp, var)

Figure 6.9: Implementation - Graph Class for Reliability evaluation.

The class RVR considers an algorithm to find the reliability estimation using the homonym algorithm (see Figure 6.10). It receives the network, a *seed* for the pseudo-random number generator ($(unsigned)time(0)$ is used) and a number of iterations. Figure 6.10 shows the implementation of RVR, where the variable *terminals* counts the number of terminals in the network. This number is decreased, either by node-elimination or contraction. The function ϕ states whether the terminal-set K belongs to the same component or not (see Figure 6.11). This is verified using depth first search (DFS) algorithm. The node-failure implies a non-operational state, so this test is part of our algorithm. The variable *boolean* is returned accordingly.

The auxiliary functions used in Figure 6.10 are briefly described:

- $GetKExtendedCut$: a terminal node $v \in K$ is picked and considers the set of adjacent nodes and incident links whose elementary reliabilities are strictly smaller than 1, adding those components to D.
- $AllFailedProb$: finds the product of the unreliabilities of all the components belonging to D.
- $GetRandomItem$: considers a uniformly distributed continuous variable in $[0, 1]$ and picks a sample of the discrete random variable V. Then,

Algorithm 10 RVR

Input: (G, K, p_v, p_e)

1. **If** $terminals$=1, **return** 0
2. **Elseif** $\phi(G, K) = 1$, **return** 1.
3. **Else**
4. $D := GetKExtendedCut(G)$
5. $Q_D := AllFailedProb(D)$
6. $index := GetRandomItem(D)$
7. $c := D[index]$
8. $remove(G, D, index - 1)$
9. $add(G, c)$
10. $return\ Q_D + (1 - Q_D) \times RVR(G)$
11. **EndIf**

Figure 6.10: Pseudocode for the RVR implementation.

Algorithm 11 $boolean = \phi(G, K)$

```
1: for all v ∈ K do
2:    if NonOperational(v) then
3:       boolean ← 0
4:    end if
5: end for
6: reached ← 0
7: v ← RandomTerminal(K)
8: reached ← DFS(v)
9: if reached = |K| then
10:   boolean ← 1
11: else
12:   boolean ← 0
13: end if
14: return boolean
```

Figure 6.11: Pseudocode for the Structure Evaluation ϕ.

it returns the corresponding *index* for the component selected from the extended cut D.

- *Remove*: deletes from G all the components until the position $index-1$. If some terminal node is involved in the set, the variable *terminals* is decreased.
- *Add*: sets the elementary reliability of the component to 1 and determines whether it is possible to contract the component or not. If positive, the contraction takes effect. If a link is perfect and some of the adjacent nodes belongs to the terminal-set K, then the identification w belongs to the new terminal-set w. If parallel links appear with elementary reliabilities r_1 and r_2, a single link replaces both parallel links, with the elementary reliability $r = r_1 + r_2 - r_1 r_2$.

We can appreciate how the network is successively reduced step-by-step, since the number of components is reduced in each step, either by eliminations or contractions. Therefore, the number of recursive calls is not greater than $|V| + |E|$. Additionally, the most demanding operation is the rule-evaluation, with order $O(|V|)$. Therefore, the computational order for RVR method is $O(|V| \times (|V| + |E|))$. In order to carry out several independent executions of this simulation, an iterative implementation is considered, The mean value of $F(G)$ is then estimated by an averaging over the sample:

$$E(F) = \frac{1}{N} \sum_{i=1}^{N} F_i,$$

and its variance using the following expression:

$$Var(F) = \frac{1}{N(N-1)} \sum_{i=1}^{N} (F_i - E(F))^2$$

For convenience, it can be computed using unbiased estimations for the first and second moments:

$$Var(F) = \frac{1}{N(N-1)} \left(\sum_{i=1}^{N} F_i^2 - \frac{1}{N} \left(\sum_{i=1}^{N} F_i \right)^2 \right)$$

The mathematical models were extracted from [74].

Chapter 7

Results

7.1 Introduction

In order to understand the effectiveness of this proposal, an extensive computational study was carried out using our main algorithm *NetworkDesign*. Recall that *NetworkDesign* involves a Construction Phase using *Greedy*, followed by a Local Search Phase with VNS (with three local searches, to know, *KeyTreeLocalSearch*, *KeyPathLocalSearch* and *SwapKeyPathLocalSearch*) and the introduction of RVR for the reliability evaluation. The underlying probabilistic model is precisely the hostile model, where both links and Steiner nodes fail [66]. Given the monotonicity of this model, the application of RVR is suitable for this purpose.

The experimental analysis was carried out in a Home-PC (Pentium Core I5, 6GB) running on Windows 2010. We selected $p_{min} = 0.8$ in all the instances under study and $k = 5$ for *Greedy*. The parameter k was selected using preliminary tests under random graphs generated using our algorithm for the construction of test-graphs, looking for different values of k. The value $k = 5$ showed acceptable results for those preliminary tests. Since this thesis proposes a reliability-centric design, it makes no sense to establish a threshold that is lower than 80%. The elementary reliabilities for both Steiner nodes and links are close to the unit. In fact, we are focused on the design of highly-reliable networks.

7.2 Description of the Test-Set

After a literature review, a possibility is to build a test-set for the computational analysis using random graphs. We considered a random graph generation for the different algorithms involved in the solution, and for preliminary tests in the main algorithm $NetworkDesign$.

In order to highlight the effectiveness of our proposal, we finally considered well-known instances from the Travelling Salesman Problem (TSP), extracted from the TSPLIB [88]. To the best of our knowledge, there are no benchmarks available for our particular problem. Therefore, we decided to adapt the instances from a well-known library with full accessibility.

Some instances from TSPLIB were selected, and then they were modified to get complete graphs, with the corresponding euclidean costs on the links. Specifically, we selected the following instances under study: att48, berlin52, brazil58, ch150, d198, eil51, gr137, gr202, kroA100, kroA150, kroB100, kroB150, kroB200, lin105, pr152, rat195, st70, tsp225, u159, rd100 and rd400. Observe that the suffix is the number of nodes in the corresponding instance (e.g., kroA100 has 100 nodes). The first column from Table 7.1 contains the name of the instance. The following columns, from the left to the right, contain respectively:

- $\%\,T$: the percentage of terminal nodes in the graph. We considered 20%, 35% y 50% for our test-set. Then, we have 3 classes of instances for each TSP instance.
- $\%$ Rel: the elementary reliabilities for Steiner nodes and links, respectively.
- $\%$ Req: this is the percentage of pairs (terminal nodes) that should meet a connectivity requirements $r_{i,j} \in \{2, 3, 4\}$, respectively.
- $Iter\_ND$: iterations considered in $NetworkDesign$, according to $\%T$.
- $Iter\_RVR$: iterations considered in RVR method.
- $\#$: number of generated instances.

The constraint imposed by the reliability threshold is carried out in a subset of test-cases, and the corresponding results are detailed in Section 7.3. The abbreviation NA (for non-applicable) appears for those instances were the reliability threshold is not performed. On the other hand, our $Greedy$ construction and the application of VNS takes place over the whole test-set.

If the name includes (E), this means that the instance is a variation of the corresponding instance, with different connectivity requirements. The number of iterations for $NetworkDesign$ is established in $Iter\_ND = 100$ for those instances with relative small CPU times (minutes), and $Iter\_ND \in \{20, 50\}$ for instances with more demanding CPU times. The number of iterations for the RVR method is $Iter\_RVR = 10^4$, selected again using preliminary tests.

Table 7.1: Test-Set

Problem	% T	%Rel	% Req	$Iter\_ND$	$Iter\_RVR$	#
att48	20-35-50	99-95	100-0-0	100-100-100	10^4	3
berlin52	20-35-50	99-95	100-0-0	100-100-100	10^4	3
brazil58	20-35-50	99-95	100-0-0	100-100-100	10^4	3
ch150	20-35-50	99-95	100-0-0	100-100-100	10^4	3
d198	20-35-50	99-95	100-0-0	20-20-20	NA	3
eil51	20-35-50	99-95	100-0-0	100-100-100	10^4	3
gr137	20-35-50	99-95	100-0-0	100-20-20	NA	3
gr202	20-35-50	99-95	100-0-0	100-100-100	10^4	3
kroA100	20-35-50	99-95	100-0-0	100-100-100	NA	3
kroA150	20-35-50	99-95	100-0-0	100-20-20	NA	3
kroB100	20-35-50	99-95	100-0-0	100-100-100	NA	3
kroB150	20-35-50	99-95	100-0-0	100-20-20	NA	3
kroB200	20-35-50	99-95	100-0-0	20-20-20	NA	3
lin105	20-35-50	99-95	100-0-0	100-100-100	NA	3
pr152	20-35-50	99-95	100-0-0	20-20-20	NA	3
rat195	20-35-50	99-95	100-0-0	20-20-20	NA	3
st70	20-35-50	99-95	100-0-0	100-100-100	10^4	3
tsp225	20-35-50	99-95	100-0-0	50-50-50	10^4	3
u159	20-35-50	99-95	100-0-0	20-20-20	NA	3
rd100	20-35-50	99-95	100-0-0	100-100-100	NA	3
rd400	20-35-50	99-95	100-0-0	50-50-50	10^4	3
berlin52(E)	20	99-90	65-25-10	100	10^4	1
eil51(E)	20	99-90	65-25-10	100	10^4	1
att48(E)	35	99-90	65-25-10	100	10^4	1
st70(E)	35	99-90	65-25-10	100	10^4	1
brazil58(E)	50	99-90	65-25-10	100	10^4	1
eil51(E)	50	99-90	65-25-10	100	10^4	1
kroB100(E)	20	99-90	65-25-10	100	10^4	1
lin105(E)	20	99-90	65-25-10	100	NA	1
kroA100(E)	35	99-90	65-25-10	20	10^4	1
rd100(E)	35	99-90	65-25-10	20	NA	1

A second test-set is considered in order to answer strategic questions that

represent the main goals of this thesis: the sensibility of the solution to perturbations in the elementary reliabilities. Different values for the elementary reliabilities for both Steiner nodes and links were used. Specifically, the nine combinations for $p_v, p_e \in \{0.99, 0.97, 0.95\}$ were introduced in different instances, being p_v and p_e the elementary reliabilities for Steiner nodes and links $e = (i, j)$ respectively. The details of the second test-set are presented in Table 7.2.

Table 7.2: Test-Set (2)

Problem	% *T*	% *Req*	*Iter_ND*	*Iter_RVR*	#
att48	20-35-50	100-0-0	100	10^4	3
att48(E)	20	0-100-0	100	10^4	1
att48(E)	20	0-0-100	100	10^4	1
att48(E)	35	65-25-10	100	10^4	1
berlin52	20-35-50	100-0-0	100	10^4	3
berlin52(E)	20	65-25-10	100	10^4	1
brazil58	20-35-50	100-0-0	100	10^4	3
brazil58(E)	50	65-25-10	100	10^4	1
eil51	20-35-50	100-0-0	100	10^4	3
eil51(E)	20	65-25-10	100	10^4	1
eil51(E)	50	65-25-10	100	10^4	1
kroA100	35	100-0-0	100	10^4	1
kroA100(E)	35	65-25-10	100	10^4	1
kroB100	20	100-0-0	100	10^4	1
kroB100(E)	20	65-20-10	100	10^4	1
ch150	20-35-50	100-0-0	100	10^4	3
gr202	20-35-50	100-0-0	100	10^4	3
tsp225	20-35-50	100-0-0	100	10^4	3
rd400	20-35-50	100-0-0	100	10^4	3

For instance, in the first row we can see that we generate 3 instances for att48 with respective percentage of terminal nodes 20-35-50, where the connectivity requirements are $r_{ij} = 2$ (100-0-0). For each instance of att48, one-hundred feasible solutions were found, using 10^4 iterations of the RVR method with the nine possible scenarios of elementary reliabilities: $99-99, 99-97, 99-95, 97-99, 97-97, 97-95, 95-99, 95-97, 95-95$.

It is worth to mention that thousands of hours of CPU times were required for this thesis in order to accomplish this number of generated instances, considering the number of iterations involved in the search of

locally optimum solutions during the VND, and the corresponding reliability evaluation. Section 7.3 reports the numerical results. As a consequence, the answers to the strategic questions of this thesis are provided in Section 7.4.

7.3 Numerical Results

Table 7.3 shows the results for each TSP instance under study. The first column contains the name of the instance. Column 2 shows the percentage of terminal nodes, and the remaining columns present, in order:

- $\%IG$: percentage of improvement of *Greedy* in relation to the original cost of the instance.
- $\%IVNS$: percentage of improvement of VNS, in relation with the output of our *Greedy* construction.
- CPU: average CPU-time per iteration of $NetworkDesign$.
- $\overline{R}$: average for the reliability estimation.
- $\overline{Var}$: average for the estimated variance.

Table 7.3: Numerical Results

Problem	% T	$\%IG$	% $IVNS$	CPU (s)	$\overline{R}$	$\overline{Var}$
att48	20	99.27	34.61	11.466	96.7	7.608E-07
att48	35	98.6	36.83	29.769	94.3	3.448E-06
att48	50	98.22	37.1	65.904	92.7	5.322E-06
berlin52	20	98.98	30.55	30.605	93.7	3.294E-06
berlin52	35	99.06	33.93	33.433	93.8	3.19E-06
berlin52	50	98.02	33.48	106.945	90.7	6.487E-06
brazil58	20	98.92	31.96	62.377	88.5	6.722E-06
brazil58	35	99.25	39.45	68.891	86	8.347E-06
brazil58	50	98.75	35.26	103.553	91	7.093E-06
ch150	20	99.76	37.51	222.552	85.59	1.029E-05
ch150	35	99.72	36.65	546.652	88.03	9.033E-05
ch150	50	99.69	34.42	1203.054	88.8	8.974E-05
d198	20	99.9	32.22	320.142	NA	NA
d198	35	99.86	34.12	2086.376	NA	NA
d198	50	99.81	33.39	5548.639	NA	NA

Table 7.3 – Numerical Results (cont.)

Problem	% T	%IG	% $IVNS$	CPU (s)	$\overline{R}$	$\overline{Var}$
eil51	20	99.34	38.79	14.87	96	1.183E-06
eil51	35	98.54	36.11	39.017	94.2	3.736E-06
eil51	50	98.56	37.32	44.798	93.7	4.284E-06
gr137	20	99.79	36.31	137.496	NA	NA
gr137	35	99.71	34.18	404.061	NA	NA
gr137	50	99.68	34.61	976.369	NA	NA
gr202	20	99.89	32.43	528.162	82.31	1.224E-05
gr202	35	99.75	34.56	3511.698	84.14	1.11E-05
gr202	50	99.74	33.36	9505.629	83.03	1.279E-05
kroA100	20	99.61	36.77	44.225	NA	NA
kroA100	35	99.53	38.23	101.498	88.97	8.525E-05
kroA100	50	99.45	35.89	280.833	NA	NA
kroA150	20	99.83	36.7	102.712	NA	NA
kroA150	35	99.75	36.3	412.97	NA	NA
kroA150	50	99.7	32.32	2035.062	NA	NA
kroB100	20	99.68	38.71	17.301	90.14	6.251E-05
kroB100	35	99.59	36.32	53.74	NA	NA
kroB100	50	99.49	34.98	191.722	NA	NA
kroB150	20	99.84	37.49	112.099	NA	NA
kroB150	35	99.77	36.05	665.676	NA	NA
kroB150	50	99.73	34.53	1327.528	NA	NA
kroB200	20	99.89	36.14	279.156	NA	NA
kroB200	35	99.84	35.06	2234.738	NA	NA
kroB200	50	99.8	33.82	7448.424	NA	NA
lin105	20	99.74	35.89	9.439	NA	NA
lin105	35	99.61	37.04	86.855	NA	NA
lin105	50	99.5	36.4	245.246	NA	NA
pr152	20	99.79	37.14	281.166	NA	NA
pr152	35	99.77	36.86	808.477	NA	NA
pr152	50	99.74	36.88	1673.465	NA	NA
rat195	20	99.88	37.31	280.948	NA	NA
rat195	35	99.82	34.7	1925.985	NA	NA
rat195	50	99.8	34.99	4599.873	NA	NA

Table 7.3 – Numerical Results (cont.)

Problem	% T	%IG	% $IVNS$	CPU (s)	$\overline{R}$	$\overline{Var}$
st70	20	99.44	39.84	39.852	91.9	4.072E-06
st70	35	99.3	39.56	63.65	90.6	5.743E-06
st70	50	99.16	36.37	128.195	91.3	7.027E-06
tsp225	20	99.88	34.98	1658.773	84.75	1.141E-05
tsp225	35	99.85	34.65	4684.367	84.64	1.249E-05
tsp225	50	99.82	33.26	12088.726	87.19	1.092E-05
u159	20	99.81	35.84	333.263	NA	NA
u159	35	99.76	36.14	864.992	NA	NA
u159	50	99.75	35.61	1278.13	NA	NA
rd100	20	99.68	37.15	22.421	NA	NA
rd100	35	99.5	34.54	126.822	NA	NA
rd100	50	99.42	36.13	245.827	NA	NA
rd400	20	99.94	35.84	88.214	80.94	14.22E-05
rd400	35	99.94	33.54	504.103	85.37	11.89E-05
rd400	50	99.93	33.16	980.701	86.43	11.51E-05
berlin52(E)	20	98.45	25.25	34.209	99.3	4.848E-07
eil51(E)	20	98.47	28.45	29.623	99.6	2.707E-07
att48(E)	35	97.45	31.74	62.967	99.4	4.93E-07
st70(E)	35	98.52	31.87	135.508	99.3	6.549E-07
brazil58(E)	50	97.48	31.84	172.636	99.4	4.825E-07
eil51(E)	50	97.26	32.67	74.473	99.1	7.942E-07
kroB100(E)	20	99.37	30.25	39.255	99.87	1.219E-06
lin105(E)	20	99.33	31.95	64.409	NA	NA
kroA100(E)	35	98.99	35.88	225.505	99.81	1.828E-06
rd100(E)	35	99.15	35.3	130.008	NA	NA
Average	NA	99.39	35.03	1026.003	91.38	2.33E-05

The improvement of VNS over the constructed solution in Greedy, $IVNS$, is bounded between 25,25% and 39,84%, according to the instance an its characteristics on the test-set.

The minimum threshold $p_{min} = 0.8$ is widely exceeded in all the instances where the average reliability $\overline{R}$ was estimated. For those instances in which the elementary reliabilities were established in 99%-95% respectively for nodes and links, the range of reliabilities is bounded between 82,31%-96,7%, while a range of 99,1% y 99,87% is observed with elementary reliabilities 99%-90%.

The estimated variance is reduced in average in all the instances under study. This suggests that the RVR method is accurate, even under reliability failures of $q = 10^{-2}$ for both Steiner nodes and links. This fact is discussed in Section 7.4. In general, the CPU times are non-prohibitive and in general are acceptable under all the test-set. In Section 7.3.1, illustrative examples of resulting networks returned by *NetworkDesign* are presented.

7.3.1 Resulting Topologies

Brazil58

Consider Brazil58 instance with at least %20 of terminal nodes, elementary reliability in Steiner nodes %99, link-reliabilities %95 and %100 of 2 node-disjoint paths between each pair of terminal nodes as the connectivity requirement. Figures 7.1 and 7.2 show correspondingly the ground graph G_B and the output of *NetworkDesign*. The resulting cost is 25106, and the reliability is 0.917399. In Figure 7.1, the ground graph is illustrated, and the the excluded nodes and links are colored in grey. In Figure 7.2, red nodes are the terminals that do not fail, and Steiner nodes are represented in orange. To simplify, all the links have unit-cost.

Berlin52

Consider Berlin52 instance under identical conditions, with at least %20 of terminal nodes, elementary reliability in Steiner nodes %99, link-reliabilities %95 and %100 of 2 node-disjoint paths between each pair of terminal nodes as the connectivity requirement. Figures 7.3 and 7.4 show correspondingly the ground graph G_B and the output of *NetworkDesign*. The resulting cost is 4534.109370, and the reliability is 0.844772. In Figure 7.3, the ground graph is illustrated, and the excluded nodes and links are colored in grey. In Figure 7.4,

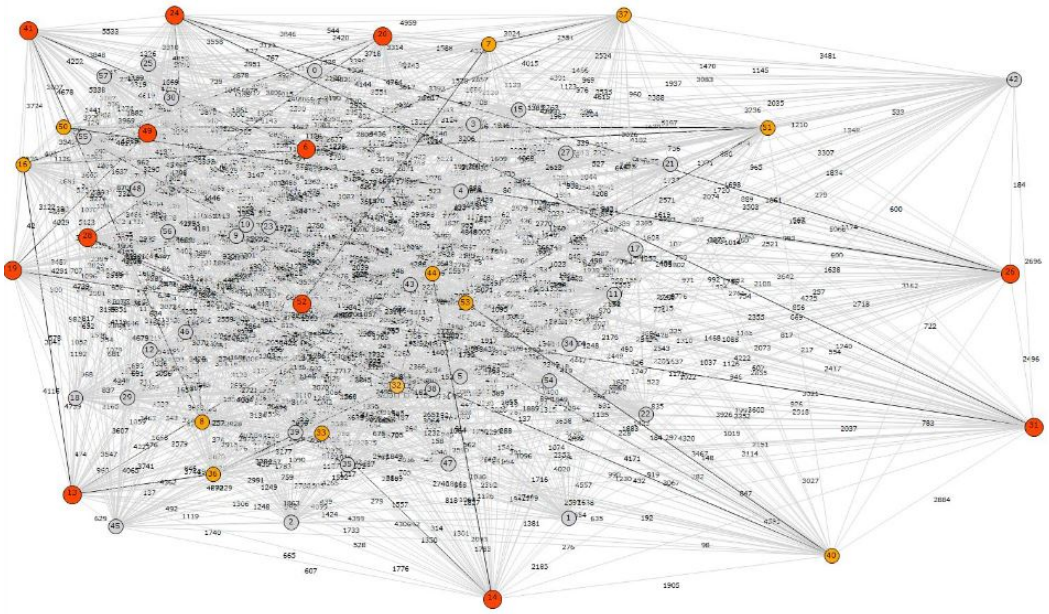

Figure 7.1: Brazil58: ground graph G_B.

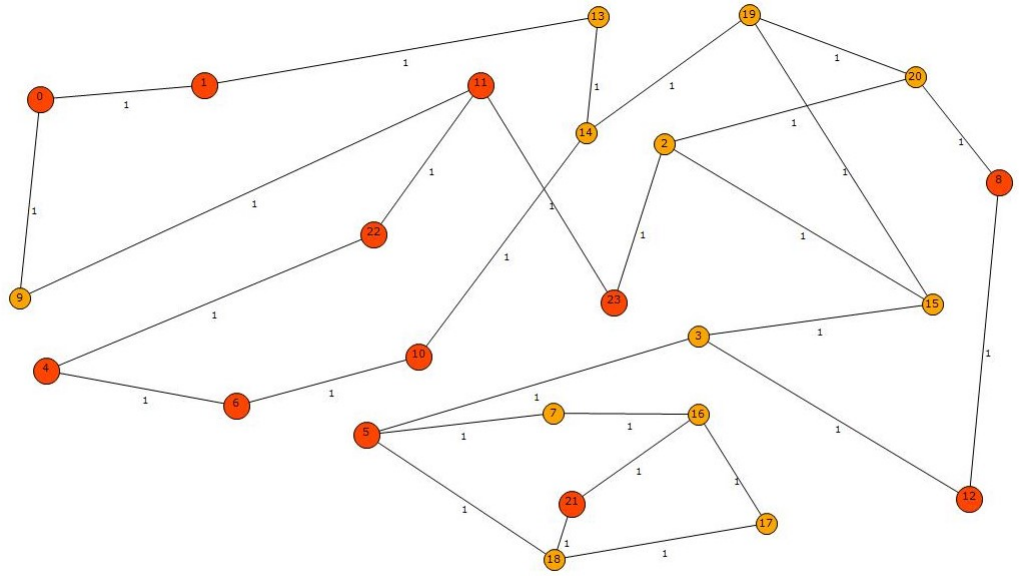

Figure 7.2: Brazil58: resulting topology.

red nodes are the terminals that do not fail, and Steiner nodes are represented in orange. To simplify, all the links have unit-cost.

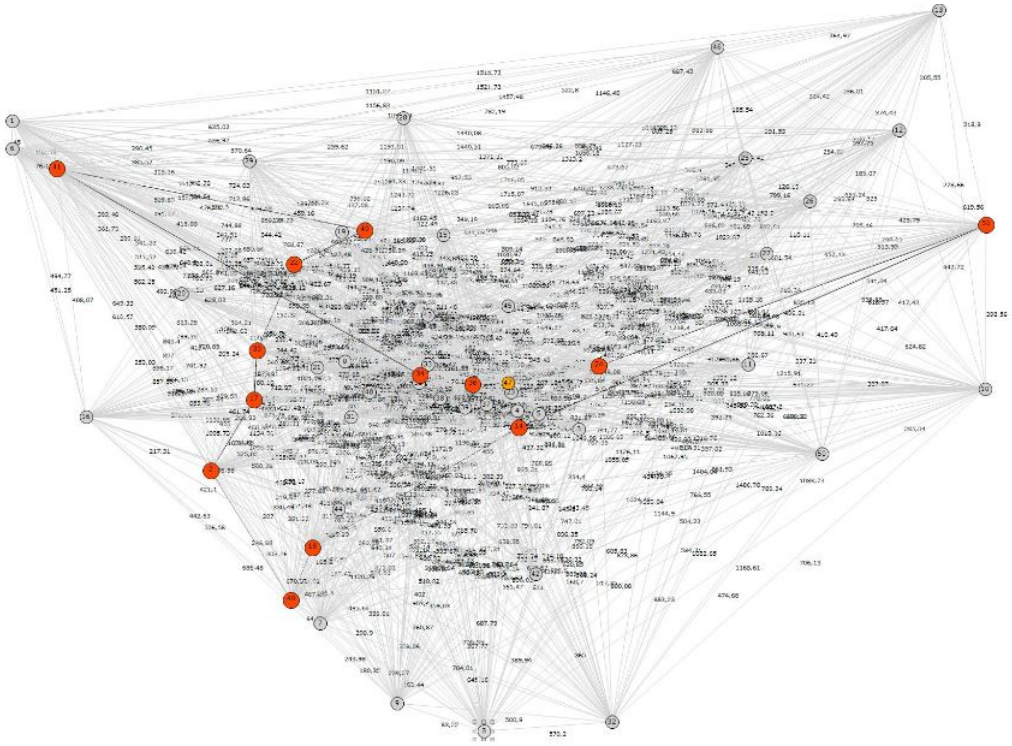

Figure 7.3: Berlin52: ground graph G_B.

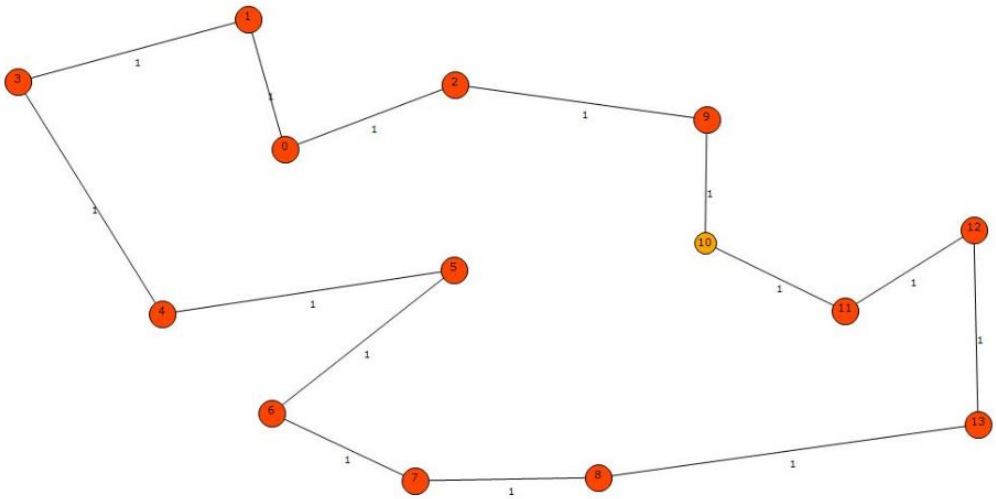

Figure 7.4: Berlin52: resulting topology.

7.4 Key Questions

Our particular interest is to answer the following questions that relate the network optimization and reliability evaluation stages that were addressed throughout this work.

Question 1. *How many feasible networks there exists given the full probabilistic model* (p_{min}, P_E, P_{V-T})*?*

We will restrict our attention to the number of feasible networks that we *can find*, for specific values in the probabilistic model. Nevertheless, this information gives us an insight of the level of feasibility provided by our main algorithm *NetworkDesign*, which serves as a guide, and to offer partial answers. Consider $p_{min} = 0.98$ and different percentage of solutions that survived for the pair of identical elementary reliabilities in Steiner nodes and links %99-%99. The respective percentages under different instances is presented in Table 7.4. We can appreciate that, in general, the number of solutions that meet the reliability threshold is high. Indeed, it is 100% of the returned solutions in most instances.

Question 2. *What is the sensibility of the model with respect to the elementary reliabilities? For instance, for any given threshold* ($p_{min} = 0.98$)*, what happens if we fix* $p_v = 0.99$ *but we pick different values for the elementary link reliabilities* $p_e \in \{0.99, 0.97, 0.95\}$*? How many feasible networks survive? Analogously, if we fix* $p_e = 0.99$ *and* $p_v \in \{0.99, 0.97, 0.95\}$*.*

Consider the threshold $p_{min} = 0.98$. Tables 7.5 and 7.6 present the percentage of feasible solutions under different scenarios.

On one hand, if we fix the node-reliability and reduce the elementary link-reliabilities, a notorious reduction in the number of feasible solutions is observed, even reaching 0 in almost-all the instances under study when the link-reliabilities are %95. It is worth to note that the feasibility is also 0 specially in large networks when the link reliabilities are %97. This pattern for the results is reasonable, since the number of links is also increased when the size of the network is larger.

On the other hand, if we fix the link-reliabilities and reduce the node-reliabilities, the reduction in the reliability is less important than in the previous setting, in terms of the percentage of feasible solutions. We can

Table 7.4: Percentage of feasible solutions such that $R \geq 0.98$

Rel %99-%99	% Feasible solutions with $R \geq 0.98$
att48 T20	100
att48 T35	100
att48 T50	100
eil51 T20	100
eil51 T35	100
eil51 T50	100
berlin52 T20	100
berlin52 T35	100
berlin52 T50	100
brazil58 T20	99
brazil58 T35	97
brazil58 T50	100
ch150 T20	100
ch150 T35	100
ch150 T50	100
gr202 T20	99
gr202 T35	100
gr202 T50	100
tsp225 T20	100
tsp225 T35	100
tsp225 T50	100
rd400 T20	100
rd400 T35	100
rd400 T50	100

Table 7.5: Solutions such that $R \geq 0.98$ (p_v fixed)

Rel %99-%x	%99-%99	%99-%97	%99-%95
att48 T20	100	90	12
att48 T35	100	53	0
att48 T50	100	20	0
berlin52 T20	100	41	0
berlin52 T35	100	50	0
berlin52 T50	100	1	0
brazil58 T20	99	15	0
brazil58 T35	97	0	0
brazil58 T50	100	5	0
ch150 T20	100	0	0
ch150 T35	100	0	0
ch150 T50	100	0	0
gr202 T20	99	0	0
gr202 T35	100	0	0
gr202 T50	100	0	0
rd400 T20	100	0	0
rd400 T35	100	0	0
rd400 T50	100	0	0

appreciate that if the number of terminal-nodes is decreased, the corresponding reduction in the percentage of feasible solutions that exceed the threshold p_{min} is not as steep as in the previous case. This is coherent, since larger networks have more perfect terminal nodes (recall that Steiner nodes are optional). Nevertheless, the global effect of network expansion is a corresponding reduction in the percentage of solutions that exceed the reliability threshold, since the number of Steiner nodes is increased. This reduction is not as steep as in the previous case.

Question 3. *How many networks survive on average, for any given probabilistic model? Understand the sensibility of the model with respect to the connectivity requirements* $r_{i,j} \in \{2, 3, 4\}$.

Consider $p_{min} = 0.98$. The notation *name* TXX *(*$Z1-Z2-Z3$*)* means the name of the graph, TXX is the percentage of terminal-nodes and $Z1-Z2-Z3$ the percentage of terminal nodes with connectivity requirements 2, 3 and 4 respectively. For example, *eil51 T50 (65-25-10)* means that the instance is eil51, with 50% of terminal nodes, where 65% have connectivity requirement 2, 25% have requirement 3 and 10% have requirement equal to 4.

Table 7.6: Solutions such that $R \geq 0.98$ (p_e fixed)

Rel %x-%99	%99-%99	%97-%99	%95-%99
tt48 T20	100	100	99
att48 T35	100	98	96
att48 T50	100	100	99
berlin52 T20	100	100	80
berlin52 T35	100	99	93
berlin52 T50	100	100	100
brazil58 T20	99	59	41
brazil58 T35	97	43	9
brazil58 T50	100	99	81
ch150 T20	100	60	20
ch150 T35	100	98	76
ch150 T50	100	100	97
gr202 T20	99	80	30
gr202 T35	100	69	16
gr202 T50	100	100	76
rd400 T20	100	16	2
rd400 T35	100	98	80
rd400 T50	100	100	100

From Table 7.7, we can appreciate that an increase in the network connectivity requirements necessarily imply a corresponding increase in the percentage of networks that meet the reliability threshold, and vice-versa. This is a nice interplay between topological network design and network reliability analysis: a more *robust* network in a deterministic manner (node-disjoint paths) is translated into a most-reliable network (under probabilistic models), and vice-versa.

Question 4. *Is it better to improve the elementary reliability of links, or the reliability of Steiner nodes, in order to meet a demanding reliability threshold?*

We can appreciate from Tables 7.5 and 7.6 that an increase in the link-reliabilities have a better impact than a corresponding increase in node-reliabilities.

Table 7.7: Solutions such that $R \geq 0.98$ (case %99-%97)

Rel %99-%97	% Feasible solutions with $R \geq 0.98$
att48 T20 (100-0-0)	90
att48 T20 (65-25-10)	100
att48 T20 (0-100-0)	100
att48 T20 (0-0-100)	100
eil51 T20 (100-0-0)	76
eil51 T20 (65-25-10)	100
eil51 T50 (100-0-0)	54
eil51 T50 (65-25-10)	100
berlin52 T20 (100-0-0)	41
berlin52 T20 (65-25-10)	100
brazil58 T50 (100-0-0)	5
brazil58 T50 (65-25-10)	100
kroA100 T35 (100-0-0)	0
kroA100 T35 (65-25-10)	100
kroB100 T20 (100-0-0)	3
kroB100 T20 (65-25-10)	100

Chapter 8

Conclusions

The object under study in this thesis is the topological design of highly reliable networks. Our goal is to combine purely deterministic aspects such as topological network design with probabilistic models coming from network reliability. For that purpose, the Generalized Steiner Problem with Node-Connectivity Contraints and Hostile Reliability (GSP-NCHR) is here introduced. The GSP-NCHR belongs to the class of $\mathcal{NP}$-Hard problems [46], since it subsumes the Generalized Steiner Problem (GSP). Therefore, the CPU-times are prohibitive, even for medium and small-sized networks. This promotes the development of approximative methods for its solution. For that reason, we considered a heuristic solution. Variable Neighborhood Search (VNS) was selected mostly because of its simplicity, flexibility and effectiveness (the reasons are detailed in Chapter 3). It is worth to remark that the network reliability evaluation under the hostile model also belongs to the $\mathcal{NP}$-Hard class. Therefore, we adopted an outstanding pointwise reliability estimation, known as Recursive Variance Reduction (RVR) method, which can be applied in general to arbitrary Stochastic Monotone Binary Systems. Since the hostile model is monotone, RVR is suitable for this model (a more detailed justification is provided in Chapter 3). The object-oriented language C++ was considered [61] for the implementation of the whole developed algorithms in this thesis, which includes a representation of random graphs, validation, and testing different algorithms.

To the best of our knowledge, the GSP-NCHR is presented for the first time in this thesis. In fact, the related work that simultaneously addresses a topological network optimization meeting reliability constraints is scarce.

Therefore, no benchmarks for this problem are available in the scientific literature. In order to study the effectiveness of our heuristic, we adapted instances taken from TSPLIB [88]. The improvement provided by the VNS phase after Greedy Construction ranges between 25.25% and 39.84%, depending on the instance under study and its characteristics. This improvement is satisfactory, for all the instances under study. In real-life scenarios, this means a notorious economical saving. The average reliability for all the networks range between 82.31% and 99.87%, depending on the elementary reliabilities for Steiner-nodes and links, and the connectivity-requirements. The estimated variance was always small, even under highly-reliable scenarios, showing the accuracy of RVR. In fact, the simple approach provided by CMC fails to estimate the reliability for highly-reliable scenarios, providing the incorrect value of unit reliability and zero variance [74]. The networks here proposed meet the minimum reliability requirement, and feasible solutions were always returned (the reader can find the numerical results in Chapter 7). The CPU-times per-iteration is acceptable, since the time is non-prohibitive, even for large-sized instances. It is fair to remark here that the network reliability estimation using RVR is not considered in this time for some instances.

In order to answer the strategic questions of the thesis, several remarks are in order. When the elementary reliability of both Steiner and links is high (99%-99%), the percentage of networks that achieve the reliability threshold is high, being 100% for most of the instances under study. On one hand, when we fix the node-reliabilities but the elementary reliabilities are dropped, we can appreciate an important degradation of this percentage, meeting 0 in almost-all instances when the link-reliabilities are 95%. A similar degradation occurs for large-sized instances when the elementary reliabilities are degraded (99%-97%). On the other hand, when only the elementary reliabilities of Steiner nodes are degraded, the percentage of resulting networks that fulfill the reliability threshold is not rapidly deteriorated. In summary, the network reliability is more sensible to link-reliabilities. When the number of terminal-nodes is increased, the number of solutions that meet the reliability threshold is greater. This is coherent, since the number of perfect nodes is increased. A reduction in the reliability can be observed for larger networks, since the number of Steiner nodes is increased. Finally, we can conclude that when the connectivity requirements are increased, the resulting networks present greater reliability.

Chapter 9

Future Work

The interplay between topological network design and network reliability is not well understood yet. In this thesis some local searches were proposed, essentially using key-path and key-tree replacements, in order to reduce costs preserving feasibility. A current research line is to develop strong reliability-increasing transformations, that replace links and/or paths in order to increase the reliability of the resulting network. The development of local searches that increase reliability and reduce costs would enrich the current solution, and it is part of future work. In terms of implementation of the algorithms and experimental analysis, the following elements are relevant for the author of this thesis:

- Variable neighborhood search is both simple and powerful. The possibility to introduce VNS in another metaheuristic in a hybrid manner [52, 49] is an interesting development to empower the solution of our network optimization problem. Tabu Search or TS [76, 75] generally considers a neighborhood search, exploiting different memory-types and movements. At first, there are two ways to combine VNS with TS: using some memory-type to guide the search during VNS, or to include VNS in TS. GRASP methodology [24, 34, 76, 90, 93, 94, 101, 112, 95, 58, 92, 89, 71, 84, 38, 37, 32, 2, 86, 11, 68, 105, 80] could be also used together with VNS, and it results a hybrid metaheuristic attractive for future work.
- VNS is based on a systematic modification of the neighborhood during the search, and it requires a finite set of neighborhood structures. Here, we considered three local searches: $KeyPathLocalSearch$,

KeyTreeLocalSearch and *SwapKeyPathLocalSearch*. Another local search in order to enrich our VNS proposal is also a hint for future work.

- Several proposals extend VNS, providing new characteristics. For the resolution of large-sized instances, we can find Variable Neighborhood Decomposition Search (VNDS), Biased VNS (BVNS) and Parallel Variable Neighborhood Search (PVNS). The study of applicability to the current problem will empower the optimization algorithm.
- A fair comparison with another metaheuristic, such as GRASP, using an identical test-set is desirable.
- An optimization of the algorithms developed in this thesis using parallel computing would achieve better CPU-times and find exact solutions for large-sized instances.

Bibliography

[1] Agrawal, A., Klein, P., and Ravi, R. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing 24*, 3 (1995), 440–456.

[2] Alkaabneh, F., Diabat, A., and Elhedhli, S. A lagrangian heuristic and grasp for the hub-and-spoke network system with economies-of-scale and congestion. *Transportation Research Part C: Emerging Technologies 102* (2019), 249 – 273.

[3] Archer, K., Graves, C., and Milan, D. Classes of uniformly most reliable graphs for all-terminal reliability. *Discret. Appl. Math. 267* (2019), 12–29.

[4] Arraga, S., and Aroztegui, M. Algoritmos Genéticos Paralelos para el Problema General de Steiner en Grafos, 2002.

[5] Barrera, J., Cancela, H., and Moreno, E. Topological optimization of reliable networks under dependent failures. *Oper. Res. Lett. 43*, 2 (2015), 132–136.

[6] Bienstock, Brickell, M. On the structure of minimum-weight k-connected spanning network. *Society for industrial and applied mathematics 3*, 3 (1990), 320–329.

[7] Boesch, F., Satyanarayana, A., and Suffel, C. A survey of some network reliability analysis and synthesis results. *Networks 54*, 2 (2009), 99–107.

[8] Botton, Q., Fortz, B., and Gouveia, L. On the hop-constrained survivable network design problem with reliable edges. *Computers and Operations Research 64* (2015), 159 – 167.

[9] Brown, J. I., and Cox, D. Nonexistence of optimal graphs for all terminal reliability. *Networks 63*, 2 (2014), 146–153.

[10] Brown, J. I., Cox, D., and Ehrenborg, R. The average reliability of a graph. *Discret. Appl. Math. 177* (2014), 19–33.

[11] Bruni, M., Beraldi, P., and Khodaparasti, S. A hybrid reactive grasp heuristic for the risk-averse k-traveling repairman problem with profits. *Computers and Operations Research 115* (2020), 104854.

[12] Brusco, M. J., and Doreian, P. Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search. *Social Networks 56* (2019), 70 – 80.

[13] Canale, E., Cancela, H., Romero, P., and Robledo, F. Recursive variance reduction in reliability analysis. Tech. rep., Instituto de Computación., Facultad de Ingeniería. Universidad de la República. Montevideo, Uruguay., 2014. Technical Report 14-15 UR.FI-INCO.

[14] Canale, E. A., Cancela, H., Piccini, J., Robledo, F., Romero, P., Rubino, G., and Sartor, P. Recursive Variance Reduction method in Stochastic Monotone Binary Systems. *Proceedings of the 7th International Workshop on Reliable Networks Design and Modeling (RNDM)* (2015), 135–141.

[15] Canale, E. A., Robledo, F., Romero, P., and Viera, J. Building reliability-improving network transformations. In *Proceedings of the 15th International Conference on the Design of Reliable Communication Networks* (2019), IEEE, pp. 107–113.

[16] Cancela, H. Adapting rvr simulation techniques for general network reliability models. Tech. rep., Investigación Operativa – InCo – Pedeciba Informática, Facultad de Ingeniería. Universidad de la República. Montevideo, Uruguay., 1999. Technical Report 00-02 INCO 99.05.

[17] Cancela, H., and El Khadiri, M. The recursive variance-reduction simulation algorithm for network reliability evaluation. *IEEE Transactions on Reliability 52*, 2 (2003), 207–212.

[18] Cancela, H., El Khadiri, M., Rubino, G., and Tuffin, B. A Recursive Variance Reduction Technique with Bounded relative Error for Communication Network Reliability Estimation. In *6th St Petersburg Workshop in Computer Simulation* (Saint Petersburg, Russia, 2009).

[19] Cancela, H., El Khadiri, M., Rubino, G., and Tuffin, B. Recursive Variance Reduction Estimators for the Static Communication Network Reliability Problem. In *8th International Workshop on Rare Event Simulation (RESIM 2010)* (United Kingdom, 2010).

[20] Cancela, H., Guerberoff, G., Robledo, F., and Romero, P. Reliability maximization in stochastic binary systems. In *Proceedings of the 21st Conference on Innovation in Clouds, Internet and Networks and Workshops* (2018), IEEE, pp. 1–7.

[21] Cancela, H., and Khadiri, M. E. Series parallel reductions in rvr reliability evaluation. Tech. rep., Investigación Operativa – InCo – Pedeciba Informática, Facultad de Ingeniería. Universidad de la República. Montevideo, Uruguay., 1996. Technical Report INCO 96-01.

[22] Cancela, H., Khadiri, M. E., and Rubino, G. A new simulation method based on the RVR principle for the rare event network reliability problem. *Annals of Operations Research 196*, 1 (2012), 111–136.

[23] Cancela, H., Khadiri, M. E., Rubino, G., and Tuffin, B. Balanced and approximate zero-variance recursive estimators for the network reliability problem. *ACM Transactions on Modeling and Computer Simulation 25*, 1 (2014).

[24] Cancela, H., Robledo, F., and Rubino, G. Network design with node connectivity constraints. In *Proceedings of the 2003 IFIP/ACM Latin America Conference on Towards a Latin American Agenda for Network Research* (New York, NY, USA, 2003), Association for Computing Machinery, p. 13–20.

[25] Carrizosa, E., Mladenović, N., and Todosijević, R. Variable neighborhood search for minimum sum-of-squares clustering on networks. *European Journal of Operational Research 230*, 2 (2013), 356 – 363.

[26] Chen, Z.-Z. Approximating unweighted connectivity problems in parallel. *Information and Computation 171*, 2 (2001), 125 – 136.

[27] Cheriyan, J., Jordan, T., and Nutov, Z. On rooted node-connectivity problems. *Algorithmica 30*, 3 (2001), 353–375.

[28] Cheriyan, J., Sebo, A., and Szigeti, Z. Improving on the 1.5-approximation of a smallest 2-edge connected spanning subgraph. *SIAM Journal on Discrete Mathematics 14*, 2 (2001), 170–180.

[29] Chimani, M., Kandyba, M., Ljubić, I., and Mutzel, P. Strong formulations for 2-node-connected steiner network problems. In *Proceedings of the 2nd International Conference on Combinatorial Optimization and Applications* (Berlin, Heidelberg, 2008), Springer-Verlag, p. 190–200.

[30] Consoli, S., and Pérez, J. A. M. Variable neighbourhood search for the k-labelled spanning forest problem. *Electronic Notes in Discrete Mathematics 47* (2015), 29 – 36. The 3rd International Conference on Variable Neighborhood Search (VNS'14).

[31] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009. Chapter 16, Greedy Algorithms.

[32] Cravo, G., and Amaral, A. A grasp algorithm for solving large-scale single row facility layout problems. *Computers and Operations Research 106* (2019), 49 – 61.

[33] Czumaj, A., and Lingas, A. On approximability of the minimum-cost k-connected spanning subgraph problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms* (USA, 1999), SODA '99, Society for Industrial and Applied Mathematics, p. 281–290.

[34] de Kerebi, G. I. F., Maneyero, M., Robledo, F., and Sabiguero, A. Modelo de Confiabilidad en Redes, 1996.

[35] Dib, O., Moalic, L., Manier, M.-A., and Caminada, A. An advanced ga–vns combination for multicriteria route planning in public transit networks. *Expert Systems with Applications 72* (2017), 67 – 82.

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

[37] dos Santos Dantas, A. P., [de Souza], C. C., and Dias, Z. A grasp for the convex recoloring problem in graphs. *Electronic Notes in Theoretical Computer Science 346* (2019), 379 – 391. The proceedings of Lagos 2019, the tenth Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS 2019).

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

[39] Fernandes, C. G. A better approximation ratio for the minimum k-edge-connected spanning subgraph problem. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (USA, 1997), SODA '97, Society for Industrial and Applied Mathematics, p. 629–638.

[40] Fishman, G. *Monte Carlo.* Springer Series in Operations Research and Financial Engineering. Springer, 1996.

[41] Ford, D. R., and Fulkerson, D. R. *Flows in Networks.* Princeton University Press, USA, 2010.

[42] Gabow, H. N., Goemans, M. X., and Williamson, D. P. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming 82* (1998), 13–40.

[43] Gabow, H. N., Goemans, M. X., and Williamson, D. P. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming 82* (1998), 13–40.

[44] Galluccio, and Proietti. Polynomial time algorithms for 2-edge-connectivity augmentation problems. *Algorithmica 36*, 4 (2003), 361–374.

[45] Gang Cheng, and Ansari, N. Finding all hops k-shortest paths. In *Proceedings of the IEEE Pacific Rim Conference on Communications Computers and Signal Processing* (2003), vol. 1, pp. 474–477.

[46] Garey, M. R., and Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., USA, 1990.

[47] Goemans, M. X., Goldberg, A. V., Plotkin, S., Shmoys, D. B., Tardos, E., and Williamson, D. P. Improved approximation algorithms for network design problems. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (USA, 1994), SODA '94, Society for Industrial and Applied Mathematics, p. 223–232.

[48] Goemans, M. X., and Williamson, D. P. A general approximation technique for constrained forest problems. *SIAM Journal on Computing 24*, 2 (1995), 296–317.

[49] Gomes, L. M., Diniz, V. B., and Martinhon, C. A. A Tutorial on Variable Neighborhood Search, 1970.

[50] Hansen, P. *Variable Neighborhood Search*. Springer International Publishing, Handbook of Heuristics, 2018.

[51] Hansen, P., and Mladenovic, N. An Hybrid GRASP+VNS Metaheuristic for the Prize-Collecting Traveling Salesman Problem, 1970.

[52] Hansen, P., Mladenovic, N., and Pérez, J. A. M. Variable neighbourhood search. *Revista Iberoamericana de Inteligencia Artificial 7*, 19 (2003), 77–92.

[53] Harary, F. The maximum connectivity of a graph. *Proceedings of the National Academy of Sciences 48*, 7 (1962), 1142–1146.

[54] Hore, S., Chatterjee, A., and Dewanji, A. Improving variable neighborhood search to solve the traveling salesman problem. *Applied Soft Computing 68* (2018), 83 – 91.

[55] Jain, K. A factor 2 approximation algorithm for the generalized steiner network problem. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science* (USA, 1998), FOCS '98, IEEE Computer Society, p. 448.

[56] Kamma, L., and Nutov, Z. Approximating survivable networks with minimum number of steiner points. In *Proceedings of the 8th International Conference on Approximation and Online Algorithms* (Berlin, Heidelberg, 2011), Springer-Verlag, p. 154–165.

[57] Karp, R. M. *Reducibility among Combinatorial Problems.* Springer US, Boston, MA, 1972, pp. 85–103.

[58] Kerivin, H., and Mahjoub, A. R. Design of survivable networks: A survey. *Networks 46*, 1 (2005), 1–21.

[59] Kleywegt, A. J., Shapiro, A., and Homem-de-Mello, T. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization 12*, 2 (2002), 479–502.

[60] Kortsarz, G., Krauthgamer, R., and Lee, J. R. Hardness of approximation for vertex-connectivity network design problems. *SIAM Journal on Computing 33*, 3 (2004), 704–720.

[61] Larman, C. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Prentice Hall, Inc., USA, 2004.

[62] Lawler, E. L. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science 18*, 7 (1972), 401–405.

[63] L'Ecuyer, P., Rubino, G., Saggadi, S., and Tuffin, B. Approximate zero-variance importance sampling for static network reliability estimation. *IEEE Transactions on Reliability 60*, 3 (2011), 590–604.

[64] Leitner, M. Integer programming models and branch-and-cut approaches to generalized 0,1,2-survivable network design problems. *Computational Optimization and Applications 65*, 1 (2016), 73.

[65] Leitner, M., and Raidl, G. R. Variable neighborhood and greedy randomized adaptive search for capacitated connected facility location. In *Computer Aided Systems Theory – EUROCAST 2011* (Berlin, Heidelberg, 2012), R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds., Springer Berlin Heidelberg, pp. 295–302.

[66] Lena, D., Robledo, F., and Romero, P. A hostile model for network reliability analysis. *J. Heuristics 8*, 2 (2017), 489–498.

[67] Ljubić, I., Mutzel, P., and Zey, B. Stochastic survivable network design problems. *Electronic Notes in Discrete Mathematics 41* (2013), 245 – 252.

[68] López-Sánchez, A., Sánchez-Oro, J., and Hernández-Díaz, A. Grasp and vns for solving the p-next center problem. *Computers and Operations Research 104* (2019), 295 – 303.

[69] Lourenco, H. R., Martin, O. C., and Stutzle, T. Iterated Local Search, 2001.

[70] Mahjoub, A., and Pesneau, P. On the steiner 2-edge connected subgraph polytope. *RAIRO Operations Research 42*, 1 (2008), 259–283.

[71] Martí, R., Campos, V., Resende, M. G., and Duarte, A. Multiobjective grasp with path relinking. *European Journal of Operational Research 240*, 1 (2015), 54 – 71.

[72] Martins, E., and Pascoal, M. A new implementation of yen's ranking loopless paths algorithm. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies 1*, 2 (2003), 121–133.

[73] Martins, E. Q. V., Pascoal, M. M. B., and Santos, J. L. E. Deviation algorithms for ranking shortest paths. *International Journal of Foundations of Computer Science 10*, 3 (1999), 247–261.

[74] Mauttone, A. Método RVR en la simulación de medidas de confiabilidad en redes, 2000.

[75] Mejía, M., and Aguirre, P. E. O. Network Topology Optimization using Tabu Search, 2005.

[76] Morales, E. Búsqueda Optimización y Aprendizaje, 2004.

[77] Nesmachnow, S. Evaluating simple metaheuristics for the generalized steiner problem. *Journal of Computer Science and Technology 5*, 4 (2005).

[78] Nesmachnow, S., and Pedemonte, M. Metaheurísticas basadas en adaptación social para el Problema de Steiner Generalizado. In *Actas VI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2009)* (New York, NY, USA, 2009), Association for Computing Machinery, p. 107–114.

[79] Nguyen, T., and Do, P. An ant colony optimization algorithm for solving group steiner problem. In *The 2013 RIVF International Conference on*

Computing Communication Technologies - Research, Innovation, and Vision for Future (RIVF) (2013), pp. 163–168.

[80] Orlov, V. I., Kazakovtsev, L. A., Rozhnov, I. P., Popov, N. A., and Fedosov, V. V. Variable neighborhood search algorithm for k-means clustering. In *Materials Science and Engineering Conference Series* (2018), vol. 450 of *Materials Science and Engineering Conference Series*, pp. 22–35.

[81] Panigrahi, D. Survivable network design problems in wireless networks. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms* (USA, 2011), Society for Industrial and Applied Mathematics, p. 1014–1027.

[82] Paolini, E. Minimal connections: the classical steiner problem and generalizations. *Bruno Pini Mathematical Analysis Seminar 3*, 1 (2012), 72.

[83] Pedemonte, M., and Cancela, H. A cellular ant colony optimisation for the generalised steiner problem. *International Journal of Innovative Computing and Applications 2* (2010), 188–201.

[84] Pessoa, L. S., Resende, M. G., and Ribeiro, C. C. A hybrid lagrangean heuristic with grasp and path-relinking for set k-covering. *Computers and Operations Research 40*, 12 (2013), 3132 – 3146.

[85] Pulsipher, J., and Zavala, V. Measuring and optimizing system reliability: a stochastic programming approach. *TOP* (2020), 1 – 20.

[86] Radojicic, N., Djenic, A., and Maric, M. Fuzzy grasp with path relinking for the risk-constrained cash-in-transit vehicle routing problem. *Applied Soft Computing 72* (2018), 486 – 497.

[87] Ravi, R., and Klein, P. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *Proceedings of the 3rd Symposium on Integer Programming and Combinatorial Optimization* (1993), pp. 39–55.

[88] Reinelt, G. TSPLIB–a traveling salesman problem library. *ORSA Journal on Computing 3*, 4 (1991), 376–384.

[89] Resende, M., and Ribeiro, C. *Optimization by GRASP*. Springer, 2016.

[90] Resende, M. G., and Velarde, J. L. G. Grasp: Greedy randomized adaptative search procedures. *Revista Iberoamericana de Inteligencia Artificial 7*, 19 (2003), 61–76.

[91] Resende, M. G. C., and Ribeiro, C. C. *GRASP: Greedy Randomized Adaptive Search Procedures*. Springer US, Boston, MA, 2014, pp. 287–312.

[92] Risso, C. *Using GRASP and GA to design resilient and cost-effective IP/MPLS networks*. PhD thesis, Universidad de la República, 2014.

[93] Robledo, F. Diseño Topológico de Redes. Casos de Estudio: *The Generalized Steiner Problem* and *The Steiner 2-Edge-Connected Subgraph Problem*. Master's thesis, Tesis de Maestría. Universidad de la República, 2000.

[94] Robledo, F. *GRASP heuristics for Wide Area Network design*. PhD thesis, Rennes University, Rennes, France, 2005.

[95] Robledo, F., and Canale, E. Designing backbone networks using the Generalized Steiner Problem. In *Proceedings of the 7th International Workshop on Design of Reliable Communication Networks* (2009), pp. 327–334.

[96] Rodríguez-Martín, I., Salazar-González, J.-J., and Yaman, H. A branch-and-cut algorithm for two-level survivable network design problems. *Computers and Operations Research 67* (2016), 102 – 112.

[97] Rodríguez-Martín, I., Salazar-González, J.-J., and Yaman, H. Hierarchical survivable network design problems. *Electronic Notes in Discrete Mathematics 52* (2016), 229 – 236. Proceedings of the 7th International Network Optimization Conference.

[98] Rodríguez-Pereira, J., Fernández, E., Laporte, G., Benavent, E., and Martínez-Sykora, A. The steiner traveling salesman problem and its extensions. *European Journal of Operational Research 278*, 2 (2019), 615 – 628.

[99] Rosenthal, A. Computing the reliability of complex networks. *SIAM Journal on Applied Mathematics 32*, 2 (1977), 384–393.

[100] Sadeghi, E., and Fan, N. On the survivable network design problem with mixed connectivity requirements. *Annals of Operations Research* (2019).

[101] Sartor, P. Problema General de Steiner en Grafos: Resultados y Algoritmos GRASP para la versión Arista-Disjunta. Master's thesis, Tesis de Maestría. Universidad de la República, 2011.

[102] Sartor, P., and Robledo, F. A heuristic for the edge-survivable general steiner problem. In *Control and Automation, and Energy System Engineering* (Berlin, Heidelberg, 2011), Springer Berlin Heidelberg, pp. 7–16.

[103] Sartor, P., and Robledo, F. Solving the generalized steiner problem in edge-survivable networks. In *Control and Automation, and Energy System Engineering* (Berlin, Heidelberg, 2011), Springer Berlin Heidelberg, pp. 7–16.

[104] Sartor, P., and Robledo, F. Grasp algorithms for the edge-survivable generalized steiner problem. *International Journal of Control and Automation 5* (2012), 27–44.

[105] Schauer, C., and Raidl, G. R. Variable neighborhood search and grasp for three-layer hierarchical ring network design. In *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature - Volume Part I* (Berlin, Heidelberg, 2012), Springer-Verlag, p. 458–467.

[106] Shen, H. Approximate algorithms for survivable network design. In *2012 Third International Conference on Networking and Computing* (2012), pp. 9–18.

[107] Suzuki, H., Ishihata, M., and Minato, S.-i. Designing survivable networks with zero-suppressed binary decision diagrams. In *WALCOM: Algorithms and Computation* (Cham, 2020), M. S. Rahman, K. Sadakane, and W.-K. Sung, Eds., Springer International Publishing, pp. 273–285.

[108] Tran, C. V., and Ha, N. H. A variable neighborhood search algorithm for solving the steiner minimal tree problem in sparse graphs. *EAI Endorsed Transactions on Context-aware Systems and Applications 5*, 15 (12 2018).

[109] Tripathi, P. A Deterministic Algorithm for the Vertex Connectivity Survivable Network Design Problem, 2010. Arxiv:1004.1208v1.

[110] Vijayaraghavan, V., Kianfar, K., Ding, Y., and Parsaei, H. A mixed integer programming based recursive variance reduction method for

reliability evaluation of linear sensor systems. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)* (2018), pp. 836–842.

[111] Williamson, D. P., Goemans, M. X., Mihail, M., and Vazirani, V. V. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica 15*, 3 (1995), 435–454.

[112] Winter, P. Steiner problem in networks: A survey. *Networks 17*, 2 (1987), 129–167.

[113] Woods, B. Generalized Steiner Problem in Halin Graphs. Master's thesis, University of Waterloo, 2010.

[114] Xiao, Y., and Konak, A. A variable neighborhood search for the network design problem with relays. *J. Heuristics 23*, 2-3 (2017), 137–164.

[115] Yuan, H., Lian, X., and Liu, J. Stochastic recursive variance reduction for efficient smooth non-convex compositional optimization. *CoRR abs/1912.13515* (2019).

Chapter 10

Appendix

10.1 Feasibility - Local Search Phase

In this section it is formally proved that the feasibility [1] is preserved during all the local searches from Chapter 6. Additionally, we describe the auxiliary procedures *General_RecConnect* and *FindSubstituteKeyPath*, that are used in the local searches *KeyTreeLocalSearch* and *SwapKeyPathLocalSearch* respectively.

Proposition 4. *KeyPathLocalSearch returns a feasible solution.*

Proof. Suppose that *KeyPathLocalSearch* does not preserve feasibility. Since the input is a feasible network, in some iteration we must have the following conditions:

- A feasible solution g_{sol}.
- The path $\hat{p}$ computed in Lines 7-8 meets the inequality $COST(\hat{p}) < COST(p)$, being $p \in K(g_{sol})$ the current key-path.
- The network $\hat{g} = \{g_{sol} \setminus p\} \cup \{\hat{p}\}$ is non-feasible: there exists $i, j \in T$ with less than $r_{i,j}$ node-disjoint paths.

This means that $INTERNAL\_NODES(\hat{p}) \cap NODES(g_{sol} \setminus p) \neq \emptyset$. This is a contradiction, since $NODES(\hat{p}) \subseteq NODES(p) \cup \{S_D - NODES(g_{sol})\}$. Therefore, $\hat{g}$ is feasible and satisfies the requirement matrix R, as we wanted to prove. □

[1] Here we consider the feasibility of the relaxation, this is, the GSPNC.

Procedure *General_RecConnect*

General_RecConnect is used during *KeyTreeLocalSearch*. Given a current solution g_{sol} and a key-node $v \in g_{sol}$, *General_RecConnect* tries to find a better key-tree T spanning the leaf-nodes belonging to T_v, where T_v is the tree associated to v. In order to preserve feasibility, T considers only Steiner nodes not included in g_{sol} and nodes belonging to T_v. Additionally, the links from the extremes of T_v are not considered.

Algorithm 12 $(g_{sol}, improve) = General\_RecConnect(G_B, C, g_{sol}, v, \overline{S})$

1: $cost \leftarrow cost\_Key\_Tree(v, g_{sol})$
2: $Y \leftarrow Nodes\_Key\_Tree(v, g_{sol})$
3: $Z \leftarrow Ends\_Key\_Tree(v, g_{sol})$
4: $\hat{S} \leftarrow Y \setminus Z \cup \overline{S}$
5: $U \leftarrow \{(i,j) \in G_B : i \in Z, j \in \hat{S}\}$
6: $\hat{\mu} \leftarrow < NODES(\hat{S} \cap G_B) >$; $\hat{\mu} \leftarrow \hat{\mu} \cup U$
7: $T \leftarrow v$
8: **while** $\exists u \in Z : u \notin T$ **do**
9: $\quad X = \{u \in Z, u \notin T\}$; $u \leftarrow Select\_Random(X)$
10: $\quad \mu \leftarrow \hat{\mu} \setminus \{Z - u\}$
11: $\quad p \leftarrow Dijkstra(u, T, \mu)$
12: $\quad T \leftarrow T \cup \{p\}$
13: **end while**
14: $T \leftarrow RemoveDegree1(\overline{S}, T)$
15: **if** $cost(T) < cost$ **then**
16: $\quad g_{sol} \leftarrow \{g_{sol} \setminus (Y \setminus Z)\} \cup \{T\}$
17: $\quad improve \leftarrow TRUE$
18: **else**
19: $\quad improve \leftarrow FALSE$
20: **end if**
21: **return** $(g_{sol}, improve)$

Figure 10.1: Pseudocode for *General_RecConnect*.

Figure 10.1 presents *General_RecConnect*. It receives the graph G_B of feasible connections, the link-costs C and the current solution g_{sol}, the current key-node v and the set of Steiner nodes $\overline{S}$ not belonging to g_{sol}. Let T_v be the associated key-tree. Line 2 computes the set of nodes Y belonging to T_v. Line 3 finds the leaf-nodes $Z \subseteq Y$. The set $\hat{S} = \{Y \setminus Z\} \cup \{\overline{S}\}$ is found in Line 4. The set $\hat{S}$ is precisely the Steiner nodes that do not belong to g_{sol}

neither the extremes of T_v. In Line 5, the set U includes all the connections from G_B with an extreme in Z and the other in $\hat{S}$. Clearly, in U there are no links between nodes belonging to Z. Let $\hat{\mu}$ be the network induced by $\hat{S}$ in g_{sol}. The set U is added to $\hat{\mu}$ in Line 6. Note that any spanning tree computed in $\hat{\mu}$ is a potential replacement for T_v in g_{sol}, since the replacement preserves feasibility. Line 7 forces the root-node v to be included in the new tree, T. The *while-loop* of Lines 8-13 iteratively builds a new key-tree, by adding nodes from Z to T. In Line 9, a node $u \in Z$ not previously picked is uniformly chosen at random. The auxiliary network $\mu = \hat{\mu} \setminus \{Z - \{u\}\}$ is considered in Line 10 to find a path from u to T. The nodes from $Z - \{u\}$ are not considered, since these nodes must be the extremes of T. Line 11 finds the shortest path from u to T in μ. Let p be that path; then p is added to T. The *while-loop* of Lines 8-13 is finished precisely when all the nodes belonging to Z are added into T. The Steiner nodes from the extremes are removed from T in Line 14. We remark that those nodes are not necessary to fulfill feasibility. Furthermore, it is straight to see that a new key-tree can be constructed from $T_v \subseteq \hat{\mu}$. The costs of both T and T_v are compared in Line 15. If T is better, the replacement takes place in the solution g_{sol} in Line 16. The indicator variable *improve* is set to $TRUE$ in Line 17 (used in $KeyTreeLocalSearch$). Otherwise, *improve* is set to $FALSE$ in Line 19. Both the indicator variable *improve* and the resulting solution g_{sol} are returned in Line 21.

Proposition 5. *General_RecConnect preserves feasibility.*

Proof. Following the previous terminology, it is straight to note that T is a tree after the *while-loop* of Lines 8-13. The following observations are in order:

- $Z \subseteq T$
- Z is precisely the extreme-nodes belonging to T.
- $NODES(T) \cap NODES(g_{sol}) = Z \cup J$, with $J \subseteq Y \setminus Z$
- There exists a root-node $\hat{s}$ of T (not necessarily $\hat{s} = v$).

If the condition from Line 15 is true, the algorithm finds $\hat{g} = \{g_{sol} \setminus T_v\} \cup \{T\}$ in Line 16. The feasibility of $\hat{g}$ is induced by the previous observations. In fact, the loss of connectivity requirements when T_v is removed is reestablished with the addition of T. Therefore, *General_RecConnect* returns a feasible network in Line 19. □

Proposition 6. *SwapKeyPathLocalSearch preserves feasibility.*

Proof. If *SwapKeyPathLocalSearch* does not preserve feasibility, in a certain step the following conditions are met:

- g_{sol} is feasible.
- The path $\hat{p}$ computed in Lines 6-8 satisfy: $COST(\overline{p} \setminus \{g_{sol} \setminus p\}) < COST(p)$, where $p \in K(g_{sol})$ is the current key-path, so Line 10 takes effect.
- The network $\hat{g} = \{g_{sol} \setminus p\} \cup \hat{p}$ is non-feasible, and there are less than $r_{i,j}$ node-disjoint paths between $i, j \in S_D^{(i)}$ in $\hat{g}$.

Let $\hat{p}_{i,j} \in P_{i,j}$ such that $p_{i,j} \subseteq \hat{p}_{i,j}$. Consider the path $p_{aux} = \{\hat{p}_{i,j} \setminus p\} \cup \{\overline{p}\}$. The nodes from $X_p(P)$ are excluded from $\hat{H}$, hence there are no nodes from $P_{i,j} \setminus \hat{p}_{i,j}$ belonging to H, and $INTERNAL\_NODES(p_{aux}) \cap INTERNAL\_NODES(p_{i,j}) \neq \emptyset$, for all $p_{i,j} \in P_{i,j} \setminus \hat{p}_{i,j}$, which contradicts that $\hat{g}_{sol}$ is non-feasible. Therefore, the network $\hat{G}$ computed in Line 10 is feasible. To complete the proof it is worth to note that:

- All the paths $\hat{p} \in P$ that include p are updated in Line 10, replacing p by $\overline{p}$. There are $r_{i,j}$ node-disjoint paths in $P_{i,j}$ connecting i to j, for all $i, j \in S_D^{(i)}$.
- The decomposition of g_{sol} into key-paths is repeated in Lines 13-15, where g_{sol} was replaced by $\hat{g}$ in Line 10. Therefore, the feasibility is preserved during each iteration of the local search.

□

Proposition 7. *KeyTreeLocalSearch preserves feasibility.*

Proof. Immediate from the feasibility of *General_RecConnect*, which is proved in Proposition 5. □

FindSubstituteKeyPath

This algorithm is called during *SwapKeyPathLocalSearch*. It receives the solution g_{sol}, the key-path p and the matrix P of paths between terminal nodes computed by the Construction phase, called *Greedy*.

Figure 10.2 presents *FindSubstituteKeyPath*. It deletes the key-path p given by the parameter of the solution g_{sol} and, using the information provided by P, it will try to reconstruct a feasible solution. After a feasible solution

is met, it returns $TRUE$ if the new solution is cheaper than g_{sol}, or $FALSE$ otherwise. The boolean *improve* and the resulting solution g_{sol} are returned.

Algorithm 13 $(g_{sol}, improve) = FindSubstituteKeyPath(g_{sol}, p, P, improve)$

```
1: cost ← cost(g_sol)
2: Disable(g_sol, p)
3: EliminatePaths(P, g_sol)
4: (g_sol, feasible) ← FindCheapestSolution(g_sol, P, cost, MAX_ATTEMPTS)

5: if feasible and cost(g_sol) < cost then
6:    g_sol ← {g_sol \ (Y \ Z)} ∪ {T}
7:    improve ← TRUE
8: else
9:    improve ← FALSE
10: end if
11: return (g_sol, improve)
```

Figure 10.2: Pseudocode for $FindSubstituteKeyPath$.

The cost of the input network is obtained in Line 1. The key-path p is deleted in Line 2, and all the paths from P that intersect with p are also removed in Line 3. A feasible solution is search in Line 4. An internal cycle is executed in this step, while a feasible solution is not found or a maximum number of iterations $MAX\_ATTEMPTS$ is met. If the resulting network is both feasible and cheaper (Line 5), the variable *improve* is set to $TRUE$ (Line 6). Otherwise, it is set to $FALSE$ (Line 9). The boolean variable *improve* and the resulting solution g_{sol} are returned in Line 11.

10.2 Feasibility - Construction Phase

In this section, the feasibility of the Greedy Randomized Construction is proved. Additionally, the methods $General\_Update\_Matrix$ and KSP are explained.

Algorithm 14 $(P, M) = General\_Update\_Matrix(G_{sol}, P, M, p, i, j)$

```
1: for all k ∈ S_D^(I), k ≠ i, j : k ∈ p do
2:   if m_{i,k} > 0 then
3:     if Nodes(P_{i,k}) ∩ Nodes(p_{(i,k)}) = {i, k} then
4:       P_{i,k} ← P_{i,k} ∪ {p_{(i,k)}}
5:       m_{i,k} ← m_{i,k} − 1; m_{k,i} ← m_{k,i} − 1
6:     end if
7:   end if
8:   if m_{k,j} > 0 then
9:     if Nodes(P_{k,j}) ∩ Nodes(p_{(k,j)}) = {k, j} then
10:      P_{k,j} ← P_{k,j} ∪ {p_{(k,j)}}
11:      m_{j,k} ← m_{j,k} − 1; m_{k,j} ← m_{k,j} − 1
12:    end if
13:  end if
14: end for
15: return (P, M)
```

Figure 10.3: Pseudocode for *General_Update_Matrix*.

Figure 10.3 presents *General_Update_Matrix*. It receives the solution obtained so far (during the construction), G_{sol}, the matrix with paths P, the connectivity matrix M, the terminals i, j and the path p found between them. The *for-loop* of Lines 1-14 studies each terminal node $k \in S_D^{(I)}$, $k \neq i, j$, such that $k \in p$. It determines whether there exists a sub-path between k and i (resp. j), or not, that must be node-disjoint with the previous paths $P_{i,k}$ (resp. $P_{k,j}$). If this is the case, the set $P_{i,k}$ (resp. $P_{k,j}$) is extended, adding $p_{(i,k)}$ (resp. $p_{(k,j)}$), and $m_{i,k}$ and $m_{k,i}$ (resp. $m_{k,j}$ and $m_{j,k}$) are decreased a unit. New paths between intermediate terminals for the path p are also included as an optimization to this algorithm.

Proposition 8. *At the end of General_Update_Matrix, the following clauses are met:*

- $P_{i,j} = \emptyset$ *if and only if* $m_{i,j} = r_{i,j}$.
- *If* $m_{i,j} = k$ *(with* $k \in \{0, \ldots, r_{i,j}\}$*), there exists at least* $r_{i,j} - k$ *node-disjoint paths from* i *to* j *in* G_{sol}.
- *The relation* $|P_{i,j}| = r_{i,j} - m_{i,j}$ *in each iteration of the construction algorithm.*

Proof. It is assumed that both P and M satisfy the statement when $General\_Update\_Matrix$ is called. Let us pick $i, j \in S_D^{(I)}$ as the pair of terminals and p the path that connects them computed by the construction algorithm. The cycle studies $\forall\, k \in S_D^{(I)}, k \in p, k \neq i, j$ the following cases:

- *Case I:* If $m_{i,k} > 0$ we know that there exists $r_{i,j} - m_{i,j}$ node-disjoint paths between i and j in G_{sol}. Further, if $m_{i,j} = r_{i,j}$ we have $P_{i,j} = \emptyset$. If the condition $NODES(P_{i,k}) \cap NODES(p_{(i,k)}) = \{i, k\}$ is true, the sub-path $p_{(i,k)}$ is added to $P_{i,k}$, since this is node-disjoint with the path belonging to $P_{i,k}$. The values $m_{i,k}$ and $m_{k,i}$ are decreased a unit, preserving the veracity of the clauses.
- *Case II:* $m_{i,k} = 0$, is analogous.

□

Proposition 9. *If $A_{i,j} < MAX\_ATTEMPT$, $\forall i, j \in S_D^{(I)}$, the construction algorithm returns a feasible solution.*

Proof. We assume that there exists a subnetwork $G_{sol} \subseteq G_B$ that is a feasible solution. In Line 1, the algorithm initializes:

- G_{sol} with a set of terminal-nodes $S_D^{(I)}$, and an empty set of links.
- The auxiliary matrix M with $m_{i,j} = r_{i,j}$, $\forall i, j \in S_D^{(I)}$.
- The matrix $P = \emptyset$.

Let us assume that, the conditional *if* from Line 2 holds in a certain iteration. In Line 3, we pick a pair of terminal nodes $i, j \in S_D^{(I)}$ such that $m_{i,j} > 0$ uniformly at random. The auxiliary network $\overline{G} = G_B \setminus P_{i,j}$ is computed in Line 4. If there exists a node-disjoint path from i to j in $\overline{G}$, this path is node-disjoint with respect to the paths belonging to $P_{i,j}$. The auxiliary costs $\overline{C}$ associated to $\overline{G}$ are found in Line 5, assigning zero-cost to the links that are included in G_{sol}. The block of Lines 5-12 look for a new path from i to j in G_{sol} considering $\overline{C}$, using the fact that the condition $|P_{i,j}| = r_{i,j} - m_{i,j}$ is met during $General\_Update\_Matrix$. Let us discuss two cases:

- $\neg\exists p \subseteq \overline{G}$ that connects i to j: in this case, Line 7 updates $P_{i,j}$ and $m_{i,j}$, since $P_{i,j}$ contains a path that is intersection of two or more node-disjoint paths that connect i and j. The construction proceeds in Line 2.

- $\exists p \subseteq \overline{G}$: a path p is selected from the list L_p in Line 9. Sine $p \nsubseteq G_{sol}$, the current solution G_{sol} is update in Line 9. In Line 10, $m_{i,j}$ is decreased a unit.

Based on the construction phase previously described, it is straight to see that, after the *for-loop* of Lines 1-14, if $m_{i,j} = 0$, $\forall i,j \in S_D^{(I)}$, the resulting solution G_{sol} satisfies the connectivity requirements R. □

We finally give details of KSP, Yen [72], used to implement the K shortest paths, without cycles between two fixed nodes s and t. This solution belongs to the class of Deviation Algorithms [73], which builds a pseudo-tree of paths without cycles.

Algorithm 15 $ksp = KSP(G, s, t, k)$

1: $ksp \leftarrow \emptyset$; $X \leftarrow \emptyset$, $k_{aux} \leftarrow 0$
2: $p \leftarrow Dijkstra(G, s, t)$
3: **if** $p \neq \emptyset$ **then**
4: $X \leftarrow X \cup \{p\}$; $d(p) \leftarrow s$
5: **while** $X \neq \emptyset$ **and** $k_{aux} < k$ **do**
6: $k \leftarrow k + 1$
7: $p_k \leftarrow GetMinCostPath(X)$, $v_i^k \leftarrow d(p_k)$
8: $X \leftarrow X - \{p_k\}$, $ksp \leftarrow ksp \cup \{p_k\}$
9: **if** $k_{aux} < k$ **then**
10: **while** $v_i^t \neq t$ **do**
11: $\overline{G} \leftarrow G - Nodes(subp_k(s, v_{i-1}^k))$
12: $\overline{G} \leftarrow \overline{G} - Arcs(v_i^k, v_{i+1}^k)$
13: $\overline{G} \leftarrow \overline{G} - Arcs(starting\ in\ v_i^k\ removed\ when\ p_k\ was\ computed)$
14: $\hat{p} \leftarrow Dijkstra(\overline{G}, v_i^k, t)$
15: **if** $\hat{p} \neq \emptyset$ **then**
16: $\hat{p} \leftarrow subp_k(s, v_{i-1}^k) + \hat{p}$, $X \leftarrow X \cup \{\hat{p}\}$
17: **end if**
18: $v_i^k \leftarrow v_{i+1}^k$
19: **end while**
20: **end if**
21: **end while**
22: **end if**
23: **return** ksp

Figure 10.4: Pseudocode for KSP.

Figure 10.4 presents KSP. It receives the graph G, two nodes s and t

and an integer $k \geq 1$ that represents the number of shortest paths without cycles between s and t. The collection of candidate paths starts as empty sets in Line 1. Line 2 applies Dijkstra algorithm [36] to find the shortest path between s and t. If there exists such path p, then p is added to the list (Line 4). In the *while-loop* of Lines 5-21, the k shortest paths are computed. During the iteration k, the last path p_k already found in X (Line 7), and its deviation node v_i^k, p_k is added into ksp and eliminated from X (Line 8). If the k shortest paths are not found yet, new deviation nodes from p_k are obtained during the *while-loop* of Lines 10-19. Only the nodes from v_i^k to t are analyzed in order to avoid repeated operations. For each node, the shortest path between v_i^k and t is found using Dijkstra (Line 14), and if $\hat{p}$ is not empty, a new path is obtained as the concatenation $\hat{p} \leftarrow subp_k(s, v_{i-1}^k) + \hat{p}$ (Line-16) and added to the list. In order to avoid cycle or previous paths, the nodes from p_k that are ancestors of v_i^k are deleted from G (Line 11), as well as the arcs (Lines 12-13). In this way, the k shortest paths without cycles between s and t are found. For each analyzed node, we must apply Dijkstra algorithm whose complexity is $O(m+nlog(n))$ [36]. In a worst case, we must analyze n nodes for every path $p_1, \ldots, p_k$, and the complexity for the KSP is $O(kn(m+nlog(n)))$. See [62] for further details.

10.3 Graphical Tools

Even though the implementation of graphical interfaces were not mandatory in this thesis, the author considers it is beneficial to have a graphical tool able to create graphs in a fast and simple manner, and to visualize them in order to facilitate the validation tests for the algorithms. For that purpose, a graphical tool (Graph Viewer) was developed. Graph Viewer allows to create graphs, and save them in XML files, for a latex loading for the same application and vice-versa. This tool was useful in order to see the resulting graphs during the validation tests, and to define input graphs as well. GraphViewer interface is illustrated in Figure 10.5

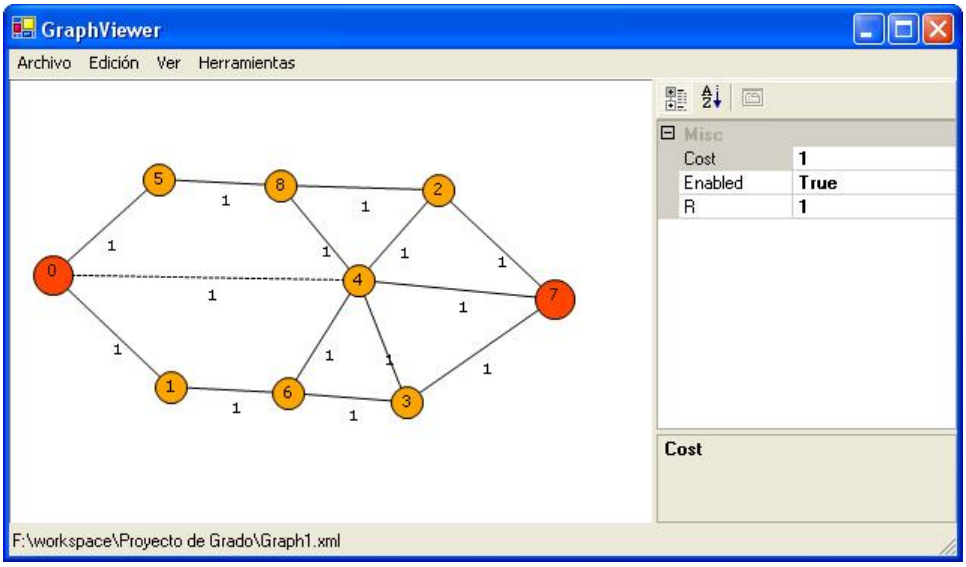

Figure 10.5: GraphViewer interface.

In order to save the results of the algorithms, we write a file in XML format, and then we choose XML, since this is standard for the information formatting. The format is the following. A root tag called *Graph* contains the information of all the graph. Inside the *Graph* tag, three tags are found: *Nodes*, *Edges* and *Connectivities*:

- *Nodes* is a sequence of *node* tags that represents each of the nodes.
- *Edges* is a sequence of *edge* tags that represents each of the edges. Each edge contains the information of the nodes that are linked.
- *Connectivities* is a sequence of *connectivity* tags that describe connectivities between each pair of terminal nodes.

```
<?xml version="1.0" standalone="yes"?>
<Graph xmlns="http://tempuri.org/Graph.xsd">
  <Nodes>
    <Node>
      <NodeId>0</NodeId>
      <IsTerminal>True</IsTerminal>
      <Enabled>True</Enabled>
      <R>1</R>
      <X>16</X>
      <Y>117</Y>
```

```
    </Node>
    <Node>
      <NodeId>1</NodeId>
      <IsTerminal>False</IsTerminal>
      <Enabled>True</Enabled>
      <R>1</R>
      <X>97</X>
      <Y>194</Y>
    </Node>
  </Nodes>
  <Edges>
    <Edge>
      <Node1Id>0</Node1Id>
      <Node2Id>5</Node2Id>
      <Enabled>True</Enabled>
      <Cost>1</Cost>
      <R>1</R>
    </Edge>
    <Edge>
      <Node1Id>5</Node1Id>
      <Node2Id>8</Node2Id>
      <Enabled>True</Enabled>
      <Cost>1</Cost>
      <R>1</R>
    </Edge>
  </Edges>
  <Connectivities>
    <Connectivity>
      <Node1Id>0</Node1Id>
      <Node2Id>7</Node2Id>
      <Value>2</Value>
    </Connectivity>
  </Connectivities>
</Graph>
```

Figure 10.6 shows the XML structure used to save the information.

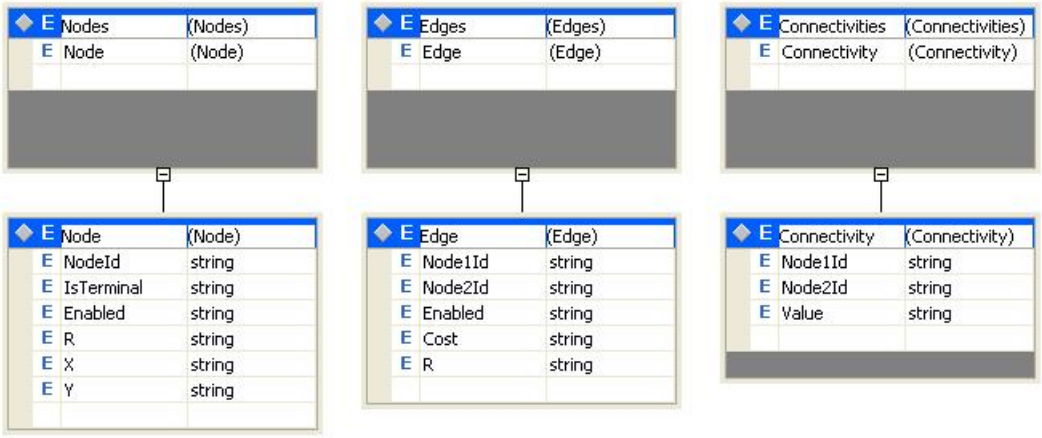

Figure 10.6: XML Structure.

In order to visualize the result of an execution, this is, the XML file generated by *NetworkDesign*, we use the application illustrated in Figure 10.7.

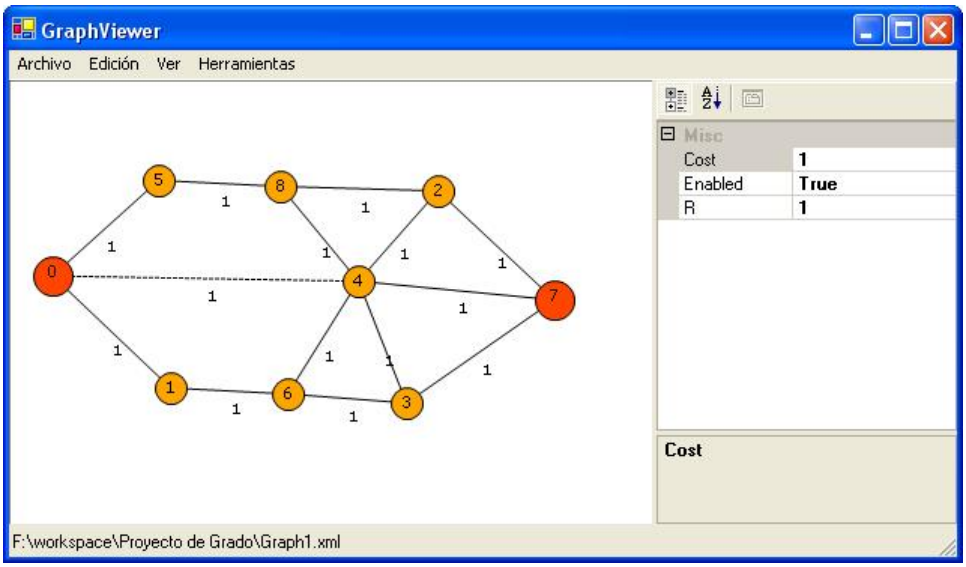

Figure 10.7: GraphViewer Application.

This application allows to create a new graph, modify it, and it greatly simplifies the creation of test-cases, standardization, and visualization. In order to modify the properties of some node or some edge, it suffices a double-click over it and edit the corresponding grid that is shown on the right. In order to add a new node we choose *Edit* and *New-node*, as shown in Figure 10.8.

Figure 10.8: Adding New Node (spanish dialogue).

Similarly, if we wish to add a new edge, we just do a click on the center of a node and move to the destination node (see Figure 10.9):

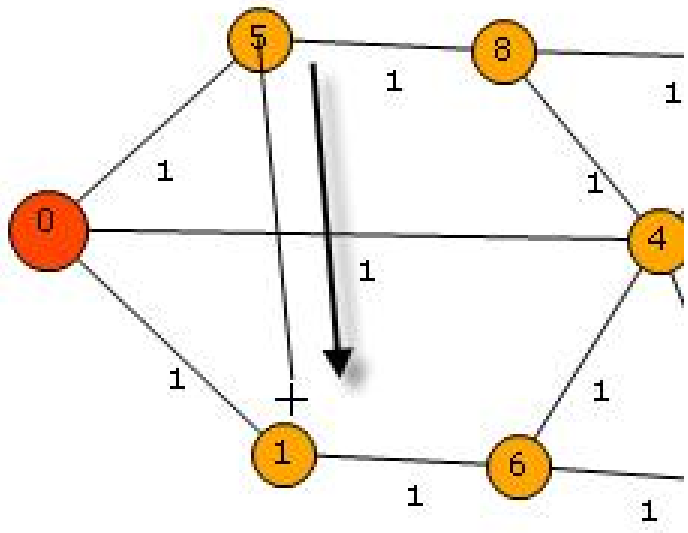

Figure 10.9: Intuitive way to create edges.

Other options include to access the connectivity matrix and edit its values. The numbers on the tags represents the numbers of the terminal-nodes, and the values from the text-boxes are the connectivity requirements (see Figure 10.10 for a graphical illustration).

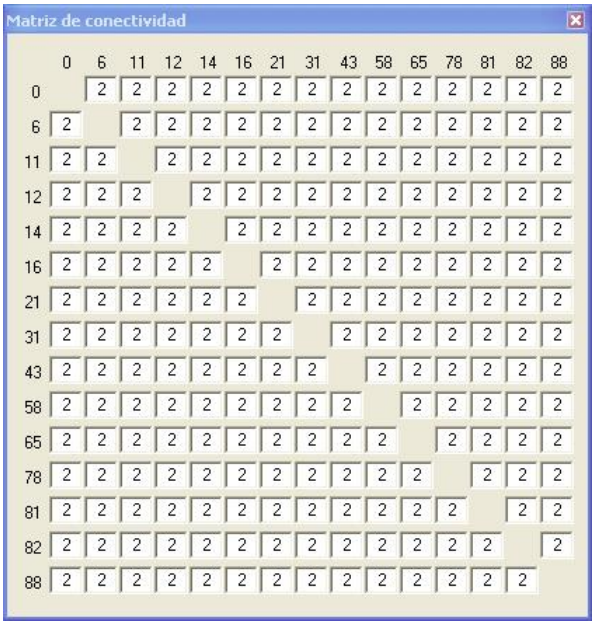

Figure 10.10: Connectivity Matrix.

Some auxiliary functionalities were included, such as Dijkstra algorithm between a pair of nodes [36], and the global cost of the graph.

10.4 Validation Tests

The goal of this section is to perform validation tests for the different algorithms that were implemented during this thesis. First, validation tests for Greedy algorithm and the three local searches of our VND proposal takes place. Then, we consider six small-sized networks in order to determine the correctness of our RVR implementation. The networks were selected in order to find the exact network reliability analytically, and thus compare the RVR estimation easily. It is worth to remark that here we not present the results of the algorithms (shown in Chapter 7), but only test validations are performed, using a graph generation algorithm. In this way, a random graph generation allow us to achieve random test cases, and these graphs do not depend on the algorithm to test. Recall that the results considered well-known test-cases taken from TSPLIB [88].

10.4.1 Greedy

Several test-cases were applied for our Greedy construction [31, 94]. Given that this algorithm is not deterministic, we considered small graphs. Here, we illustrate one validation. All the tests for the mentioned algorithms as well as other implementations can be found in the class Test.h. Figures 10.11-10.15 illustrate the operation of Greedy algorithm [31, 94]. Its correctness under several test-cases was confirmed. In the example, the ground graph G_B has cost equal to 100 (see Figure 10.11).

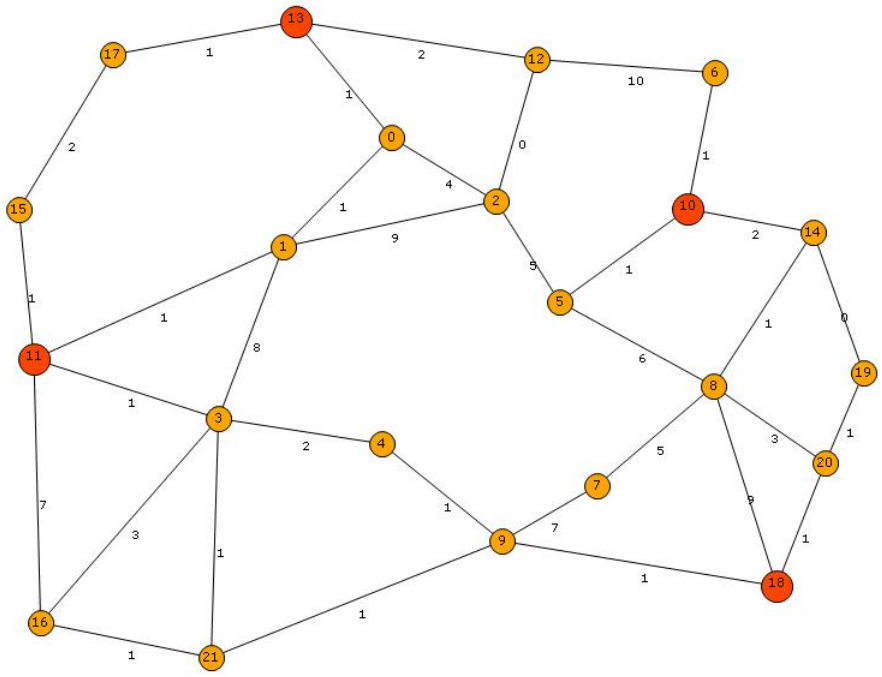

Figure 10.11: Validation of Greedy Construction: ground graph G_B.

At the beginning, the solution consists of terminal nodes only, finding node-disjoint paths between terminal nodes iteratively; see Figures 10.12-10.14. A low-cost feasible solution is met after Greedy execution. Its cost is equal to 30. The reader can appreciate that at least 2 node-disjoint paths can be found for every pair of terminal nodes (Figure 10.15).

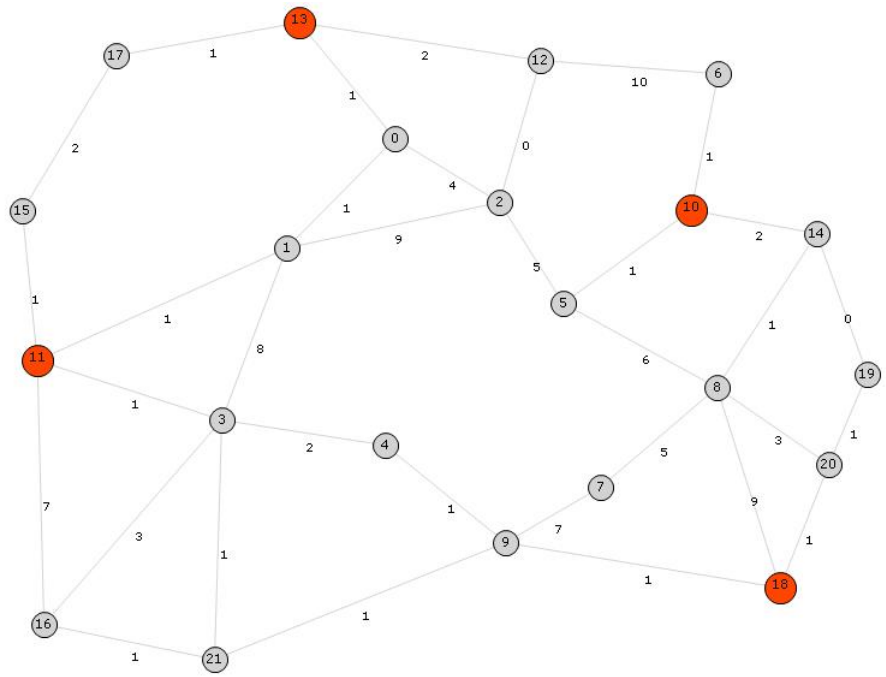

Figure 10.12: Validation of Greedy Construction.

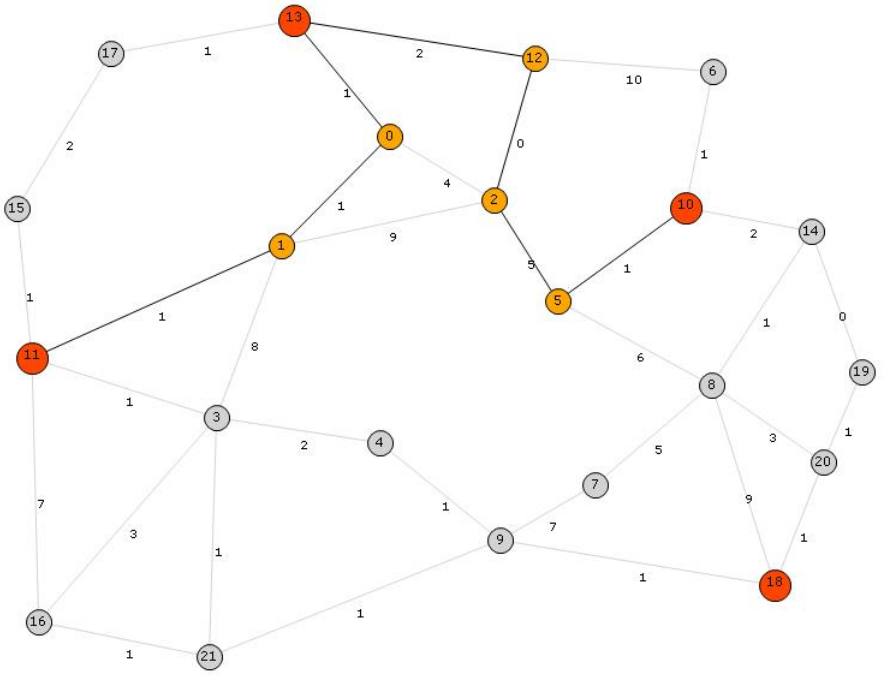

Figure 10.13: Validation of Greedy Construction.

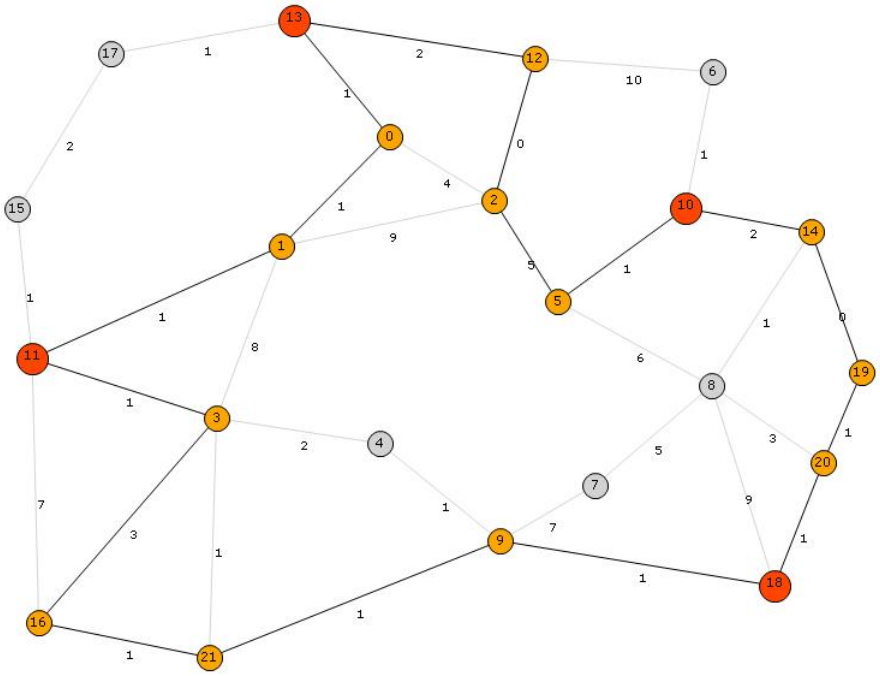

Figure 10.14: Validation of Greedy Construction.

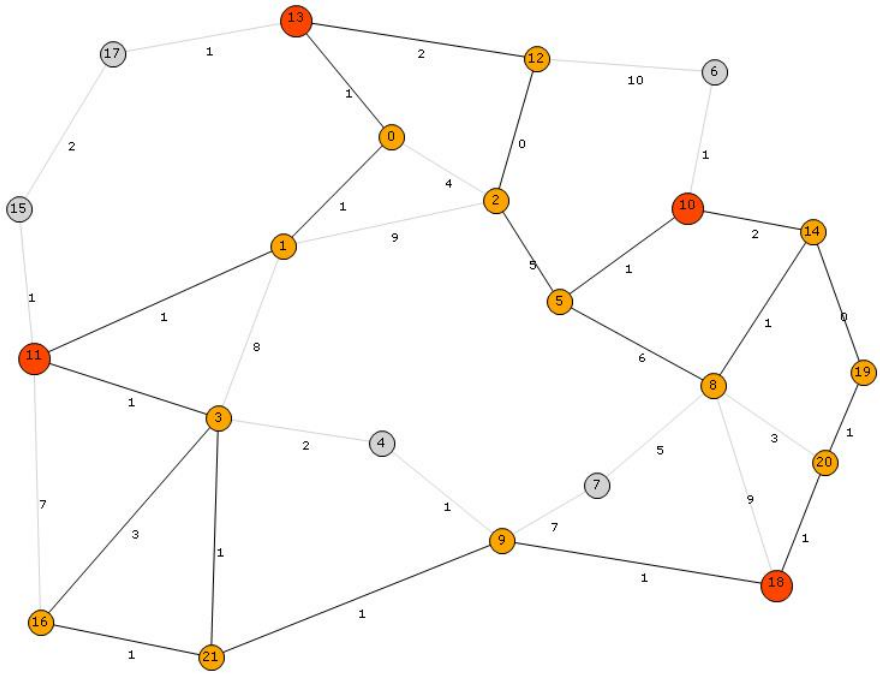

Figure 10.15: Validation of Greedy Construction.

10.4.2 Local Search

The VND phase considers three local searches. These algorithms include randomization, and several validations using small-sized networks were considered as in Greedy algorithm. Here we illustrate some validations (the tests for the mentioned algorithms can be found in the class Test.h.)

KeyTreeLocalSearch

Figures 10.16-10.20 illustrate the operation of *KeyTreeLocalSearch*. The ground graph G_B from Figure 10.16 has initial cost of 54. Figure 10.17 shows that the first selected node is 1. The key-tree is marked with red lines. Figure 10.18 shows another key-tree that replaces the first one. After this replacement, the cost of the resulting graph is 51. The second node selected by the algorithm is node 5. The corresponding key-tree is marked in Figure 10.19. A contraction of node 5 takes place, since all the branches of the key-tree go directly to node 7; therefore, node 5 is eliminated. The result is illustrated in Figure 10.20. The successive steps cannot find new improvements, and the final cost after this local search is 38.

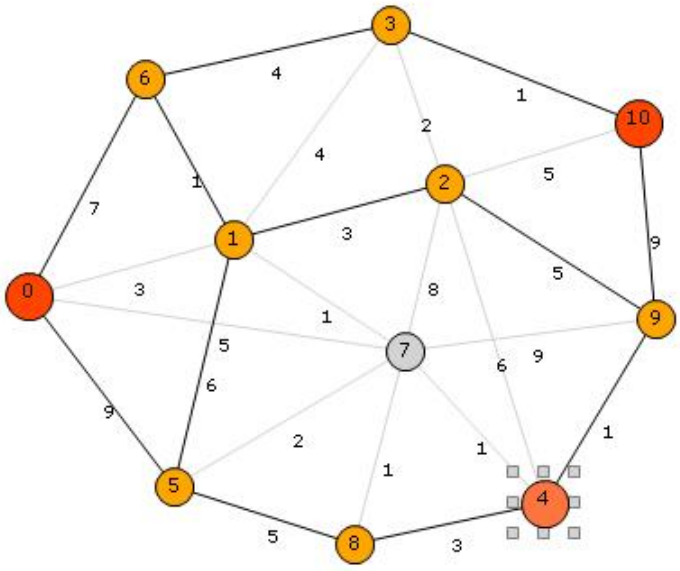

Figure 10.16: Validation of *KeyTreeLocalSearch*: ground graph G_B.

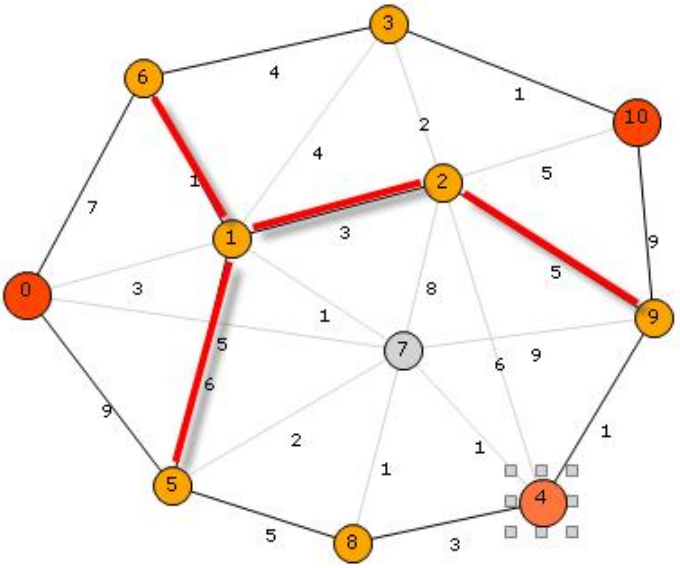

Figure 10.17: Validation of *KeyTreeLocalSearch*: Key-Tree

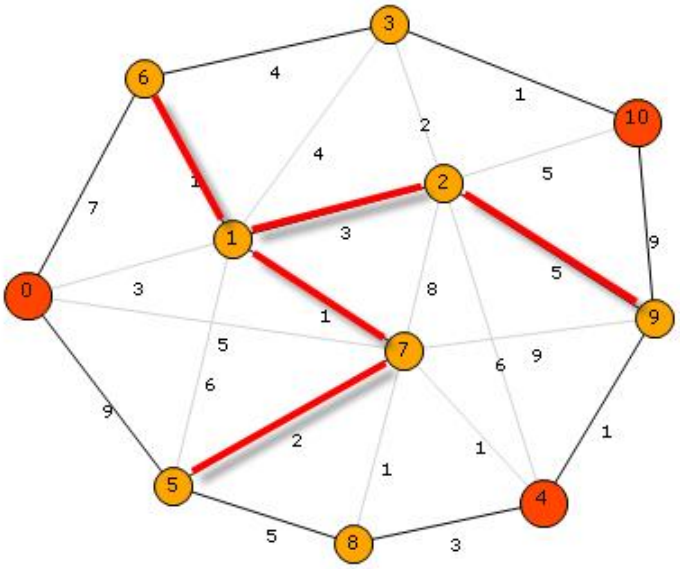

Figure 10.18: Validation of *KeyTreeLocalSearch*: Replacement

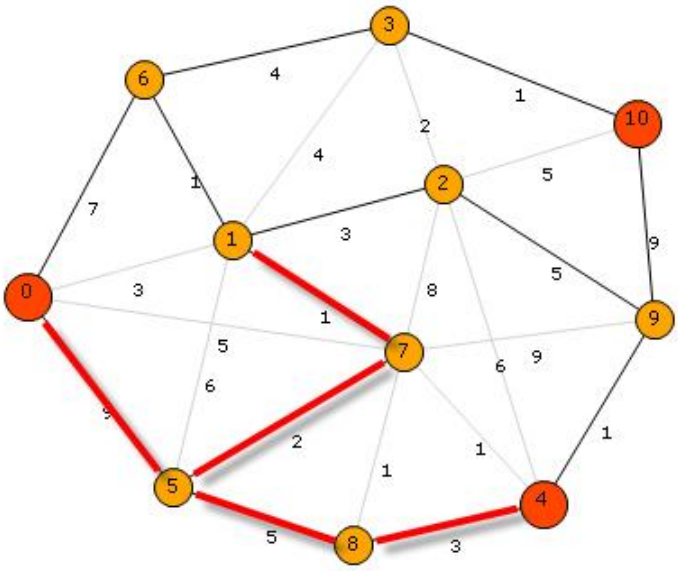

Figure 10.19: Validation of *KeyTreeLocalSearch*: Key-Tree rooted at node 5.

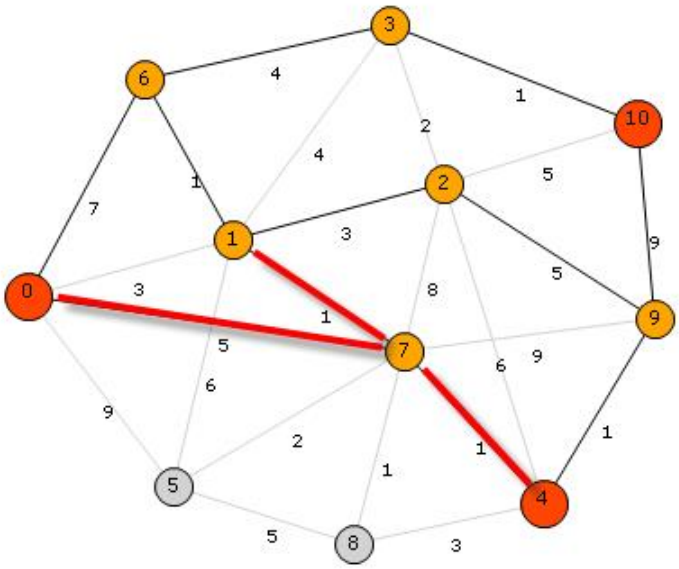

Figure 10.20: Validation of *KeyTreeLocalSearch*: result after the elimination of node 5.

KeyPathLocalSearch

We apply *KeyPathLocalSearch* over the same ground graph G_B. The cost of the initial graph is 54. A single key-path replacement takes effect, as it can be appreciated from Figure 10.22. The cost of the resulting graph is 49.

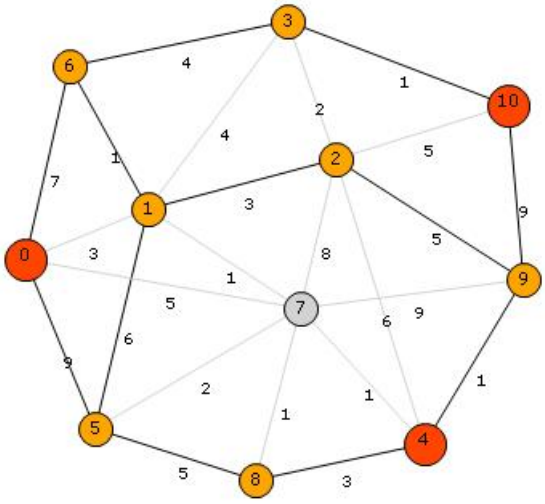

Figure 10.21: Validation of *KeyPathLocalSearch*: ground graph G_B.

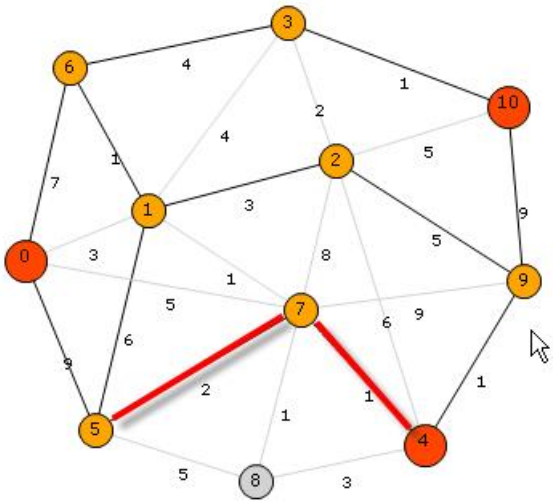

Figure 10.22: Validation of *KeyPathLocalSearch*: after the replacement of the key-path.

SwapKeyPathLocalSearch

In order to carry out the validation test for *SwapKeyPathLocalSearch*, recall that this local search considers the connectivity matrix, since the remaining local searches do not delete redundancies. In this local search, some key-paths could be discarded, if the resulting solution is feasible. As we can appreciate from Figures 10.23 and 10.24, the initial cost is 54 as well as in the previous validation tests, and the connectivity requirements are 2 in all the cases. From Figure 10.25 we can appreciate the key-path $0-5$, that is replaced by the link $\{0,1\}$. The cost of the resulting graph after the replacement is 48. Figure 10.27 shows that the key-path $6-1$ is eliminated, resulting the graph from Figure 10.28, whose cost is 47. Then, the algorithm eliminates the key-path $9-10$ and adds the link $\{2,10\}$ (see Figures 10.29 and 10.30), and the partial cost is reduced to 43. The key-path $1-2$ from Figure 10.31 is eliminated, decreasing the cost to 40. Finally, the key-path $0-1-5-8-4$ is replaced by the path $0-7-4$, as illustrated in Figures 10.33 and 10.34. The cost of the final graph is 29. We can appreciate that the global cost after this local search is much lower than in the remaining local searches, since the redundancies in the connectivity requirements are eliminated, preserving feasibility.

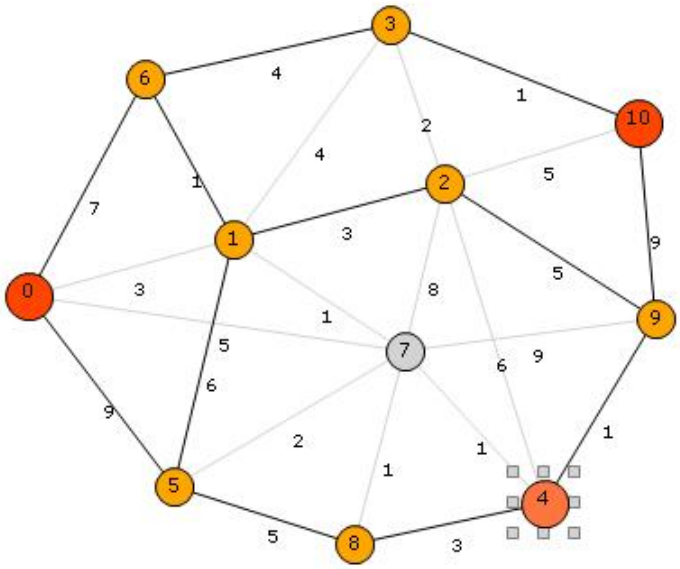

Figure 10.23: Validation of *SwapKeyPathLocalSearch*: ground graph G_B.

Figure 10.24: Validation of *SwapKeyPathLocalSearch*: connectivity matrix R.

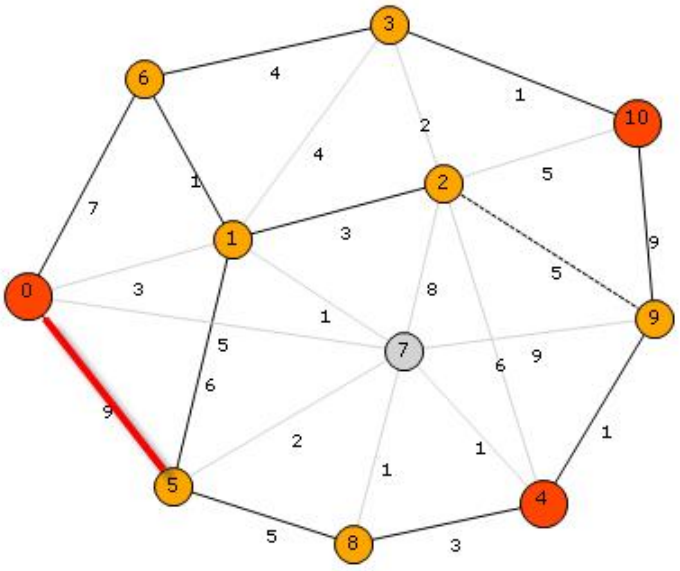

Figure 10.25: Validation of *SwapKeyPathLocalSearch*: key-path to replace.

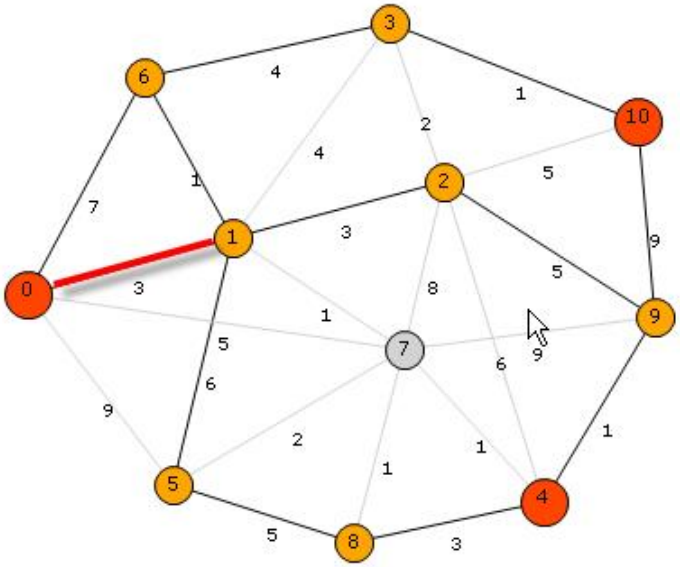

Figure 10.26: Validation of *SwapKeyPathLocalSearch*: link replacement.

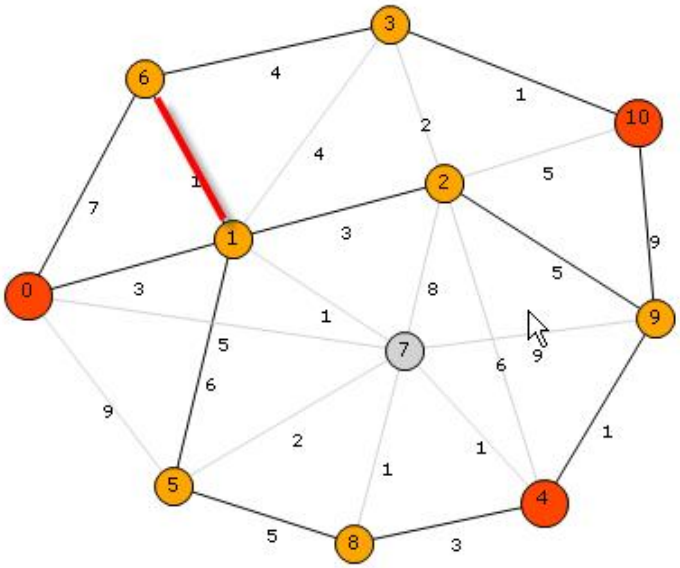

Figure 10.27: Validation of *SwapKeyPathLocalSearch*: resulting key-path.

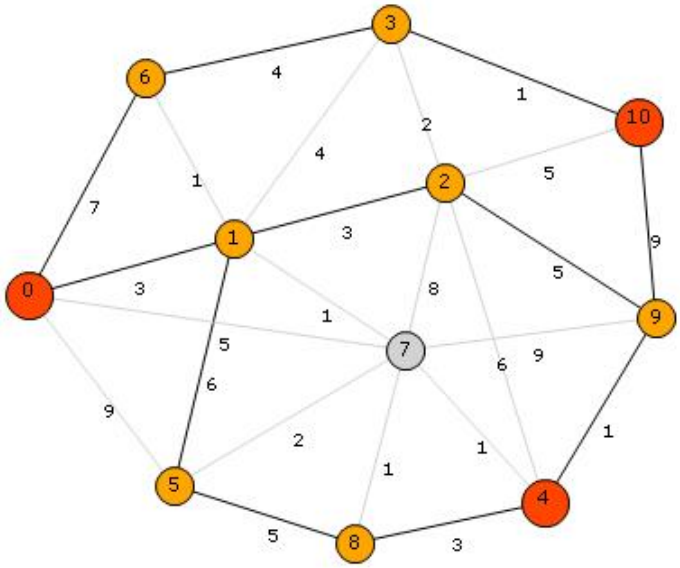

Figure 10.28: Validation of *SwapKeyPathLocalSearch*: resulting graph.

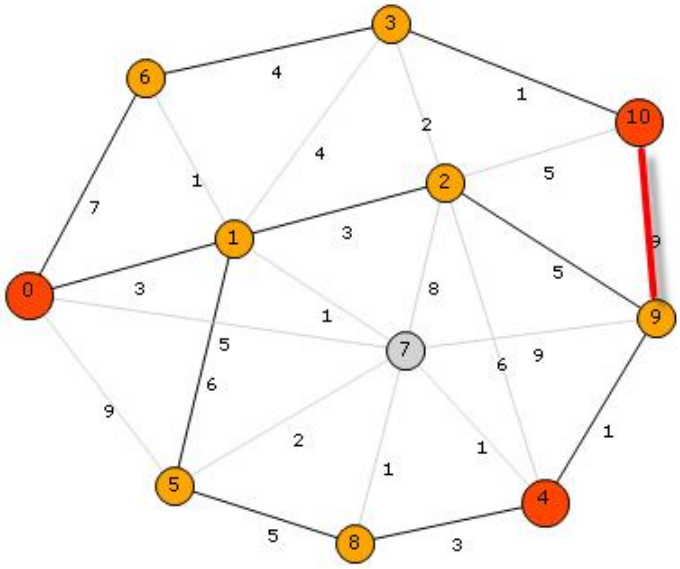

Figure 10.29: Validation of *SwapKeyPathLocalSearch*: key-path to replace.

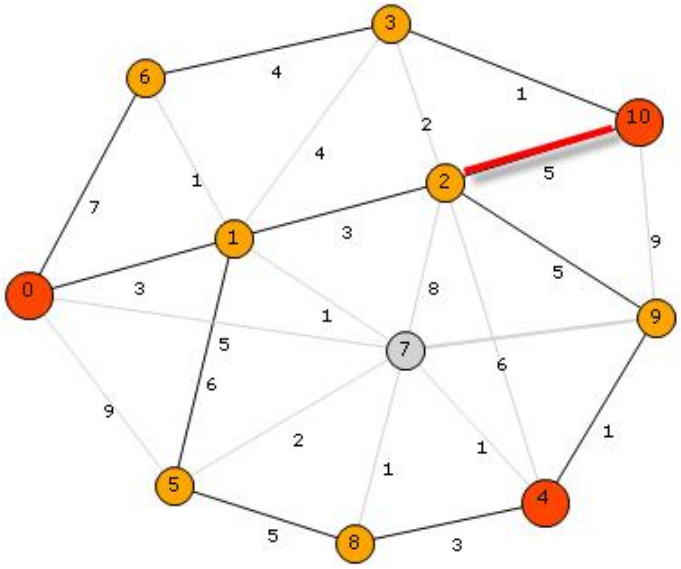

Figure 10.30: Validation of *SwapKeyPathLocalSearch*: link replacement.

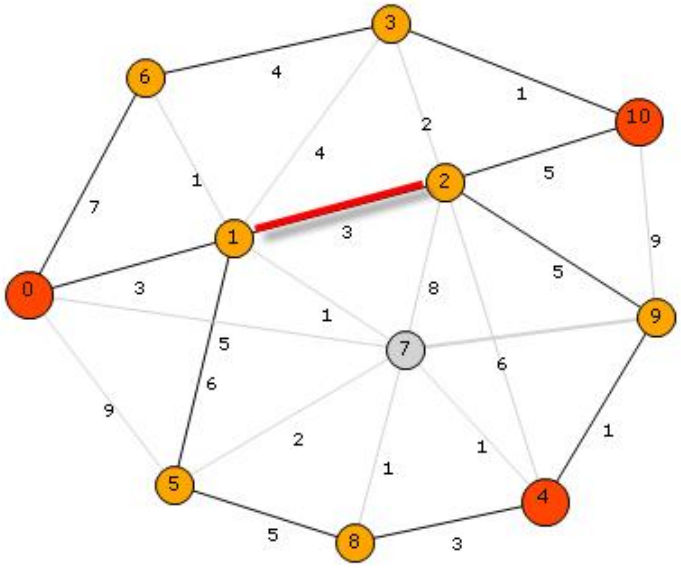

Figure 10.31: Validation of *SwapKeyPathLocalSearch*: redundant key-path.

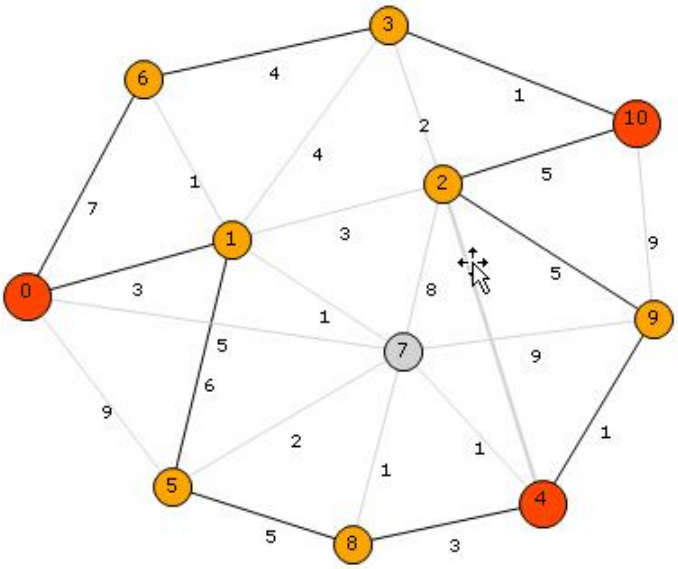

Figure 10.32: Validation of *SwapKeyPathLocalSearch*: resulting graph.

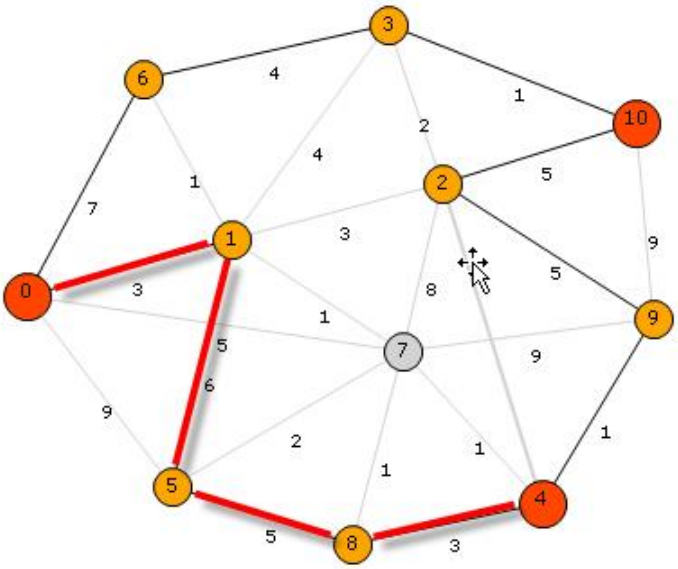

Figure 10.33: Validation of *SwapKeyPathLocalSearch*: key-path to replace.

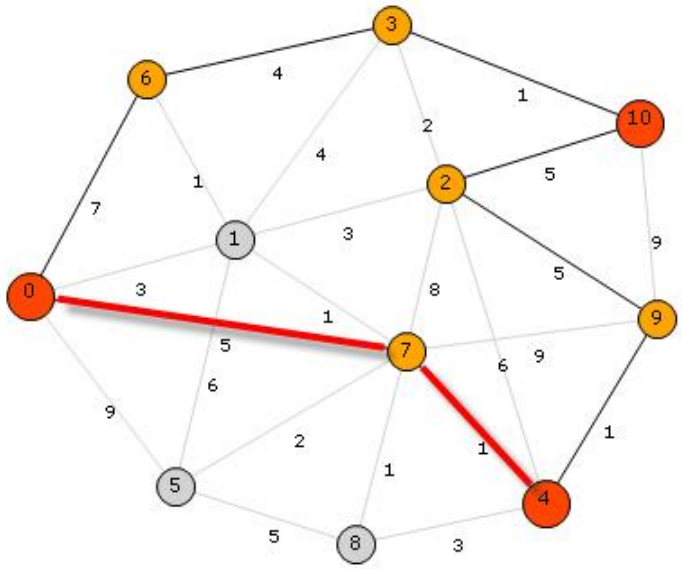

Figure 10.34: Validation of *SwapKeyPathLocalSearch*: link replacement.

10.4.3 *RVR*

Here we provide details of our validation tests for Recursive Variance Reduction (RVR). The pre-condition is that the implementation does not present run-time errors, but the potential existence of logical errors were not discarded, which are avoided during the validation process shown in the following paragraphs. Six small-sized topologies are presented with different elementary reliabilities, in which an exact reliability evaluation is analytically possible. Finally, we compare the exact reliability values with the resulting RVR estimation, expecting small gaps. The validation tests are based on the section under the same title presented in the work [74]. The following target network reliability measures are considered:

- $R_V : G \to [0, 1]$, or *all-terminal reliability measure*. This is the probability that all the nodes are mutually reachable in the resulting graph.
- $R_{s,t} : G \times V \times V \to [0, 1]$, or *source-terminal reliability measure*. This is the probability that s and t are connected by some path in the resulting graph.
- $R_K : G \times K \to [0, 1]$, or *K-terminal reliability measure*. This is the probability that all the terminal-set K belongs to the same connected component in the resulting graph.

The six validation tests will be presented in order. We will denote $R_{s,t}$, R_K or R_V to the exact reliability evaluation according to the scenario under study, $\hat{R}$ to the reliability estimation using RVR and $\hat{V}$ to its estimated variance. The number of independent samples in RVR is selected as $N = 10^4$. Terminal nodes are represented using black circles, while Steiner nodes are represented using white circles.

Case I: Single Link

Consider first just a direct link between two-terminal nodes. Figure 10.35 illustrates the network, together with the elementary reliability. The source-terminal reliability is $R_{s,t}(G) = pp'p''$. The results from Table 10.1 shows a perfect estimation of RVR with a null gap, and practically null variance.

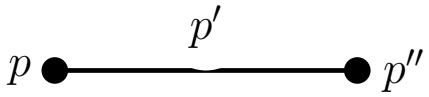

Figure 10.35: Single link with simultaneous node/link failures.

Table 10.1: RVR Validation Test: single link

Instance	$R_{s,t}$	$\hat{R}$	$\hat{V}$
$p = p' = p'' = 0.9$	0.729	0.729	$3.13631E-17$

Case II: Two-Path

The second test consists of an elementary path composed by two links, with elementary reliability p'. The end-points have reliability p, while the central point has reliability p'' (see Figure 10.36). The source-terminal reliability between the end-points is $R_{s,t} = p^2(p')^2p''$. Table 10.2 shows a small gap between the correct value $R_{s,t}$ and its estimation $\hat{R}$. The estimated variance is reduced as well.

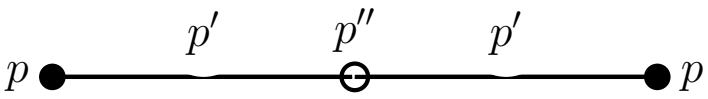

Figure 10.36: Elementary path with simultaneous node/link failures.

Table 10.2: RVR Validation Test: elementary path

Instance	$R_{s,t}$	$\hat{R}$	$\hat{V}$
$p = p' = p'' = 0.9$	0.59049	0.593354	$3.3341E-06$

In the following cases we consider perfect terminals only, with possible failures of Steiner nodes.

Case III: Triangle

Consider the complete graph composed by three terminal-nodes, or the triangle K_3, with identical elementary link-reliabilities p (see Figure 10.37). A complete graph with high elementary reliabilities for both links and nodes is therefore highly-reliable. Observe that if two or more links fail, the system is down. Otherwise, the system works. By elementary combinatorics, the all-terminal reliability is in this case $R_V = \binom{3}{0}p^3 + \binom{3}{1}p^2(1-p) = p^3 + 3p^2(1-p)$, where the first term means that *no link fails*, and the second means that *precisely one link fails*. From Table 10.3 we can observe that the gap between R_V and $\hat{R}$ is smaller than 10^{-3}, and the variance is extremely small, showing a good performance of RVR, as expected for small-sized networks.

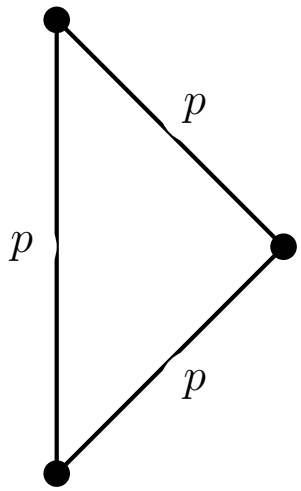

Figure 10.37: Triangle with link failures.

Table 10.3: RVR Validation Test: Triangle

Instance	R_V	$\hat{R}$	$\hat{V}$
$p = 0.9$	0.972	0.972473	$6.21454E-08$

Case IV: Triangle with a pending link

Consider a triangle with a pending link presented in Figure 10.38, where the central node is a Steiner node (and a cut-point, since it disconnect the network if fails). The K-terminal reliability depends strongly on the operational reliabilities of the central node and the bridge. A straight calculation leads to see that $R_K = p'p''[p^3 + 3p^2(1-p)]$. From Table 10.4, we can appreciate that the gaps in the reliability is again smaller than 10^{-3}, and the variance of the estimator is really small.

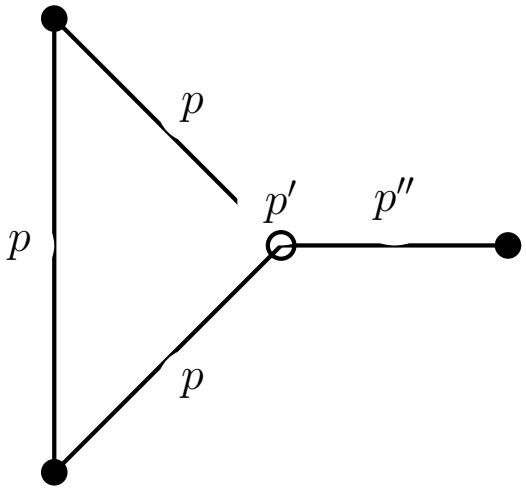

Figure 10.38: Triangle with pending link. Potential failure in central node.

Table 10.4: RVR Validation Test: Triangle with a pending link

Instance	R_K	$\hat{R}$	$\hat{V}$
$p = 0.9$, $p' = 0.5$, $p'' = 0.9$	0.4374	0.438066	$3.18052E-07$
$p = 0.9$, $p' = 0.9$, $p'' = 0.5$	0.4374	0.437479	$4.32176E-07$

Case V: Tree

Consider the tree from Figure 10.39, where the terminals are the leaf-nodes, and all the links and Steiner nodes operate independently, with identical reliability. In a tree, the failure of only component disconnects some pair of leaf-nodes. Consequently, the reliability of trees with several components is reduced, and it is the product of all the elementary reliabilities of its components: $R_K = p^9$. From Table 10.5, we can appreciate that both the reliability gap and the variance are greater than in the previous cases.

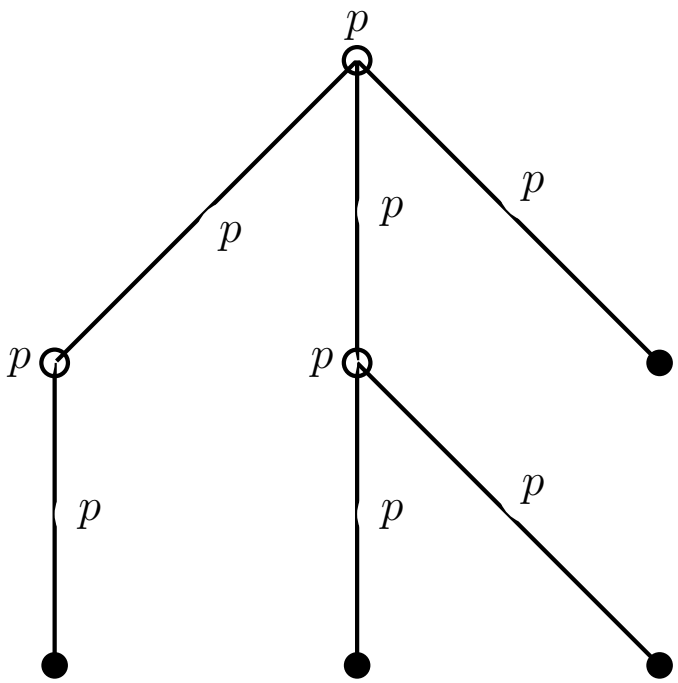

Figure 10.39: Tree with failures in links and non-leaf nodes.

Table 10.5: RVR Validation Test: Tree-graph

Instance	R_K	$\hat{R}$	$\hat{V}$
$p = 0.9$	0.387420489	0.391676	$1.53487e-05$

Case VI: Wheatstone Bridge

In the Wheatstone Bridge from Figure 10.40, our measure of interest is the source-terminal reliability $R_{s,t}$. It strongly depends on the elementary node-reliability p' of Steiner nodes, since these represent intermediate nodes. By an exhaustive enumeration of pathsets, a closed-form for the reliability can be obtained:

$$R_{s,t} = p^2p' + p^2(1-p)p' + p^3p'(1-p') + (1-p)p^3(p')^2 + p^3(1-p)^2(p')^2 + p^3(1-p)^2(p')^2.$$

From Table 10.6, we conclude that the reliability gaps and variance are acceptable.

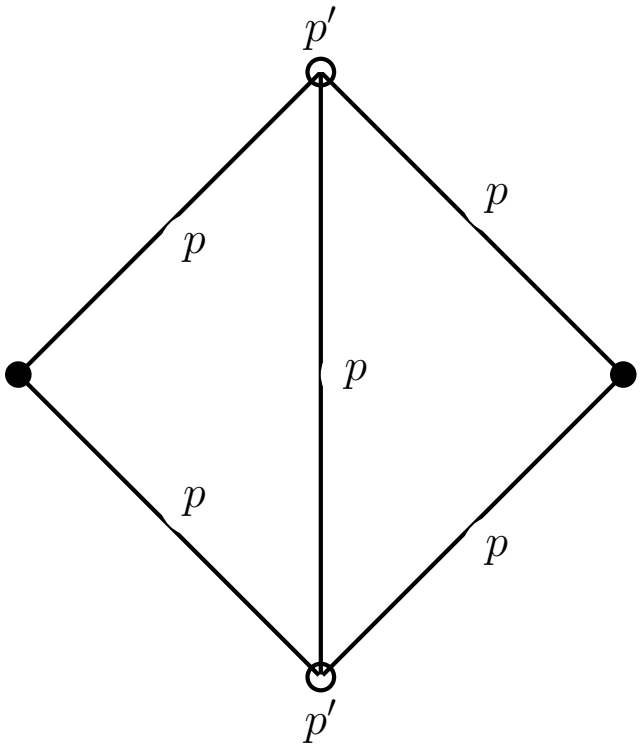

Figure 10.40: Wheatstone Bridge with failures in links and a couple of nodes.

Table 10.6: RVR Validation Test: Wheatstone Bridge

Instance	R_K	$\hat{R}$	$\hat{V}$
$p = 0.9$, $p' = 0.5$	0.64962	0.654653	$6.7791E-06$
$p = 0.9$, $p' = 0.9$	0.9383688	0.937484	$1.89235E-06$

Comments on the Results

Several validation tests were carried out using different elementary reliabilities for nodes and links. Some errors were detected during the validation tests, which served to correct the implementation of RVR with the corresponding logical modifications. Double-precision arithmetic was used for the operations, with up to 15 digits.

List of Figures