UNIVERSIDAD DE LA REPÚBLICA

FACULTAD DE INGENIERIA

INSTITUTO DE COMPUTACIÓN -INCO

PEDECIBA Informática

P H D T H E S I S

to obtain the title of

PhD on Informatics

of the Universidad de la República

Defended by

Gabriel Francisco BAYÁ MANTANI

Topological Design of Survivable Networks

Thesis Advisor: Franco ROBLEDO

Thesis Advisor: Antonio MAUTTONE

Defended on October 25, 2017

Jury:

Reviewers :	Dr. Guillermo DURÁN	-	Universidad de Buenos Aires (Argentina)
	Dr. Eduardo MORENO	-	Universidad Adolfo Ibañez (Chile)
President :	Dr. Ing. Eduardo FERNÁNDEZ	-	Universidad de la República (PEDECIBA INFORMÁTICA)
Examinators :	Dr. Víctor ALBORNOZ	-	Universidad Federico Santa María (Chile)
	Dr. Ing. Pablo RODRÍGUEZ BOCCA	-	Universidad de la República (PEDECIBA INFORMÁTICA)

Contents

Abstract

In the field of telecommunications there are several ways of establishing links between different physical places that must be connected according to the characteristics and the type of service they should provide. Two main considerations to be taken into account and which require the attention of the network planners are, in one hand the economic effort necessary to build the network, and in the other hand the resilience of the network to remain operative in the event of failure of any of its components. A third consideration, which is very important when quality of services required, such as video streaming or communications between real-time systems, is the diameter constrained reliability. In this thesis we study a set of problems that involve such considerations.

Firstly. we model a new combinatorial optimization problem called Capacitated m Two Node Survivable Star Problem (CmTNSSP). In such problem we optimize the costs of constructing a network composed of 2-node-connected components that converge in a central node and whose terminals can belong to these connected 2-node structures or be connected to them by simple edges. The CmTNSSP is a relaxation of the Capacitated Ring Star Problem (CmRSP), where the cycles of the latter can be replaced by arbitrary 2-node-connected graphs. According to previous studies, some of the structural properties of 2-node-connected graphs can be used to show a potential improvement in construction costs, over solutions that exclusively use cycles. Considering that the CmTNSSP belongs to the class of $\mathcal{NP}$-Hard computational problems, a GRASP-VND metaheuristic was proposed and implemented for its approximate resolution, and a comparison of results was made between both problems (CmRSP and CmTNSSP) for a series of instances. Some local searches are based on exact Integer Linear Programming formulations. The results obtained show that the proposed metaheuristic reaches satisfactory levels of accuracy, attaining the global optimum in several instances.

Next, we introduce the Capacitated m Ring Star Problem under Diameter Constrained Reliability (CmRSP-DCR) wherein DCR is considered as an additional restriction, limiting the number of hops between nodes of the CmRSP problem and establishing a minimum level of network reliability. This is especially useful in networks that should guarantee minimum delays and quality of service. The solutions found in this problem can be improved by applying some of the results obtained in the study of the CmTNSSP.

Finally, we introduce a variant of the CmTNSSP named Capacitated Two Node Survivable Tree Problem, motivated by another combinatorial optimization problem most recently treated in the literature, called Capacitated Ring Tree Problem (CRTP). In the CRTP, an additional restriction is added with respect to CmRSP, where the terminal nodes are of two different types and tree structures are also allowed. Each node in the CRTP may be connected exclusively in one cycle, or may be part of a cycle

or a tree indistinctly, depending on the type of node. In the variant we introduced, the cycles are replaced by 2-node-connected structures. This study proposes and implements a GRASP-VND metaheuristic with specific local searches for this type of structures and adapts some of the exact local searches used in the resolution CmTNSSP. A comparison of the results between the optimal solutions obtained for the CRTP and the CTNSTP is made. The results achieved show the robustness and efficiency of the metaheuristic.

Keywords— Network Optimization, Diameter Constrained Reliability, CmRSP, CmTNSSP, CRTP, CTNSTP, GRASP, VND.

List of publications issued from this thesis work

This thesis was written using a Swedish PhD style. The chapters are based on the following published papers:

1. "Capacitated *m* Two-Node Survivable Star Problem", Gabriel Bayá, Antonio Mauttone, Franco Robledo, and Pablo Romero. In Proccedings of the 7th International Network Optimization Conference (INOC 2015), Warzaw, Poland. Electronic Notes in Discrete Mathematics, Elsevier, Vol 52, Pages 253-260.

2. "The Capacitated m Two-Node Survivable Star Problem: A Hybrid Metaheuristic Approach", Gabriel Bayá, Antonio Mauttone, Franco Robledo, and Pablo Romero. In Proceedings of the 10th International Workshop on Hybrid Metaheuristics (HM 2016), June 8-10, 2016, Plymouth, UK, Lecture Notes in Computer Science, Springer, Vol 9668, pages 171-186,

3. "The Capacitated m Two-Node Survivable Star Problem", Gabriel Bayá, Antonio Mauttone and Franco Robledo. Yugoslav Journal of Operations Research (YUJOR), vol 27, number 2, year 2017.

4. "Capacitated m Ring Star Problem under Diameter Constrained Reliability", Gabriel Bayá, Antonio Mauttone, Franco Robledo, Pablo Romero and Gerardo Rubino. In Proceedings of 2nd International Workshop on Understanding the inter-play between Sustainabiliy, Resilience, and Robustness in networks (USSR 2014, collocated with RNDM 2014), Barcelona, Spain, Electronic Notes in Discrete Mathematics, Elsevier, vol 51, pages 23-30.

5. "Capacitated Two-Node Survivable Tree Problem", Gabriel Bayá, Antonio Mauttone, Franco Robledo, and Pablo Romero. Submitted to the 9th. International Workshop on Resilient Network Design and Modelling (RNDM 2017) Alghero, Sardinia, Italy. [1]

[1] A shorter version of this paper named "GRASP Heuristics for a Generalized Capacitated Ring Tree Problem" will be published in Proceedings of the Third International Conference on Machine Learning, Optimization and Big Data (Mod 2017), September 14-17, 2017, Volterra, Tuscany, Italy, Lecture Notes in Computer Science, Springer.

Agradecimientos

Al emprender un trabajo de este porte, muchas son las personas vinculadas de una u otra manera. Intentaré estar a la altura y expresar mi gratitud a todos aquellos que que lo hicieron posible.
A mi director de tesis, el Dr. Antonio Mauttone quien con su apoyo, sus acertadas sugerencias, sus comentarios detallados, experiencia y conocimiento dieron un gran realce a este trabajo.
A mi director académico y de tesis, el Dr. Franco Robledo, el alma mater de todo esto, quien me dio la confianza para empezar este desafío y terminarlo, que me brindó la oportunidad de mostrar algunos de los trabajos aquí publicados en las conferencias donde fueron presentados y de quien recibí todo el apoyo necesario para hacer de este trabajo una realidad. Vaya para él mi mas sentida expresión de gratitud.
Al Dr. Pablo Romero, co-autor en la mayoría de mis trabajos, quien con su aguda sutileza y conocimiento ha sido un tutor más de esta tesis.
Al Dr. Alessandro Hill, que me brindó los casos de prueba de su trabajo, imprescindibles para una buena medida de los resultados.
A los revisores y al resto de tribunal, quienes me honrarán con la lectura de este trabajo.
Y finalmente a mi esposa Elena y a mis hijos Claudia y Rodrigo, quienes llevan la parte mas ingrata en todo esto, soportarme en los momentos difíciles y darme su apoyo incondicional.

Acknowledgments

When undertaking a job of this size, there is a list of people who contributed in one way or another, so I would like to be thankful to all those that made it possible.
To my thesis advisor, Dr. Antonio Mauttone who, with his support, suggestions, detailed comments, experience, and knowledge, highlighted this work.
To my academic and thesis advisor, Dr. Franco Robledo, the true alma mater, who gave me the confidence to start this challenge and finish it, who gave me the opportunity to show some of the works published here at the conferences where they were presented, and from whom I received all the necessary support to make this work a reality. My heartfelt expression of gratitude.
To Dr. Pablo Romero, co-author in the majority of my works, who with his sharp subtlety and knowledge has been another advisor of this thesis.
To Dr. Alessandro Hill, who gave me the test cases of his work, essential for a good measure of the results.
To the reviewers and the rest of the jury, who would honor me with the reading of this work.
And finally to my wife Elena and my children Claudia and Rodrigo, who carry the most ungrateful part in all of this, endure me in difficult times and give me their unconditional support.

Chapter 1

Introduction

1.1 Topological Network Design

Before this introduction begins, we would like to define the concept of *network* in a simple way. Certainly there are a plenty of definitions for this term, but there is one of them that seems to apply in all cases: A network is a set of elements linked through some kind of communication. Veins, arteries, nerves, lymphatic system, electrical power, telephone, water, sewer, radio, television, transportation, distribution, surveillance, telecommunications, Internet, health, aid, social, and even terrorism. All of them are networks, all of its components are communicated. But what does comprise a good communication between them? On one hand, the communication channel should work, and on the other hand, even if the channel does not work, the components must be communicated in some other way. Hereupon, the goal of studying networks and their structures is clear, at least in an intuitive way.

In this thesis we work with networks, in particular telecommunication networks. Networks are represented by graphs. Components can be either nodes or links which connect nodes. Therefore communication of the network is a correlation between certain properties, such as connectivity, of the underlying graph used to represent it.

The main motivation for studying topological network design is its application in the area of telecommunications (Stoer, 1992). The study of the structure, the introduction of minimum levels of connectivity between their nodes, redundancy and resilience are main factors to avoid outages in case of a failure. Basically, the goal is to achieve structures with the desired level of redundancy and fault-tolerance in some of their nodes or links, and to allow savings in construction costs.

Initially, topological network design covered mainly availability aspects (e.g. public switched telephone network using simple connectivity). However, new applications over the Internet infrastructure show the weakness of the minimal way to connect nodes (i.e. tree-like structures). On the other hand, mesh-like structures present valuable connectivity properties, but their deployment is prohibitively expensive. A natural approach to an acceptable level of connectivity is to connect all terminals in a ring or a cycle, because in this topology there are two independent path between all pairs of nodes. The cheapest way to connect nodes in a ring, is known as Traveling Salesman Problem (Dantzig et al., 1954), and it is widely studied in the scientific literature. In the physical design of a telephony deployment, it is useful to consider several two-connected components joined to a perfect telephone exchange, but if some terminal nodes are far away from each other, it is better to connect them in more than one ring. A cost-effective "shape" of a solution is provided in (Baldacci et al., 2007), where given a *depot*, several terminal nodes, and optional nodes, in order to connect all terminals, the

authors propose to find the cheapest m rings joined in the depot, while some terminals can be pending on some node of a ring. The number of nodes within a ring must not exceed the depot capacity, and the cost of pending nodes is different from the cost of the connections within the rings. The minimum-cost design of the m-rings is called Capacitated m Ring Star Problem, termed here CmRSP, for short. This problem is the starting point of our study.

1.2 Structure of the Thesis

This thesis follows the Swedish style, and it is organized in three parts. These parts have been ordered according to the logic of studying the survivability in networks design first, then reliability related to survivable networks, and finally a generalization of a different more recent problem is dealt. Chapter 2 introduces the Capacitated Two-Node survivable Star Problem, Chapter 3 includes a hybrid metaheuristic point of view for its approximated resolution, while Chapter 4 shows a complete study of the problem. Chapter 5 introduces diameter constrained reliability in the Capacitated *m* Ring Star Problem and Chapter 6 deals with the Capacitated Two-Node Survivable Tree Problem.

1. In Part I we study a new problem of topological design of survivable networks. Chapter 2 introduces the CmTNSSP, Chapter 3 shows a resolution using a hybrid metaheuristic based on Integer Linear Programming and Chapter 4 is an exhaustive study of the problem, its formal definition and a proposal of an ILP model.

2. In Part II we study how to the Diameter Constrained Reliability impacts in the design of networks, particularly in the Capacitated *m* Ring Star Problem. This issue is addressed in the Chapter 5.

3. In Part III we study a relaxation of a more recent combinatorial optimization problem, the Capacitated Ring Tree Problem. In Chapter 6 we define the Capacitated Two-Node Survivable Tree Problem and we addressed a metaheuristic to its approximate resolution.

Each chapter includes a corresponding peer-reviewed article. They are all accepted and published (except the article from Chapter 6 which is submitted and at the time of writing this thesis there was no acceptance notification yet).

1.3 Main Contributions

1.3.1 Topological Design of Survivable Networks

Framework

In Chapter 2, a new problem is introduced and its resolution using a metaheuristic is achieved. Results are compared with a related problem, the C*m*RSP. A hybrid metaheuristic based on Integer Linear Programming (ILP) is proposed and implemented in Chapter 3. Chapter 4 deals with a deeper study of the C*m*TNSSP.

Contributions

Chapter 2: The Capacitated *m* Two-Node Survivable Star Problem

- A new problem of Topological Design of Survivable Networks is introduced.
- Taking into account the conclusions of Clyde Monma about the 2-node-connected graphs, the total cost of the solutions can outerperform those solutions that use exclusively cycles.

Chapter 3: The Capacitaded *m* Two-Node Survivable Star Problem: A hybrid metaheuristic approach

- We define two ILP models which are integrated in local searches used to resolves approximately the C*m*TNSSP.

Chapter 4: A complete study of the Capacitated *m* Two-Node Survivable Star Problem

- An ILP model of the C*m*TNSSP is defined and an exhaustive computational study of the proposed metaheuristic is made.

1.3.2 Diameter Constrained Reliability in Design of Networks

Framework

In Chapter 5, a new constraint is added to a known problem. The C*m*RSP under Diameter Constrained Reliability is defined adding DCR to the C*m*RSP. Here we use the best results obtained in the experimental analysis of Part I, to apply DCR and study the behavior of the problem under this restriction.

A Greedy Randomized Adaptative Search Procedure enriched with a Variable Neighborhood Descent (GRASP-VND) metaheuristic is proposed and implemented to resolve this problem.

Contributions

Chapter 5: Capacitated *m* Ring Star Problem under Diameter Constrained Reliability

- A combinatorial optimization problem is formally presented. The goal is to minimize the network cost regarding a defined maximum diameter and a minimum total reliability of the network. This work combines the topological design and the network reliability in the same problem.
- Solutions of the problem empirically show that CmTNSSP is more adequate to deal with considerations of diameter and reliability than CmRSP.

1.3.3 Generalizing other problem of Topological Network Design

Framework

In several real-world applications of topological networks design, we should use different types of terminal nodes and structures like trees and rings jointly. The Capacitated Ring Tree Problem (CRTP) deals with this topics. In Chapter 6, we propose a combinatorial optimization problem that generalizes the CRTP using 2-node-connected structures instead of purely rings.

Contributions

Chapter 6: The Capacitated Two-Node Survivable Star Problem

- The Capacitated Two-Node Survivable Star Problem (CTNSTP) is introduced.
- A GRASP suitably customized heuristic enriched with a VND and a post-optimization shaking scheme has been developed.
- The effectiveness of our metaheuristic has been tested by comparing global optima values of the CRTP.

1.4 Bibliography

Baldacci, R., Dell'Amico, M., and González, J. J. S. (2007). The capacitated m-ring-star problem. *Operations Research*, 55(6):1147–1162.

Bayá, G., Mauttone, A., and Robledo, F. (2017). The capacitated m two node survivable star problem. *Yugoslav Journal of Operations Research*, 27(2).

Bayá, G., Mauttone, A., Robledo, F., and Romero, P. (2016a). Capacitated m two-node survivable star problem. *Electronic Notes in Discrete Mathematics*, 52:253 – 260. {INOC} 2015 7th International Network Optimization Conference.

Bayá, G., Mauttone, A., Robledo, F., Romero, P., and Rubino, G. (2016b). Capacitated m ring star problem under diameter constrained reliability. *Electronic Notes in Discrete Mathematics*, 51:23 – 30. USRR 2014.

Bayá, G., Mauttone, A., Robledo, F., and Romero, P. G. (2015). Capacitated m two-node survivable star problem. In *INOC 2015 - 7th International Network Optimization Conference (INOC'15)*, Warsaw, Poland. Electronic Notes in Discrete Mathematics (ENDM).

Bayá, G., Mauttone, A., Robledo, F., and Romero, P. G. (2016). Capacitated m two-node survivable star problem: A hybrid metaheuristic approach. In *HM 2016 - 10th International Workshop on Hybrid Metaheuristics*, volume 9668, pages 171–186, Plymouth, United Kingdom. Springer Lecture Notes in Computer Science. (LNCS).

Bayá Mantani, G. (2014). Diseño Topológico de Redes. Caso de Estudio: Capacitated *m* Two-Node Survivable Star Problem. Master's thesis, Universidad de la República. Pedeciba Informática, Montevideo, Uruguay.

Bhandari, R. (1997). Optimal physical diversity algorithms and survivable networks. In *Computers and Communications, 1997. Proceedings., Second IEEE Symposium on*, pages 433–441.

Canale, E., Monzón, P., and Robledo, F. (2009). Global synchronization properties for different classes of underlying interconnection graphs for kuramoto coupled oscillators. In *Future Generation Information Technology*, volume 5899 of *Lecture Notes in Computer Science*, pages 104–111. Springer Berlin Heidelberg.

Canale, E., Robledo, F., Romero, P., and Sartor, P. (2014). Monte carlo methods in diameter-constrained reliability. *Optical Switching and Networking*, 14, Part 2(0):134 – 148. Special Issue on RNDM 2013.

Cancela, H., Khadiri, M. E., and Petingi, L. (2011). Polynomial-time topological reductions that preserve the diameter constrained reliability of a communication network. *IEEE Transactions on Reliability*, 60:845–851.

Cancela, H. and Petingi, L. (2004). Reliability of communication networks with delay constraints: computational complexity and complete topologies. *International Journal of Mathematics and Mathematical Sciences*, 2004:1551–1562.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410.

Dreyfus, S. E. and Wagner, R. A. (1971). The steiner problem in graphs. *Networks*, 1(3):195–207.

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

Frederickson, G. N. and Ja'Ja', J. (1981). Approximation algorithms for several graph augmentation problems. *SIAM Journal on Computing*, 10(2):270–283.

Hill, A. (2014). Multi-exchange neighborhoods for the capacitated ring tree problem. In *International Conference on Numerical Methods and Applications*, pages 85–94. Springer.

Hill, A. and Voß, S. (2016). Optimal capacitated ring trees. *EURO Journal on Computational Optimization*, 4(2):137–166.

Hoshino, E. A. and De Souza, C. C. (2012). A branch-and-cut-and-price approach for the capacitated m-ring-star problem. *Discrete Appl. Math.*, 160(18):2728–2741.

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

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

Labbé, M., Laporte, G., Martín, I. R., and González, J. J. S. (2004). The ring star problem: Polyhedral analysis and exact algorithm. *Networks*, 43(3):177–189.

Labbé, M., Laporte, G., Martín, I. R., and González, J. J. S. (2005). Locating median cycles in networks. *European Journal of Operational Research*, 160(2):457–470.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247.

Mauttone, A., Nesmachnow, S., Olivera, A., and Robledo, F. (2008). Solving a ring star problem generalization. In *Conference: 2008 International Conferences on Computational Intelligence for Modelling, Control and Automation (CIMCA 2008), Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC 2008), Innovation in Software Engineering (ISE 2008).*, pages 981–986.

Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Monma, C. L., Munson, B. S., and Pulleyblank, W. (1990). Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46(1-3):153–171.

Naji-Azimi, Z., Salari, M., and Toth, P. (2010). A heuristic procedure for the capacitated m-ring-star problem. *European Journal of Operational Research*, 207(3):1227–1234.

Naji-Azimi, Z., Salari, M., and Toth, P. (2012). An integer linear programming based heuristic for the capacitated m-ring-star problem. *European Journal of Operational Research*, 217(1):17–25.

Recoba, R., Robledo, F., Romero, P., and Viera, O. (2016). Two-node-connected star problem. *International Transactions in Operational Research*, pages n/a–n/a.

Reinelt, G. (1990). TSPLIB - A t.s.p. library. Technical Report 250, Universität Augsburg, Institut für Mathematik, Augsburg.

Resende, M. and Ribeiro, C. (2003). *Greedy Randomized Adaptive Search Procedures*. In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers.

Resende, M. and Ribeiro, C. (2014). Grasp: Greedy randomized adaptive search procedures. In Burke, E. K. and Kendall, G., editors, *Search Methodologies*, pages 287–312. Springer US.

Resende, M. G. (2009). Greedy randomized adaptive search procedures. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*, pages 1460–1469. Springer US.

Richey, M. B. (1990). Optimal location of a path or tree on a network with cycles. *Networks*, 20(4):391–407.

Robledo, F. (2005). *GRASP heuristics for Wide Area Network design*. PhD thesis, INRIA.

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

Stoer, M. (1992). *Design of Survivable Networks (Lecture Notes in Mathematics)*. Springer.

Suurballe, J. W. (1974). Disjoint paths in a network. *Networks*, 4(2):125–145.

Suurballe, J. W. and Tarjan, R. E. (1984). A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336.

Zhang, Z., Qin, H., and Lim, A. (2014). A memetic algorithm for the capacitated m-ring-star problem. *Appl. Intell.*, 40(2):305–321.

Zorpette, G. (1989). Keeping the phone lines open. *Spectrum, IEEE*, 26(6):32–36.

Part I

Designing survivable networks

Chapter 2

Capacitated m Two-Node Survivable Star Problem

A natural approach to reach two connectivity is to connect all terminals in a ring or cycle. In the physical design of a telephony deployment, it is useful to consider several rings joined to a perfect telephone exchange, but if some terminal nodes are far away, it is better to connect them in more than one ring. Other terminals can be connected to the ring using a simple link. CmRSP deals with such topologies. A relaxation of CmRSP is introduced in this chapter. The CmTNSSP problem belongs to the class of $\mathcal{NP}$-Hard problems.

Capacitated m Two-Node Survivable Star Problem

Gabriel Bayá, Antonio Mauttone, Franco Robledo, Pablo Romero [1]

Dpto. de Inv. Operativa. Universidad de la República. Montevideo, Uruguay

Abstract

A traditional method to connect multiterminal systems is to use rings. The goal of the Capacitated m Ring Star Problem (CmRSP) is to connect terminals by m rings joined only with a source node, and possibly some pending links, at minimum cost.

In this paper, we introduce a relaxation for the CmRSP, called Capacitated m Two-Node Survivable Star Problem (CmTNSSP for short). The CmTNSSP belongs to the class of $\mathcal{NP}$-Hard computational problems. Therefore, we address a heuristic GRASP resolution. In consonance with predictions provided by Clyde Monma, the network can be equally robust but cheaper than in the original CmRSP.

Keywords: Network Optimization, CmRSP, CmTSSP, GRASP.

1 Motivation

A natural approach to reach two connectivity is to connect all terminals in a ring or cycle in the cheapest way. This problem is called Traveling Salesman Problem, and it is widely studied in the scientific literature. A cornerstone in the area of topological network design was offered by Clyde Monma et.

[1] Email: (gbaya,mauttone,frobledo,promero)@fing.edu.uy

al [6]. They proved that a minimum-cost two-node connected metric network is either a Hamilton tour or presents a special graph topology as an induced subgraph. This topology is sketched in Figure 1. They are called Monma graphs for the first time in [3]. We will stick to this terminology.

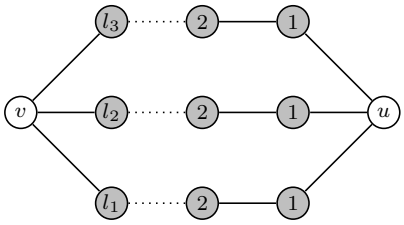

Fig. 1. Monma's graph structure.

In the physical design of a telephony deployment, it is useful to consider several two-connected component joined to a perfect telephone exchange, for if some terminal nodes are far away, it is better to connect them in more than one ring. A cost-effective "shape" of a solution is provided by Roberto Baldacci et. al. [1]. We are given a *depot*, several terminal nodes and optional nodes. In order to connect all terminals, the authors propose to find the cheapest m rings joined precisely in the depot, while some terminals can be pending on some node of a ring. The number of nodes within a ring must not exceed the depot capacity, and the cost of pending nodes is different than the cost of the connections within the rings. The minimum-cost design of the m-rings is called Capacitated m Ring Star Problem, termed here CmRSP for short.

Inspired in the potential savings predicted by Clyde Monma et. al., we relax the condition of rings, and consider arbitrary two-node connected components instead. The goal of this paper is to introduce the Capacitated m Two-Node Survivable Star Problem (CmTNSSP for short), solve it under classical instance provided by Baldacci et. al. and analyze the savings with respect to a pure ring topology. This article is organized as follows. The formal definitions for both problems, to know, CmRSP and CmTNSSP, are presented in Section 2. A greedy randomized adaptive search procedure (GRASP) is developed for its resolution in Section 3. A contrast between the design with rings (CmRSP) and the one with 2-node connected components (CmTNSSP) is presented in Section 4. Concluding remarks and trends for future work are discussed in Section 5.

2 Capacitated m-Ring Star Problem

We are given a simple graph $G = (V, E)$, a positive integer m and a tri-partition $V = \{s\} \cup V_S \cup V_T$, being s the depot, V_S optional Steiner nodes and V_T terminal nodes. The source s has a capacity q_s, and there are two classes of connections with different costs: ring-connections are given by a cost-matrix $R = (r_{i,j})$, $v_i, v_j \in V$; pending-connections are given by another cost-matrix $C = (c_{i,j})$, $v_i \in V - \{s\}$, $v_j \in V_T$. In the CmRSP, the goal is to choose a minimum cost spanning subgraph $H = \cup_{i=1}^{m} C_{l_i} \cup S_i$, where the C_{l_i}s are cycles that only meet on the depot $s \in C_{l_i}$ and have length l_i, and S_i are pending links from nodes belonging to C_{l_i}. The capacity constraint implies that $|S_i| + l_i \leq q_s$ for all $i \in \{1, \ldots, m\}$. The CmRSP belongs to the class $\mathcal{NP}$-Hard, since the Traveling Salesman Problem is included in CmRSP (choose $m = 1$ and a matrix C with infinite costs [1]). Therefore, the problem has been heuristically addressed in several opportunities [4,9].

If we consider arbitrary two-node connected components instead of the rings C_{l_i}, we obtain the Capacitated m Two-Node Survivable Star Problem (CmTNSSP). The CmTNSSP also belongs to the class of $\mathcal{NP}$-Hard problems, since the design of one component ($m = 1$, $q_s = +\infty$, $V_S = \emptyset$) is the minimum-cost two-connected spanning network problem, which is $\mathcal{NP}$-Hard [6].

3 GRASP Resolution

Greedy Randomized Adaptive Search Procedure (GRASP) is a powerful multi-start or iterative process, with great success in telecommunications [8]. In GRASP, feasible solutions are produced in a first phase, while neighbor solutions are explored in a second phase. The best overall solution is returned as the result. There is a trade-off between greediness (intensification) and randomization (diversification), by means of a restricted candidate list. We invite the reader to consult [7] for a comprehensive study of this metaheuristic. Here, we will sketch the main ingredients of our particular GRASP design, to know, Construction Phase and Local Search Phase.

3.1 Construction Phase

During the Construction Phase, components will be iteratively built, and no pending links will be considered. The goal is to produce a feasible solution, for the sake of higher costs (which will be reduced during Local Search Phase).

Let us consider an arbitrary instance for the CmTNSSP, a positive integer

k and a maximum number of iterations $MaxIter$. In order to define our construction phase, the following four functions will be used:

1 $Picking(m, G, R, MaxIter)$: returns m terminal nodes $v_1, \ldots, v_m$ from different components.

2 $Connecting(s, node, k)$: connects each node v_i with k node-disjoint paths with the source-node s.

3 $ChooseTwo(k)$: chooses 2 paths out of m uniformly at random.

4 $ConnectAllOthers$: connects nodes that are not included in the construction with some component.

Algorithm 1 Construction Phase

```
1: input G, C, k, m, iter
2: G_Sol ← ∅
3: component_nodes ← ∅
4: non_connected ← V_T
5: {v_1, ..., v_m} ← Picking(m, G, R, iter)
6: for i=1 to m do
7:     node = Random(v_1, ..., v_m)
8:     C ← Connecting(G, C, s, node, k, non_connected)
9:     C_i ← ChooseTwo(C)
10:    G_Sol ← G_Sol ∪ C_i
11:    component_nodes[i] ← component_nodes[i] ∪ C_i
12:    non_connected ← non_connected − C_i
13: end for
14: G_Sol ← G_Sol ∪ ConnectAllOthers(non_connected, G, C)
15: return G_Sol
```

The previous functions will be called sequentially. $Picking$ runs $MaxIter$ independent random sets of m terminal nodes. It returns the set with maximum global cost between all the pairs of the set. Once the set $v_1, \ldots, v_m$ is obtained, $Connecting(s, v_i, k)$ is called for each node v_i. It applies Ramesh Bhandari's algorithm [2] in order to find the cheapest set of k node-disjoint paths between the depot and terminal v_i. Function $ChooseTwo$ just chooses uniformly at random two disjoint paths out of k from each component. Finally, in $ConnnectAllOthers$, non-connected nodes are randomly chosen and iteratively added to the component with the least number of nodes. In this way, the capacity constraint is met during the construction phase, even though the cost could be high. Consider a non-connected node v and the (two-node connected) component C (Figure.2). All links that are part of other components will be deleted, and the costs of all links from C will be zero for a

moment. We add an artificial node v' connected with all nodes from C (this is a cone with ground set C and vertex v'). Bhandari's algorithm is applied in order to find k (or possibly less) node-disjoint paths between v and v' in the resulting network. Only two disjoint paths between v and v' will be uniformly chosen. Finally, the resulting links that connect v with C are added to the solution.

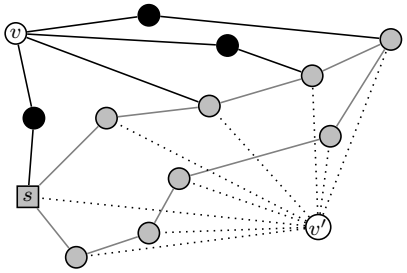

Fig. 2. Including node v into component C.

3.2 Local search phase

The five following functions determine different neighborhood structures, which are applied following a variable neighborhood descend order (sequentially; if there is an improvement we return to the first function again):

- *Swapping*: takes a random terminal node and swaps it with its closest possible terminal node (the possibility means that the cost is decreased),
- $Extract - Insert$: extracts the links of a node, reconnects its neighbors and greedily inserts the node (i.e., minimum cost insertion).
- *Crossing*: takes two close terminal nodes from different components, deletes one link incident from each node and reconnects components in the best manner,
- *BestPath* replaces a simple path with pendant nodes p, by the best of them (with the same ends), using an exact algorithm based on an Integer Linear Programming (ILP) model.
- *BestComponent*: each cycle is replaced by its best 2-node connected component, using and exact algorithm based on ILP.

These movements are explained in more detail in the thesis [5]. It is worth to remark that neighbor solutions are feasible, so feasibility is preserved during the local search phase. In order not to stuck in local optima, a perturbation process takes place. Function *Shake* randomly disconnects a percentage p of

terminal nodes and reconnects them in another way. *Shake* is called whenever the previous five functions are stucked in a solution and do not have activity (i.e., they do not produce better solutions). Figure 3 entails the full flow chart of the GRASP-VND metaheuristic.

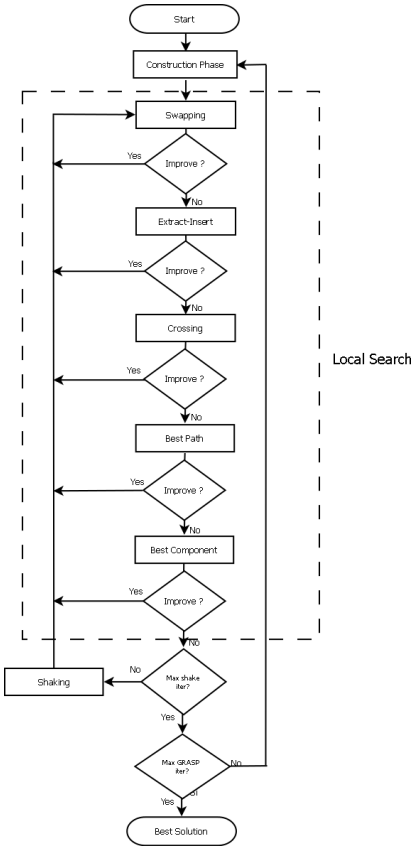

Fig. 3. Flow Diagram of GRASP-VND Metaheuristic for CmTNSSP.

4 Empirical Results

Observe that CmTNSSP is a relaxation of CmRSP, and the cost of feasible solutions for the CmTNSSP can be better than optimal solution of CmRSP. In order to highlight the main challenges of the new problem and the gap offered by our GRASP methodology, we will contrast against optimal solutions for the CmRSP, choosing instances developed by Roberto Baldacci, Mauro Dell' Amico and José Luis Salazar González [1]. The authors considered instances from TSPLIB. Instances are divided into two classes (A and B) using graphs with 26,51,76 and 101 nodes. Both classes have the same topology, but edge

costs are different. In class A, the cost of each link equals the Euclidean distance $r_{i,j} = c_{i,j} = d_{i,j}$, while in class B, $r_{i,j} = \lceil 7d_{i,j} \rceil$ and $c_{i,j} = \lceil 3d_{i,j} \rceil$. We used $m \in \{3, 4, 5\}$. The GRASP resolution has been executed using $k = 4$ for the restricted candidate list and $p = 0,3 \times |V_T|$ for shaking, which were tuned with other smaller TSPLIB instances from Classes A and B.

Table 1 presents a contrast between the optimum solution for the CmRSP ($\bar{Z}$) and the cost in CmTNSSP (Z_{best}) for those instances where improvements were obtained respect to best values of CmRSP, from a total of 90 instances tested. The acronyms PN, CN, SN stands for the number of Pending Nodes, Connected Nodes and Steiner Nodes in the solutions, respectively. The parameter *gap* is a measure of our GRASP-VND effectiveness, and it is defined as follows:

$$gap = \frac{Z_{best} - \bar{Z}}{\bar{Z}}. \tag{1}$$

INSTANCE	V_T	q_s	***CN***	***PN***	***SN***	Z_{best}	$\bar{Z}$	***gap %***	***t(s)***
A26-n076-m04	37	11	36	1	3	456	460	-0,870	7200.00
A29-n076-m04	56	16	49	7	1	519	523	-0,765	7200,00
A33-n076-m05	75	17	68	7	0	651	654	-0,459	7200.00
B26-n076-m04	37	11	34	3	4	3134	3138	-0,127	28825.80
B28-n076-m03	56	21	40	16	4	3044	3088	-1,425	28815.84
B29-n076-m04	56	16	44	12	2	3439	3447	-0,232	14418.05
B30-n076-m05	56	13	44	12	2	3635	3648	-0,356	3797.03
B31-n076-m03	75	28	55	20	0	3724	3740	-0,428	2112.23
B37-n101-m03	50	19	40	10	8	3331	3332	-0,030	7200,00

Table 1
Instances where GRASP-VND found better results than optimum in CmRSP.

5 Concluding Remarks and Future Work

The Capacitated m Two-Node Survivable Star Problem (CmTNSSP) has been introduced. As far as we know, it has not been studied in prior literature. The need for redundancy and cheaper costs in network deployment is remarkable. Inspired in predictions from Clyde Monma and the previous CmRSP, we propose an alternative problem, where rings are replaced by arbitrary two-node connected components. Both problems are computationally intractable. Therefore, heuristics are suitable for large case scenarios. As a corollary, the CmTNSSP has been heuristically addressed, following a GRASP metaheuristic enriched with a variable neighborhood descend (VND). The resulting topol-

ogy can be cheaper than the one offered from the CmRSP, and two-connected as well. As a future work, we wish to apply these techniques to the design of real-life networks. Indeed, optimal solutions for the CmTNSSP are both robust and cheaper than that of CmRSP. We encourage TELCOs operators to choose two-node connected components configured in star, mainly in the design of the physical layer for FTTH systems.

References

[1] Roberto Baldacci, Mauro Dell'Amico, and Juan José Salazar González. The capacitated m-ring-star problem. *Operations Research*, 55(6):1147–1162, 2007.

[2] Ramesh Bhandari. Optimal physical diversity algorithms and survivable networks. In *Computers and Communications, 1997. Proceedings., Second IEEE Symposium on*, pages 433–441, 1997.

[3] Eduardo Canale, Pablo Monzón, and Franco Robledo. Global synchronization properties for different classes of underlying interconnection graphs for kuramoto coupled oscillators. In *Future Generation Information Technology*, volume 5899 of *Lecture Notes in Computer Science*, pages 104–111. Springer Berlin Heidelberg, 2009.

[4] E. A. Hoshino and C. C. De Souza. A branch-and-cut-and-price approach for the capacitated m-ring-star problem. *Discrete Appl. Math.*, 160(18):2728–2741, December 2012.

[5] Gabriel Bayá Mantani. Diseño Topológico de Redes. Caso de Estudio: Capacitated m Two-Node Survivable Star Problem. Master's thesis, Universidad de la República. Pedeciba Informática, Montevideo, Uruguay, 2014.

[6] Clyde Monma, Beth Spellman Munson, and William R. Pulleyblank. Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46(1-3):153–171, 1990.

[7] Mauricio Resende and Celso Ribeiro. *Greedy Randomized Adaptive Search Procedures.* In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers, 2003.

[8] Franco Robledo. *GRASP heuristics for Wide Area Network design.* PhD thesis, INRIA/IRISA, Université de Rennes I, Rennes, France, 2005.

[9] Zizhen Zhang, Hu Qin, and Andrew Lim. A memetic algorithm for the capacitated m-ring-star problem. *Appl. Intell.*, 40(2):305–321, 2014.

Chapter 3

The Capacitaded m Two-Node Survivable Star Problem: A hybrid metaheuristic approach

The CmTNSSP belongs to the class of $\mathcal{NP}$-Hard problems. Therefore, we developed heuristic methods.

In Metaheuristics, hybridization comprises among others, the use of exact methods embedded in some local searches. We use a hybrid metaheuristic to solve the CmTNSSP using ILP.

The Capacitated m Two-Node Survivable Star Problem: A hybrid metaheuristic approach.

Gabriel Bayá, Antonio Mauttone, Franco Robledo, Pablo Romero

Dpto. de Inv. Operativa. Universidad de la República. Montevideo, Uruguay

(gbaya,mauttone,frobledo,promero)@fing.edu.uy

Abstract. In telecommunications, a traditional method to connect multiterminal systems is to use rings. The goal of the Capacitated m Ring Star Problem (CmRSP) is to connect terminals by m rings which meet at a distinguished node, and possibly by some pendant links, at minimum cost. In this paper, we introduce a relaxation for the CmRSP, called Capacitated m Two-Node Survivable Star Problem (CmTNSSP for short). The CmTNSSP belongs to the $\mathcal{NP}$-Hard class of computational problems. Therefore, we address a GRASP hybrid metaheuristic which alternates local searches that obtain incrementally better solutions, and exact resolution local searches based on Integer Linear Programming models. In consonance with predictions provided by Clyde Monma, the network can be equally robust but cheaper than in the original CmRSP.

Keywords: Network Optimization, CmRSP, CmTSSP, Hybrid Metaheuristics, GRASP, VND, ILP.

1 Motivation

A natural approach to reach two-node connectivity is to connect all terminals in a ring or cycle in an economic way. In this scenario nodes are connected to one another by two independent paths. This problem is called Traveling Salesman Problem, and it is widely studied in the scientific literature. Clyde Monma et. al [9] described what is considered to be a cornerstone in the area of topological network design. They proved that a minimum-cost 2-node-connected metric network is either a Hamiltonian Tour or presents a special graph topology as an induced subgraph. This topology is sketched in Figure 1; it was refered to as Monma graphs for the first time in [4]. We will stick to this terminology. In the physical design of a telephony deployment, it is useful to consider several 2-node-connected components joined to a perfect telephone exchange, and to connect some distant terminal nodes to some ring. A cost-effective "shape" of a solution is provided by Roberto Baldacci et. al. [1]. We are given a distinguished node (or *depot*), several terminal nodes and optional nodes. In order to connect all terminals, the authors propose to find the cheapest structure of m rings which share the depot, while some terminals can be pendant on some node of a ring.

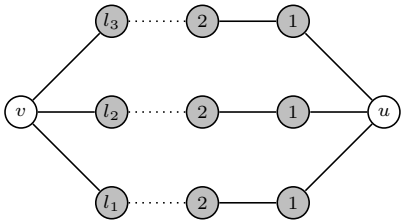

Fig. 1. Monma's graph structure.

The number of nodes within a ring must not exceed the depot capacity, and the cost of pendant edges is different than the cost of the edges within the rings. The minimum-cost design of the structure composed by the m rings and pendant nodes is called Capacitated m Ring Star Problem, termed herein CmRSP for short.

Inspired by the potential savings predicted by Clyde Monma et al, and supported by their theorem where the cost of the best ring could be even 4/3 times larger than the cost of the best 2-node-connected topology, we relaxed the condition of rings and considered arbitrary 2-node-connected components instead. The goal of this paper is to design a resilient cost-effective network from a topological stand point, suitable for delay sensitive applications on an Internet infrastructure. The main contributions are the following:

- The Capacitated m Two-Node Survivable Star Problem (CmTNSSP) is introduced.
- Given its intractability, a heuristic resolution is developed. We adopted a GRASP approach enriched with a Variable Neighborhood Descent, or GRASP-VND using some local searches based on Integer Linear Programming.
- A fair comparison with prior works in the area promotes the design of arbitrary 2-node-connected components, instead of rings (which were previously used by Baldacci et al.).

This article is organized in the following manner. The formal definitions for both problems, namely CmRSP and CmTNSSP, are presented in Section 2. A greedy randomized adaptive search procedure (GRASP) is developed for its resolution in Section 3. A comparison between the design with rings (CmRSP) and the one with arbitrary 2-node-connected components (CmTNSSP) is presented in Section 4. Concluding remarks and trends for future work are discussed in Section 5.

2 Capacitated m Two-Node Survivable Star Problem

Inspired by fiber optics design, Martine Labbé et. al. introduce the Ring Star Problem, or RSP for short [7]. The core is a ring, and the remaining termi-

nals are pendant from the ring. The goal is to find the minimum-cost topology meeting the previous constraints, given different costs in the ring-connections and pendant-connections. A further generalization, the CmRSP, is introduced by Roberto Baldacci et. al. [1]. The authors considered a depot and m rings, with the depot as the only common node. The main difference with the RSP is the presence of m rings instead of one. Both problems belong to the $\mathcal{NP}$-Hard class, since they represent a generalization of the Hamiltonian Tour [6]. Therefore, the CmRSP has been heuristically addressed in several opportunities [5, 13].

We are given a simple graph $G = (V, E)$, a positive integer m and a tripartition $V = \{s\} \cup V_S \cup V_T$, being s the depot, V_S the optional Steiner nodes and V_T the terminal nodes. The source s has a capacity q_s, and there are two classes of connections with different costs: ring-connections are given by a cost-matrix $R = (r_{i,j})$, $v_i, v_j \in V$ and pendant-connections are given by another cost-matrix $C = (c_{i,j})$, $v_i \in V - \{s\}$, $v_j \in V_T$. In the CmRSP, the goal is to choose a minimum cost spanning subgraph $H = \cup_{i=1}^{m} C_{l_i} \cup S_i$, wherein the C_{l_i}s are cycles that only meet on the depot $s \in C_{l_i}$ and have a length l_i, and the S_is are pendant links from nodes belonging to C_{l_i}. The capacity constraint implies that $|S_i| + l_i \leq q_s$ for all $i \in \{1, \ldots, m\}$.

If we consider arbitrary 2-node-connected components instead of the rings C_{l_i}, we obtain the Capacitated m Two-Node Survivable Star Problem (CmTNSSP). The CmTNSSP also belongs to the $\mathcal{NP}$-Hard class of problems, since the design of one component ($m = 1$, $q_s = +\infty$, $V_S = \emptyset$) is the minimum-cost 2-node-connected spanning network problem (MW2NCSN) , which is $\mathcal{NP}$-Hard. Monma et al. in their work [9] proved this for metric distances. They assigned a value 1 to the cost of the edges, then there exists a Hamiltonian cycle if and only if the minimum cost of MW2NCSN is equals to the number of nodes. Finally since "Hamiltonian Tour" belongs to Karp list [6] then MW2NCSN is $\mathcal{NP}$-Complete.

3 GRASP Resolution

Greedy Randomized Adaptive Search Procedure (GRASP) is a powerful multi-start or iterative process, with great success in telecommunications [12]. In GRASP, feasible solutions are produced in a first phase, while neighbor solutions are explored in a second phase. The best overall solution is returned as the result. There is a trade off between greediness (intensification) and randomization (diversification), by means of a restricted candidate list. For a comprehensive study of this metaheuristic see [10] and [11]. The main components of our particular GRASP design, namely Construction Phase and Local Search Phase, are depicted below.

3.1 Construction phase

In this phase we build a feasible solution (see Algorithm 1). Each one of the m components are iteratively added to the solution, starting with one ring per component and then adding paths between two nodes of the same component

until all terminal nodes are assigned. During the Construction Phase, no pendant links will be considered. The goal is to produce a feasible solution, despite the potential high cost of it (which will be reduced during Local Search Phase).

Let us consider an arbitrary instance for the CmTNSSP, a positive integer k and a maximum number of iterations *iter*. In order to define our construction phase, the following four functions will be used:

1 $Picking(m, G, R, iter)$: returns m terminal nodes $v_1, \ldots, v_m$, one for each component to build.
2 $Connecting(G, R, \hat{C}, s, node, k, non\_connected)$: connects each node v_i with the source-node s by k node-disjoint paths.
3 $ChooseTwo(\hat{C})$: randomly chooses 2 paths out of k using uniform distribution. At this stage one cycle per component is obtained.
4 $ConnectAllOthers(non\_connected, G, \hat{C})$: connects nodes that are not yet included in the construction with a component, adding a path between two nodes of such component.

Algorithm 1 Construction Phase

```
1:  input G, R, V_T, s, k, m, iter
2:  G_Sol ← ∅
3:  Ĉ ← ∅
4:  component_nodes[m] ← ∅ {Array with m empty positions}
5:  non_connected ← V_T
6:  {v_1, ..., v_m} ← Picking(m, G, R, iter)
7:  for i=1 to m do
8:      node = Random(v_1, ..., v_m)
9:      Ĉ ← Connecting(G, R, Ĉ, s, node, k, non_connected)
10:     C_i ← ChooseTwo(Ĉ)
11:     G_Sol ← G_Sol ∪ C_i
12:     component_nodes[i] ← component_nodes[i] ∪ C_i
13:     non_connected ← non_connected − C_i
14: end for
15: G_Sol ← G_Sol ∪ ConnectAllOthers(non_connected, G, Ĉ)
16: return G_Sol
```

The previous functions will be run sequentially. *Picking* function returns a set of m terminal nodes by considering *iter* sets of randomly chosen m nodes and returning the set with the greatest sum of costs of the edges determined by each pair of nodes (line 6).

Once the set $\{v_1, \ldots, v_m\}$ is obtained, *Connecting* function (line 9) connects node with the source-node s. Thus function is called for each *node* v_i which is selected randomly using the function *Random* (line 8). It applies Ramesh Bhandari's algorithm [3] in order to find the cheapest set of k node-disjoint paths between the depot and terminal node v_i.

Function $ChooseTwo$ (line 10) just chooses uniformly at random two disjoint paths out of k from each component. Up to this point m rings that share the depot have been built.

Finally, in $ConnnectAllOthers$ function (line 15), non-connected nodes are randomly chosen and iteratively added to the component with the lowest number of nodes. In this way, the capacity constraint is met during the construction phase, even though the cost could be high. Consider a non-connected node v and the (2-node-connected) component $\hat{C}$ (Figure 2). All links that belong to other components will be deleted (i.e. only one component is treated at a time), and the costs of all links from $\hat{C}$ (grey edges) will temporarily be zero. We add an artificial node v' connected with all nodes from $\hat{C}$ using edges at zero cost (dotted edges). Bhandari's algorithm is applied in order to find the better k (or possibly less) node-disjoint paths between v and v' in the resulting network. Only two disjoint paths between v and v' will be uniformly chosen. Finally, we delete node v' and the resulting two paths that connect v with C are added to the solution.

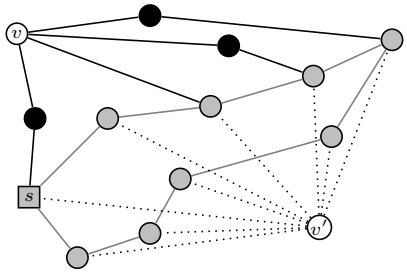

Fig. 2. Including node v into component $\hat{C}$.

3.2 Local search phase

The following operations fully determine neighborhood structures. A Variable Neighborhood Descent (VND [8]) scheme will be use to combine them.

- $Swapping(G, R, C, V_T, G_{sol})$: picks a random terminal node in G_{sol} and swaps it with its closest possible terminal node (the possibility means that the cost is decreased and the solution remains feasible),
- $ExtractInsert(G, R, C, V_T, G_{sol})$: extracts the links of a node, reconnects its neighbors and greedily inserts the node in G_{sol}(i.e., minimum cost insertion),
- $Crossing(G, R, C, V_T, G_{sol})$: picks two close terminal nodes from different components in G_{sol}, deletes one incident link from each node and reconnects components in the best manner,
- $BestPath(G, R, C, V_T, G_{sol})$ replaces any simple path with pendant nodes l in G_{sol} by the best of them (with the same endpoints), using an exact algorithm based on an Integer Linear Programming (ILP) model.

– $Best2NC(G, R, C, V_T, G_{sol})$: each cycle in G_{sol} is replaced by its best 2-node-connected component, using an exact algorithm based on ILP.

The full algorithm of GRASP-VND used in this paper, with the Construction phase and the sequence of local searches, is depicted in Algorithm 2.

Algorithm 2 Model of GRASP-VND used.

```
1: input G, R, C, V_T, s, k, m, iter, grasp_iter, shk_iter,
2: repeat
3:    G_sol ← Construction_phase(G, R, V_T, s, k, m, iter)
4:    G_iter ← G_sol
5:    repeat
6:       improve=true
7:       while improve do
8:          improve=Swapping(G, R, C, V_T, G_sol)
9:          if not improve then
10:            improve=ExtractInsert(G, R, C, V_T, G_sol)
11:            if not improve then
12:               improve=Crossing(G, R, C, V_T, G_sol)
13:               if not improve then
14:                  improve=BestPath(G, R, C, V_T, G_sol)
15:                  if not improve then
16:                     improve=Best2NC(G, R, C, V_T, G_sol)
17:                  end if
18:               end if
19:            end if
20:         end if
21:      end while
22:      if cost(G_sol) < cost(G_iter) then G_iter ← G_sol end if
23:      G_sol ← Shaking(C, R, V_T, G_sol)
24:   until shk_iter are reached
25:   if cost(G_iter) < cost(G_best) then G_best ← G_iter end if
26: until grasp_iter are reached
27: return G_best
```

The first three local searches involve moves that have been usually applied to several network-based combinatorial optimization problems and they are explained in more detail in the thesis of Gabriel Bayá [2]. The remaining two local searches are detailed below.

Best path with pendants This local search named $BestPath$, is based on an integer linear programming model. A preliminary concept is first introduced.

Definition 1 ***Path with pendant nodes****. Given an undirected graph* $G = (V, E)$ *we say that* G *is a path with pendant nodes which has endpoints* a *and* $z \in V$ *when there exists a path* $l(a, z) \subseteq G$ *that connects nodes* a *and* z *(which we call main path), and the following conditions are met:*

- G *is a tree.*
- *All nodes that do not belong to* l *are directly connected to some node of* l*.*

Given a feasible solution to the CmTNSSP we should identify all simple cycles that exist in each component and we should divide them in paths, adding their pendants nodes. Each path with pendants which has endpoints a and z is replaced by the best path with pendants with the same endpoints. This algorithm is based on an integer linear programming model.

We consider the following definitions:

Let $G = (V, E)$ be an undirected graph.

Let $\hat{T}$ be the set of terminal nodes of G.

Let $Adj(i)$ be the set of adjacent nodes to node $i \in V$ such:

$$Adj(i) = \{j \in V : (i, j) \in E\}$$

Let a and z be two distinguished terminal nodes such that $a \in \hat{T}$ and $z \in \hat{T}$.

Let $T = \hat{T} \setminus (\{a\} \cup \{z\})$ be the set of terminal nodes without a and z.

We define $R = \{r_{ij}\}_{i,j \in V}$ as the routing cost matrix of the graph, for each edge (i, j) which belongs to the main path $l(a, z)$.

Let us now define $C = \{c_{ij}\}_{i,j \in V}$ as the connection cost matrix of the graph, that is the cost of the edge (i, j) when one endpoint belongs to the main path and the other one does not belong to such path.

Let $W = V \setminus \hat{T}$ be the set of Steiner nodes. Let us now define the decision variables.

$$X_i = \begin{cases} 1 \text{ if node } i \in \hat{T} \text{ belongs to the main path} \\ 0 \text{ otherwise} \end{cases}$$

$$Y_i = \begin{cases} 1 \text{ if node } i \in T \text{ is a pendant node} \\ 0 \text{ otherwise} \end{cases}$$

$$z_{i,j} = \begin{cases} 1 \text{ if } i \in \hat{T} \text{ and } j \in V \text{ are connected, being } i \text{ a pendant node and} \\ \quad j \text{ j a node that belongs to the main path} \\ 0 \text{ otherwise} \end{cases}$$

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

$$w_{i,j} = \begin{cases} 1 \text{ if edge } (i, j) \text{ is a pendant edge and is used in the solution} \\ 0 \text{ otherwise} \end{cases}$$

$$y_{i,j}^{u,v} = \begin{cases} 1 \text{ if edge } (i, j) \text{ is used in the path that goes from node } u \text{ to node } v \\ 0 \text{ otherwise} \end{cases}$$

The integer linear programming model is defined as follows:

$$min(\sum_{i,j \in V} r_{ij}(x_{ij} - w_{ij}) + \sum_{i,j \in V} c_{ij} w_{ij}) \tag{1}$$

subject to:

$$X_i + Y_i = 1 \ \ \forall i \in T \tag{2}$$

$$X_i = 1 \ \ \forall i \in (\{a\} \cup \{z\}) \tag{3}$$

Equation 2 guarantees thay any terminal node which is not and endpoint either belongs to the main path or is pendant from the main path by a pendant edge, whereas constraint 3 ensures that the endpoints a and z belong exclusively to the main path.

$$z_{ij} \leq X_j \ \ \forall i \in T \quad \forall j \in Adj(i) \tag{4}$$

$$Y_i = \sum_{j \in Adj(i)} z_{ij} \quad \forall i \in T \tag{5}$$

$$\sum_{j \in V} w_{i,j} \leq Y_i \ \ \forall i \in T \tag{6}$$

Constraint 4 implies that if i and j are connected and node i is a pendant node then node j belongs to the main path. Constraint 5 implies that if node i is pendant from the main path then it does so only by one edge. Constraint 6 ensures there is only one edge incident to a pendant node.

$$z_{i,j} = w_{i,j} \ \ \forall i \in T \quad j \in Adj(i) \tag{7}$$

$$\sum_{j \in Adj(i)} x_{i,j} \leq M(1 - Y_i) + 1 \ \ \forall i \in T, \ \ M \in \mathbb{Z}^+,$$
$$M \geq max(|Adj(i)|) \quad i = 1 \cdots |V| \tag{8}$$

$$w_{i,j} \leq x_{i,j} \ \ \forall i \in T \quad j \in Adj(i) \tag{9}$$

Constraint 7 implies that if node i is pendant from node j then the edge (i, j) belongs to the solution. Inequality 8 constraints the degree of pendant nodes to

1 and it allows any other node of the main path to have any degree. Constraint 9 implies that if an edge is pendant then it belongs to the solution.

$$\sum_{j \in Adj[u]} y_{u,j}^{u,v} = 1 \ \ \forall u, v \in \hat{T}, u \neq v, \tag{10}$$

$$\sum_{i \in Adj[v]} y_{i,v}^{u,v} = 1 \ \ \forall u, v \in \hat{T}, v \neq u, \tag{11}$$

$$\sum_{i \in Adj[p]} y_{(i,p)}^{u,v} - \sum_{i \in Adj[p]} y_{p,i}^{u,v} \geq 0 \ \forall u, v \in \hat{T}, \forall p \in V \setminus u, v \tag{12}$$

$$y_{i,j}^{u,v} + y_{j,i}^{u,v} \leq x_{i,j} \ \forall u, v \in \hat{T}, u \neq v, \ \ \forall (i,j) \in E \tag{13}$$

Constraints 10 and 11 are simple connectivity constraints between nodes of any path (u, v). Constraint 12 is the balance equation of the internal nodes of the path. Constraint 13 guarantees that the path is edge-disjoint (i.e. a path which does not repeat any edge).

$$Y_i = 0 \ \ \forall i \in W \tag{14}$$

$$(\sum_{i \in Adj[j]} z_{i,j} + 2X_j - \sum_{i \in Adj[j]} x_{j,i} = 0) \ \ \forall j \in W \tag{15}$$

$$\sum_{i \in Adj[j]} (z_{i,j} + z_{j,i}) + 2X_j - \sum_{i \in Adj[j]} x_{j,i} = 0 \ \ \forall j \in T \tag{16}$$

$$\sum_{i \in Adj[j]} (z_{i,j}) + X_j - \sum_{i \in Adj[j]} x_{j,i} = 0 \ \ \forall j \in (\{a\} \cup \{z\}) \tag{17}$$

In Equation 14 it is ensured that Steiner nodes exclusively belong to the main path and constraints 15 to 17 are adjustment equations for Steiner, terminal and endpoint nodes. Algorithm 3 describes a local search which involves the replacement of a path with pendants by the best path with pendants. It begins by taking as input the graph G_{Sol}, which is a feasible solution of CmTNSSP. For each m components of G_{Sol} all of its cycles are counted, which are then identified and stored in the indexed list *all_cycles* (lines 3 and 4). Next, each of the cycles identified in the previous steps is treated, running the operations during **for** loop (lines 5 to 13) until all cycles are considered. Each cycle is divided into a certain number of paths of variable length. Next, we entered into a repetitive loop during the second **for** loop (lines 7 to 12), wherein each path obtained in the

previous step is added with pendant nodes present in G_{Sol}, using the function **add_pendants** (line 8) obtaining a path with pendants P. In the next step, we generated the graph H induced by nodes of the path with pendants P with respect to the original graph G (line 9). Graph H thus generated is input of stage **best_pwp** which returns the best path with pendants (line 10). To accomplished this goal, **best_pwp** resolves the integer linear programming model depicted in (1)-(17). In line 11 P is replaced by P_{best} obtaining a better solution cost G_{best}. After processing all paths within each cycle, the best solution G_{best} is returned (line 15).

Algorithm 3 Best path with pendant nodes.

```
1: input G, R, C, V_T, G_sol
2: G_best ← G_sol
3: q_cycles = cycles_count(G_sol) {Numbers of cycles of G_sol}
4: all_cycles ← cycles(G_sol) {Array with all cycles of G_sol}
5: for (i = 1 to q_cycles) do
6:    paths=divide_into_paths(all_cycles[i], q_paths)
7:    for (j= 1 to q_paths) do
8:       P ← add_pendants(G_sol, paths[j])
9:       H ← induced_graph_path(P, G)
10:      P_best ← best_pwp(G_sol, P, R, C, H)
11:      G_best ← G_best - P + P_best
12:   end for
13: end for
14: improve=(Cost(G_best) < Cost(G_sol))
15: return improve, G_best
```

Best 2-Connected Component This local search named $Best2NC$ is also based on integer linear programming. As in the previous local search, given a feasible solution to the problem, Algorithm 4 identifies all cycles that exist in each component. For each cycle we applied an exact algorithm getting the best replacement solution that changes a cycle by 2-node-connected topology.
As stated in Section 1, the best 2-node-connected solution covering a certain set of nodes is not necessarily a cycle, so this local search may include such topologies in our solution (see Figure 1). This algorithm takes as input the induced sub-graph of the original graph with nodes of the cycle and some Steiner nodes, and returns the best 2-node-connected sub-graph, i.e it can potentially change a cycle by a 2-node-connected topology if such change improves solution costs. In order to model this local search we used a particular case of **GSP** (General Steiner Problem) wherein connectivity of all its terminal nodes is two. We considered the following definitions:

Let $G = (V, E)$ be an undirected graph where V is the set of vertices and E is the set of edges of graph G.

Let $\hat{T}$ be the set of terminal nodes of graph G.

Define $R = \{r_{ij}\}_{i,j \in V}$ as the routing cost matrix, i.e. the costs when edge (i, j) belongs to the 2-node-connected structure of the component. In this local search, we only used such routing cost matrix since pendant nodes hitherto generated were not considered.

Model variables are defined below.

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

$$y_{i,j}^{u,v} = \begin{cases} 1 \text{ if edge } (i,j) \text{ is used in a path from node } u \text{ to } v \\ 0 \text{ otherwise} \end{cases}$$

Once the variables were specified, the integer linear programming model was defined as follows:

$$min(\sum_{i,j \in V} r_{ij} x_{ij}) \tag{18}$$

subject to:

$$\sum_{j \in Adj[u]} y_{u,j}^{u,v} = 2 \ \ \forall u, v \in \hat{T}, u \neq v, \tag{19}$$

$$\sum_{i \in Adj[v]} y_{i,v}^{u,v} = 2 \ \ \forall u, v \in \hat{T}, v \neq u, \tag{20}$$

$$\sum_{i \in Adj[p]} y_{i,p}^{u,v} - \sum_{i \in Adj[p]} y_{p,i}^{u,v} \geq 0 \ \forall u, v \in \hat{T}, \forall p \in V \setminus u, v \tag{21}$$

$$y_{i,j}^{u,v} + y_{j,i}^{u,v} \leq x_{i,j} \ \forall u, v \in \hat{T}, u \neq v, \ \ \forall (i,j) \in E \tag{22}$$

Analogously to Algorithm 3, Algorithm 4 counts and identifies the cycles present in G_{sol} (lines 3 and 4). For each of these cycles the stage **best_component** (line 6) returns the best 2-node-connected structure and the cycle is replaced by the latter (performed in line 7). The function **best_component** resolves the integer linear programming model depicted in (18)-(22). It should be noted that neighbor solutions are feasible, so feasibility is preserved during the local search phase.

Algorithm 4 Best 2-node-connected component.

```
1: input G, G_sol
2: G_best ← G_sol
3: q_cycles = cycles_count(G_sol) {Number of cycles of G_sol}
4: all_cycles ← cycles(G_sol) {Array with cycles of G_sol}
5: for (i = 1 to q_cycles) do
6:    best = best_component(G_best, G, R, all_cycles(i))
7:    G_best ← G_best - all_cycles(i) + best
8: end for
9: improve=(Cost(G_best) < Cost(G_sol))
10: return improve, G_best
```

In order not to get stuck in a local optimum, a perturbation process takes place. Function *Shaking* randomly disconnects a proportion h of terminal nodes in the local optimal solution and reconnects them otherwise. *Shaking* is called whenever the previous five functions are stuck in a solution and do not have activity (i.e., they do not produce better solutions) as it can be seen in Algorithm 2.

4 Empirical Results

It must be observed that CmTNSSP is a relaxation of CmRSP. Therefore, the cost of feasible solutions for the CmTNSSP could be better than optimal values for the CmRSP. In order to highlight the main challenges of the new problem and the improvement offered by our GRASP methodology, we made a comparison with optimal solutions for the CmRSP, choosing instances developed by Roberto Baldacci, Mauro Dell' Amico and José Luis Salazar González in [1]. The authors considered instances from TSPLIB. Such instances are divided into two classes (A and B) using graphs with 26, 51, 76 and 101 nodes. Both classes have the same topology, but edge costs are different. In class A, the cost of each link equals the Euclidean distance $d_{i,j} = r_{i,j} = c_{i,j}$, while in class B, $r_{i,j} = \lceil 7d_{i,j} \rceil$ and $c_{i,j} = \lceil 3d_{i,j} \rceil$. We used $m \in \{3, 4, 5\}$. The GRASP algorithm has been executed using $k = 4$ for the restricted candidate list and $h = \lfloor 0,3 \times |V_T| \rfloor$ for shaking, which were tuned with other smaller TSPLIB instances from Classes A and B. The heuristic was fully coded in C language using the CPLEX Callable Library to resolve integer linear programming models. Hardware where algorithms were run, consists of a computer with Intel I7 processor with 8 Gb. RAM and OS Fedora Core 20.

Tables 1 and 2 present a comparison between the optimal solution for the CmRSP (Z_1) found by Baldacci et al [1] and the cost in CmTNSSP (Z_{best}) found by our proposed algorithm, for instances of Classes A and B, from a total of 90 instances tested; $\hat{Z}$ is the mean of 20 independent experiments for each instance and Z_2 is the best known value for CmRSP, recently published in [13]. The acronyms PN, CN, SN stand for the number of Pending Nodes, Connected Nodes and Steiner Nodes in the solutions, respectively.

INSTANCE	$\|T\|$	Q	*CN*	*PN*	*SN*	$\tilde{Z}$	Z_{best}	Z_1	Z_2	*gap %*	*t(s)*
A01-n026-m03	12	5	12	0	1	242	242	242	242	0,000	1.61
A02-n026-m04	12	4	12	0	1	261	261	261	261	0,000	0.97
A03-n026-m05	12	3	12	0	1	292	292	292	292	0,000	13.77
A04-n026-m03	18	7	18	0	0	301	301	301	301	0,000	34.29
A05-n026-m04	18	5	18	0	0	339	339	339	339	0,000	62.58
A06-n026-m05	18	4	18	0	0	375	375	375	375	0,000	2.67
A07-n026-m03	25	10	24	1	0	325	325	325	325	0,000	14.06
A08-n026-m04	25	7	25	0	0	362	362	362	362	0,000	3.99
A09-n026-m05	25	6	25	0	0	383	382	382	382	0,000	3.99
A10-n051-m03	12	5	12	0	0	242	242	242	242	0,000	20.09
A11-n051-m04	12	4	12	0	3	261	261	261	261	0,000	6.42
A12-n051-m05	12	3	11	1	2	286	286	286	286	0,000	37.69
A13-n051-m03	25	10	22	3	3	322	322	322	322	0,000	130.85
A14-n051-m04	25	7	24	1	1	360	360	360	360	0,000	49.75
A15-n051-m05	25	6	23	2	2	379	379	379	379	0,000	117.67
A16-n051-m03	37	14	33	4	1	373	373	373	373	0,000	296.60
A17-n051-m04	37	11	33	4	1	405	405	405	405	0,000	80.49
A18-n051-m05	37	9	33	4	1	434	432	432	432	0,000	2720.60
A19-n051-m03	50	19	45	5	0	461	458	458	458	0,000	1674.86
A20-n051-m04	50	14	48	2	0	492	490	490	490	0,000	3429.11
A21-n051-m05	50	12	43	7	0	521	520	520	520	0,000	6338.64
A22-n076-m03	18	7	17	1	5	332	330	330	330	0,000	36.13
A23-n076-m04	18	5	15	3	7	385	385	385	385	0,000	112.97
A24-n076-m05	18	4	17	1	4	448	448	448	448	0,000	109.91
A25-n076-m03	37	14	35	2	2	403	403	402	402	0,249	3624.35
A26-n076-m04	**37**	**11**	403**36**	**1**	**3**	**458**	**456**	**460**	**457**	**-0,870**	**7200.00**
A27-n076-m05	37	9	36	1	4	483	483	479	479	0,835	7200.00
A28-n076-m03	56	21	48	8	1	474	474	471	471	0,637	7200.00
A29-n076-m04	**56**	**16**	**49**	**7**	**1**	**522**	**519**	**523**	**519**	**-0,765**	**7200,00**
A30-n076-m05	56	13	50	6	2	555	547	545	545	0,367	7200.00
A31-n076-m03	75	28	71	4	0	572	571	564	564	1,241	7200.00
A32-n076-m04	75	21	73	2	0	614	611	606	602	1,808	7200.00
A33-n076-m05	**75**	**17**	**68**	**7**	**0**	**657**	**651**	**654**	**640**	**-0,459**	**7200.00**
A34-n101-m03	25	10	21	4	7	370	363	363	363	0,000	199.27
A35-n101-m04	25	7	21	4	9	417	415	415	415	0,000	1023.84
A36-n101-m05	25	6	22	3	9	453	448	448	448	0,000	1264.62
A37-n101-m03	50	19	46	4	8	503	500	500	500	0,000	4020.65
A38-n101-m04	50	14	47	3	6	545	538	532	528	1,128	7200.00
A39-n101-m05	50	12	46	4	5	578	573	568	567	0,880	7200.00
A40-n101-m03	75	28	69	6	5	616	613	595	595	3,025	7200.00
A41-n101-m04	75	21	73	2	1	656	651	625	623	4,160	7200.00
A42-n101-m04	75	17	70	5	2	680	677	662	657	2,266	7200.00
A43-n101-m03	100	38	84	16	0	665	662	646	646	2,477	7200.00
A44-n101-m04	100	28	87	13	0	684	680	680	679	0,000	7200.00
A45-n101-m05	100	23	84	16	0	722	713	700	700	1,857	7200.00

Table 1. Values found for Class A instances.

The parameter *gap* is a measurement of our GRASP-VND effectiveness, and it is defined as follows:

$$gap = \frac{Z_{best} - Z_1}{Z_1}. \tag{23}$$

In particular, Tables 1 and 2 show the gaps with respect to Z_1, where negative values are highlighted in boldface. We can observe that for Class A, the objective value obtained for the CmTNSSP is lower than its counterpart for CmRSP in 3 instances and equal in 29 instances out of 45 with an average gap of 0.741 %. For Class B, the same fact can be observed in 6 and 21 respectively out of 45 instances with an average gap of 0.890 %, suggesting that the cost structure of this class promotes the application of CmTNSSP solutions.

INSTANCE	$\|T\|$	Q	*CN*	*PN*	*SN*	$\hat{Z}$	Z_{best}	Z_1	Z_2	*gap %*	*t(s)*
B01-n026-m03	12	5	11	1	1	1684	1684	1684	1684	0,000	3.09
B02-n026-m04	12	4	12	0	1	1827	1827	1827	1827	0,000	1.09
B03-n026-m05	12	3	11	1	2	2041	2041	2041	2041	0,000	10.68
B04-n026-m03	18	7	17	1	1	2104	2104	2104	2104	0,000	24.90
B05-n026-m04	18	5	17	1	1	2370	2370	2370	2370	0,000	78.21
B06-n026-m05	18	4	17	1	2	2615	2615	2615	2615	0,000	47.01
B07-n026-m03	25	10	24	1	0	2251	2251	2251	2251	0,000	35.13
B08-n026-m04	25	7	24	1	0	2512	2510	2510	2510	0,000	51.65
B09-n026-m05	25	6	25	0	0	2677	2674	2674	2674	0,000	150.31
B10-n051-m03	12	5	10	2	2	1681	1681	1681	1681	0,000	2035.19
B11-n051-m04	12	4	10	2	3	1821	1821	1821	1821	0,000	49.26
B12-n051-m05	12	3	10	2	2	1976	1975	1972	1972	0,152	930.42
B13-n051-m03	25	10	21	4	3	2176	2176	2176	2176	0,000	1724.28
B14-n051-m04	25	7	22	3	3	2471	2470	2470	2470	0,000	626.97
B15-n051-m05	25	6	21	4	4	2596	2579	2579	2579	0,000	92.66
B16-n051-m03	37	14	29	8	2	2498	2490	2490	2490	0,000	3699.45
B17-n051-m04	37	11	29	8	2	2747	2735	2721	2721	0,515	3605.47
B18-n051-m05	37	9	32	5	2	2931	2908	2908	2908	0,000	197.51
B19-n051-m03	50	19	39	11	0	3028	3015	3015	3015	0,000	871.33
B20-n051-m04	50	14	39	11	0	3284	3267	3260	3260	0,215	7200,00
B21-n051-m05	50	12	38	12	0	3426	3404	3404	3404	0,000	3773.22
B22-n076-m03	18	7	15	3	4	2258	2253	2253	2253	0,000	186.10
B23-n076-m04	18	5	13	5	8	2661	2620	2620	2620	0,000	90.78
B24-n076-m05	18	4	15	3	9	3142	3155	3059	3059	3,138	7200,00
B25-n076-m03	37	14	32	5	6	2747	2731	2720	2720	0,404	7200,00
B26-n076-m04	**37**	**11**	**34**	**3**	**4**	**3142**	**3134**	**3138**	**3100**	**-0,127**	**7200.00**
B27-n076-m05	37	9	36	1	3	3327	3329	3311	3284	0,544	7217.19
B28-n076-m03	**56**	**21**	**40**	**16**	**4**	**3060**	**3044**	**3088**	**3044**	**-1,425**	**7200.00**
B29-n076-m04	**56**	**16**	**44**	**12**	**2**	**3448**	**3439**	**3447**	**3415**	**-0,232**	**7200.00**
B30-n076-m05	**56**	**13**	**44**	**12**	**2**	**3676**	**3635**	**3648**	**3632**	**-0,356**	**3797.03**
B31-n076-m03	**75**	**28**	**55**	**20**	**0**	**3742**	**3724**	**3740**	**3652**	**-0,428**	**2112.23**
B32-n076-m04	75	21	57	18	0	4102	4096	4026	3964	1,739	7200,00
B33-n076-m05	75	17	58	17	0	4512	4489	4288	4217	4,688	7200,00
B34-n101-m04	25	7	19	6	9	2452	2445	2434	2434	0,369	7200,00
B35-n101-m04	25	7	19	6	6	2804	2795	2782	2782	0,467	7200,00
B36-n101-m05	25	6	18	7	4	3015	3009	3009	3009	0,000	597.71
B37-n101-m03	**50**	**19**	**40**	**10**	**8**	**3338**	**3331**	**3332**	**3322**	**-0,030**	**7200,00**
B38-n101-m04	50	14	38	12	8	3616	3560	3533	3533	0,764	7200,00
B39-n101-m05	50	12	41	9	8	3895	3873	3872	3834	0,026	7200,00
B40-n101-m03	75	28	68	7	5	3958	3931	3923	3887	0,204	7200,00
B41-n101-m04	75	21	68	7	6	4345	4332	4125	4082	5,018	7200,00
B42-n101-m05	75	17	69	6	6	4556	4494	4458	4358	0,808	7200,00
B43-n101-m03	100	38	96	4	0	4413	4403	4110	4110	7,129	7200,00
B44-n101-m04	100	28	95	5	0	4560	4526	4506	4355	0,444	7200,00
B45-n101-m05	100	23	96	4	0	4645	4639	4632	4565	0,151	7200,00

Table 2. Values found for Class B instances.

5 Concluding Remarks and Future Work

The Capacitated m Two-Node Survivable Star Problem (CmTNSSP) has been introduced. As far as we are know, it has not been studied in prior literature. The need for redundancy and cheaper costs in network deployment is remarkable. Inspired by predictions from Clyde Monma and the previous CmRSP, we proposed an alternative problem, where rings are replaced by arbitrary 2-node-connected components. Both problems are computationally intractable. Therefore, heuristics are suitable for large case scenarios. As a corollary, the CmTNSSP has been heuristically addressed, following a hybrid GRASP metaheuristic that combines the resolutions of ILP models. The resulting topology could be cheaper than the one offered by the CmRSP but 2-node-connected as well. As a future work, we wish to apply these techniques to the design of real-life networks. Indeed, opti-

mal solutions for the CmTNSSP could by equally robust and more cost-effective than that of CmRSP.

References

1. Roberto Baldacci, Mauro Dell'Amico, and Juan José Salazar González. The capacitated m-ring-star problem. *Operations Research*, 55(6):1147–1162, 2007.
2. Gabriel Bayá Mantani. Diseño Topológico de Redes. Caso de Estudio: Capacitated m Two-Node Survivable Star Problem. Master's thesis, Universidad de la República. Pedeciba Informática, Montevideo, Uruguay, 2014.
3. Ramesh Bhandari. Optimal physical diversity algorithms and survivable networks. In *Second IEEE Symposium on Computers and Communications, 1997. Proceedings.*, pages 433–441, 1997.
4. Eduardo Canale, Pablo Monzón, and Franco Robledo. Global synchronization properties for different classes of underlying interconnection graphs for kuramoto coupled oscillators. In *Future Generation Information Technology*, volume 5899 of *Lecture Notes in Computer Science*, pages 104–111. Springer Berlin Heidelberg, 2009.
5. Edna Ayako Hoshino and Cid Carvalho de Souza. A branch-and-cut-and-price approach for the capacitated m-ring-star problem. *Discrete Appl. Math.*, 160(18):2728–2741, December 2012.
6. Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
7. Martine Labbé, Gilbert Laporte, Inmaculada Rodríguez Martín, and Juan José Salazar González. The ring star problem: Polyhedral analysis and exact algorithm. *Networks*, 43(3):177–189, 2004.
8. Nenad Mladenovic and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100, 1997.
9. Clyde Monma, Beth Spellman Munson, and William R. Pulleyblank. Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46(1-3):153–171, 1990.
10. Mauricio Resende and Celso Ribeiro. *Greedy Randomized Adaptive Search Procedures.* In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers, 2003.
11. Mauricio Resende and Celso. Ribeiro. Grasp: Greedy randomized adaptive search procedures. In Edmund K. Burke and Graham Kendall, editors, *Search Methodologies*, pages 287–312. Springer US, 2014.
12. Franco Robledo. *GRASP heuristics for Wide Area Network design.* PhD thesis, INRIA/IRISA, Université de Rennes I, Rennes, France, 2005.
13. Zizhen Zhang, Hu Qin, and Andrew Lim. A memetic algorithm for the capacitated m-ring-star problem. *Appl. Intell.*, 40(2):305–321, 2014.

Chapter 4

A complete study of the Capacitated m Two-Node Survivable Star Problem

The Capacitated m Two-Node Survivable Star Problem is not defined previosuly in the literature. An exact model based on ILP for this problem is introduced and results of its approximated resolution are contrasted with the results of the exact resolution of the CmRSP.

Yugoslav Journal of Operations Research

xx (yyyy), Number nn, zzz–zzz

DOI: (will be added later)

THE CAPACITATED m TWO NODE SURVIVABLE STAR PROBLEM

Gabriel BAYÁ
Departamento de Investigación Operativa
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
Montevideo-Uruguay
gbaya@fing.edu.uy

Antonio MAUTTONE
Departamento de Investigación Operativa
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
mauttone@fing.edu.uy

Franco ROBLEDO
Departamento de Investigación Operativa
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
frobledo@fing.edu.uy

Received: Month yyyy / Accepted: Month yyyy

Abstract: In this paper, we address the problem of network design with redundant connections, often faced by operators of telephone and internet services. The network connects customers with one master node and it is built by taking into account the rules that shape its construction, such as number of customers, number of components and types of links, in order to meet operational needs and technical constraints. We propose a combinatorial optimization problem called CmTNSSP (Capacitated m Two-Node-Survivable Star Problem), a relaxation of CmRSP (Capacitated m Ring Star Problem). In this variant of CmRSP, the rings are not constrained to be cycles; instead, they can be two-node connected components. The contributions of this paper are: (a) the introduction and definition of a new problem, (b) the specification of a mathematical programming model of the problem to be treated, and (c) the approximate resolution thereof through a GRASP metaheuristic,

which alternates local searches that obtain incrementally better solutions, and exact resolution local searches based on mathematical programming models, particularly Integer Linear Programming ones. Computational results obtained by the developed algorithms show robustness and competitiveness when compared to results of the literature relative to benchmark instances. Likewise, the experiments show the relevance of considering the specific variant of the problem studied in this work.

Keywords: Topological Network Design, Survivability, Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Search (VNS), Metaheuristics.

MSC: 90B06, 90C05, 90C08.

1. INTRODUCTION

Along with the evolution of telephone communications, the development of computers and digital data transmission has also begun. To communicate two remote computers, the telephone network was used as a transmission medium. This fact generated a number of associated services settled in a communications infrastructure, whose growth was not sufficiently planned. The lack of planning led to occurance of the events with devastating consequences. One example is the burning of a telephone exchange in a suburb of Chicago in May 1988, which rendered uncommunicated 35,000 local subscribers and affected 120,000 long distance trunk lines, compromising the functioning at O'Hare air traffic control and outaging the 911 service, as detailed in [21]. These accidents reveal, among other things, the need for proper planning of telephone networks and data transmission. Beyond all preventive actions that can be taken to avoid accidents as the one quoted above, a key element to mitigate such impact is a proper design of telecommunication networks. The study of the structure, the introduction of minimum levels of connectivity between their nodes, and redundancy are crucial to avoid catastrophic events in case of a failure. The main motivation for studying topological network design is its application in the area of telecommunications [19]. Basically, the goal is to obtain structures with the desired level of redundancy and fault-tolerance in some of their nodes or links, and to allow savings in construction costs. Initially, topological network design covered mainly availability aspects (e.g. public switched telephone network). However, new applications over the Internet infrastructure reveal the shortcomings of tree-like structures. On the other hand, mesh-like structures present valuable connectivity properties, but their deployment is prohibitively expensive. A natural approach to an acceptable level of connectivity is to connect all terminals in a ring or a cycle in the cheapest way. This problem, known as Traveling Salesman Problem [4], is widely studied in the scientific literature. In the physical design of a telephony deployment, it is useful to consider several two-connected components joined to a perfect telephone exchange, but if some terminal nodes are far away from each other, it is better to connect them in more than one ring. A cost-effective "shape" of a solution is provided in [1], where given a *depot*, several terminal nodes, and optional nodes, in order to connect all terminals, the authors propose to find the cheapest m rings joined in the depot, while some terminals can be pending on some node of a ring. The number of nodes within a ring must not exceed the depot capacity, and the cost of pending nodes is different from the cost of the connections within the rings. The

minimum-cost design of the m-rings is called Capacitated m Ring Star Problem, termed here CmRSP, for short. Furthermore, a cornerstone in the area of topological network design was offered in [12]. The authors fully characterize the structure of minimum-cost two-node connected sub-networks in metric graphs. They proved that a minimum-cost two-node connected metric network is either a Hamiltonian tour or presents a special graph topology as an induced sub-graph, sketched in Figure 1. Motivated by this result, we studied a problem with two-node-connected structures that can potentially have lower cost than the cost of cycles.

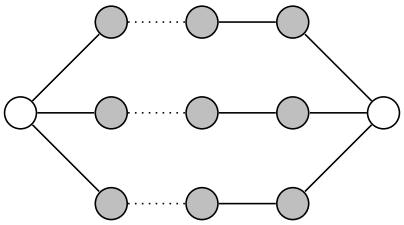

Figure 1: Monma's graph structure.

We have not found references in the literature for the *Capacitated m Two-Node-Survivable Star Problem* itself. The related work developed in [1] treats the exact resolution of the *Capacitated m Ring Star Problem*. Such problem is slightly different from problem treated in this paper. In CmRSP, 2-node-connected structures are exclusively cycles, whereas in our problem (CmTNSSP), other two-node-connected structures are allowed. The CmTNSSP is therefore a CmRSP relaxation. In [1], two mathematical programming formulations of CmRSP are considered to solve the problem exactly. The authors propose a set of test instances comprising up to 100 nodes. Some authors also treat CmRSP and solve it exactly [7], while other authors do it by using approximate methods. For example we can cite [13] and [10], who use iterated heuristics and the GRASP metaheuristic, respectively. Moreover, in [14], integer linear programming (ILP) heuristics for the CmRSP are proposed; also, the authors proposed larger instances comprising up to 200 nodes. More recently, in [20] a memetic algorithm is proposed, which improves previous results; also, the authors explore new instances comprising new cost structures.

There are studies that share some common characteristics with the CmRSP. The problem of *Locating Median Cycles in Networks* is a particular case of CmRSP and is studied in [9]. In that work, the authors seek to build a network which consists of a main loop and nodes attached to it, whose total cost should be minimum. Cost of the network is the cost of the edges that belong to the cycle (routing costs) plus the costs of connection of the edges with incidence in attached nodes. Here, the total connection cost is bounded to a given value. In [8], the same authors solve the RSP (*Ring Star Problem*), without imposing cost constraints on the edges that do not belong to the cycle. Only service constraint are considered in this problem, such as number of attached nodes connected to the same node belonging to the cycle. In that study, the RSP is solved exactly. Other similar problems, with differences in the structures, are discussed in [17]. In the CmRSP and in the CmTNSSP (the problem addressed in this paper), the structure of feasible solutions are cycles or two-connected structures, while in the problems mentioned above, they are

simple connected structures without redundancy such as paths or trees.

In this paper we propose an alternative (to the best of our knowledge not yet studied) to design 2-node-connected low-cost solutions, useful in the context of telecommunications networks with some required level of survivability. We define the CmTNSSP and propose an ILP model to solve exactly small instances. Also, we propose and implement a hybrid metaheuristic which is then applied to known instances from literature, and to other tests cases specifically designed. This article is organized as follows. The description and formal definition of the problem are presented in Section 2. An integer lineal programming model is presented in Section 3. A GRASP-VND metaheuristic is developed for the approximated resolution in Section 4. Computational results are reported in Section 5. Finally, conclusions and trends for future work are discussed in Section 6.

2. PROBLEM DEFINITION

The problem to be described aims to constitute a planning framework that must be followed to build fault-tolerant networks that meet some operational needs and technical constraints.

2.1. Problem description

Given a simple non directed graph $G = (V, E)$ with a set of vertices V and a set of edges E, we want to get a sub-graph (network) that meets certain topology, formally defined in Section 2.2. In this graph G we have a distinguished node d that we call depot. Within the scope of this article the term node is used to refer to any vertex within the set of vertices of any of the defined graphs. Both terms will be used interchangeably. The set of remaining vertices $V \setminus \{d\}$ will be partitioned into two disjoint sets, one called, the terminal nodes T, and the other, called the auxiliary or Steiner nodes W. Terminal nodes must be necessarily present in the network, and auxiliary ones participate in the solution only if its inclusion improves construction costs of such network.

A feasible solution consists of a certain number m of related sub-graphs, which will share the d node, so that if we remove this node, the resulting graph would be divided into m connected-components. Each component connects the depot d with a set of terminal nodes which cardinality cannot exceed a given capacity Q. This parameter narrows the number of nodes of each component in response to connection constraints and latency in communications. Terminal nodes present in each of these m connected-components either belong to an associated structure with redundancy which is part of the component, or are attached to such structure by an edge. In this associated structure with redundancy, every pair of vertices are connected by two independent paths. Steiner nodes, if included, can belong to redundancy associated structures but cannot be attached to these structures by any edge.

The graph G has two associated matrix costs. One of them determines the cost of connecting each pair of vertices if both are part of the related structure with redundancy (routing costs), and the other determines the cost of connecting a pair of vertices if one of them is attached to the structure by an edge (connection costs). Usually, when designing networks the cost of the core routers is greater than the cost of access routers, therefore this situation is covered by the definition of different costs.

Our problem consist in getting a sub-graph of G, which is of minimum cost and built under the above assumptions. We will call this problem *Capacitated m Two Node Survivable Star Problem* (CmTNSSP). In Figure 2, we can see an example of a feasible solution, where the rectangular node is the depot, black nodes are terminals and the white node is optional. Edges drawn with full lines describe routing costs, and the dotted ones denote connection costs.

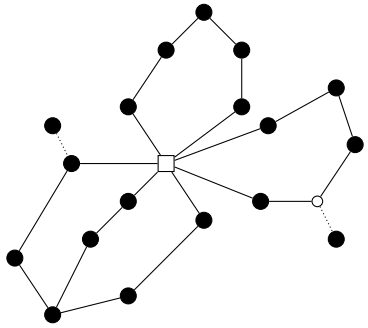

Figure 2: An example of CmTNSSP solution.

2.2. Formal definition

To give a formal definition of CmTNSSP, we establish definitions and conventions which we will work with hereinafter. Network design problems with connectivity requirements can be defined in two ways:

- With respect to the number of edges (links) that may fail in the network without leaving any two terminal nodes disconnected. These requirements translate into edge-disjoint paths between pairs of terminal nodes.
- With respect to the number of nodes that can fail (together with their incident edges) without leaving any two terminal nodes disconnected. These requirements result in node-disjoint paths between pairs of terminal nodes.

The following definitions are taken from [19].

Definition 1. *A pair of nodes* $(i, j) \in V \times V$ *has k-edge-connectivity or is k-edge-connected in G, when at least k edge-disjoint paths (which share no edge) connect i with j.*

This definition is equivalent to stating that any cut in the graph for nodes i, j contains at least k edges.

Definition 2. *We say that a graph* $G = (V, E)$ *is k-edge-connected if, for every pair of nodes* (i, j) *in* V*, this couple is k-edge-connected.*

Analogously, the node-connectivity concepts are defined.

Definition 3. *We say that a pair of nodes (i, j) has k-node-connectivity or is k-node-connected in a given graph, when at least k node-disjoint paths (i.e. they do not share any nodes except i and j) connect i with j.*

Definition 4. *We say that a graph is k-node-connected if every pair of nodes i, j is k-node-connected.*

Readers can note that if two paths with the same endpoints i, j are node-disjoint, then they are also edge-disjoint, but not reciprocally.

Definition 5. *Given a graph $G = (V, E)$ and a vertex $i \in V$, we call degree of i and we noted $\delta(i)$ to the number of incident edges to node i.*

Once specified these definitions, let us now turn to the formal definition of CmTNSSP.

Let $T \subseteq V \setminus \{d\}$ be a set of nodes, which we call terminal nodes of the graph G.
Let $\hat{T} = T \cup \{d\}$ be the set of terminals, including the depot.
Let $W = V \setminus \hat{T}$ be a set of optional (or Steiner nodes) of G.

We want to construct a graph H in such a way that

$$H = H_1 \cup H_2 \cup H_3 \cdots\cdots \cup H_m \tag{1}$$

where each component H_i is defined as

$$H_i = G'_i \cup S_i \quad i = 1, \cdots m \tag{2}$$

and meets

- $G'_i = (U'_i, E'_i) \qquad U'_i \subseteq V, E'_i \subseteq E, \qquad i = 1, \cdots m \qquad$ are 2-node-connected graphs,
- $S_i = (\bar{V}_i \cup \bar{U}_i, \bar{E}_i), \quad \bar{U}_i \subseteq U'_i, \quad \bar{V}_i \subset T, \quad \bar{V}_i \cap U'_i = \phi,$
  $\bar{E}_i = \{(u_i, v_i)\}, \quad u_i \in \bar{U}_i, \quad v_i \in \bar{V}_i, \quad \bar{E}_i \subset E$
  $\delta(v_i) = 1 \quad \forall v_i \in \bar{V}_i \qquad i = 1, \cdots m\,.$

Hereinafter, the set of nodes $v_i \in \bar{V}_i$ will be called pendant nodes, the set of nodes $u_i \in \bar{U}_i$ will be called base of pendant nodes, and the set of edges $\{(u_i, v_i)\} \in \bar{E}_i$ will be called pendant edges. Let $T(H_i)$ be the set of terminal nodes of the *i-th* component of the graph H. Then, there is a capacity constraint such that

$$|T(H_i)| \leq Q \tag{3}$$

For the distinguished node d, the following condition is met

$$d = H_1 \cap H_2 \cap H_3 ..... \cap H_m \tag{4}$$

We also define $C = \{c_{ij}\}_{i,j \in V}$ as the routing costs, i.e. the cost of a certain edge (i, j) which belongs to some G'_k, with $k = 1 \cdots m$. Analogous, let us now define $D = \{d_{ij}\}_{i,j \in V}$ as the connection costs matrix, i.e. the cost of the edge (i, j) when this edge belongs to S_k,

with $k = 1 \cdots m$.

Our goal is to construct a graph H, as defined above, which should be of minimum cost, where the cost includes routing and connection terms.

Proposition 2.1. ***(Complexity)*** *CmTNSSP belongs to class of $\mathcal{NP}$-Hard problems.*

Proof:. Given an undirected graph $G = (V, E)$, the Minimum-Weight Two-Connected Spanning Network [12] is a particular case of CmTNSSP with $m = 1$, $Q = |V|$, $W = \phi$, and $\bar{V}_1 = \phi$. The last condition can be forced by making the elements of the connection costs matrix D enough large. As the Minimum-Weight Spanning Two-Connected Network belongs to the class of $\mathcal{NP}$-Hard problems [12], this demonstrates that CmTNSSP also belongs to the same class.

□

3. INTEGER LINEAR PROGRAMMING MODEL

In this section we propose an integer linear programming model for the CmTNSSP. This model was translated to an algebraic language and solved, as will be shown in Section 5. First, we define the set of adjacent nodes to node $i \in V$ as $Adj(i) = \{j \in V : (i, j) \in E\}$ and the following decisions variables:

$$X_i^k = \begin{cases} 1 & \text{if node } i \in V \text{ belongs to } G'_k \text{ (2-connected structure of sub-network } H_k) \\ 0 & \text{otherwise} \end{cases}$$

$$Y_i^k = \begin{cases} 1 & \text{if node } i \in T \text{ is a pendant node of } G'_k \text{ (2-connected structure of sub-network } H_k) \\ 0 & \text{otherwise} \end{cases}$$

$$Z_{i,j}^k = \begin{cases} 1 & \text{if } i \in T \text{ and } j \in V \text{ are connected by edge } (i, j) \in E, \\ & \text{being } i \text{ a pendant node of } G'_k \text{ (2-connected structure of sub-network } H_k) \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,j}^{u,v,k} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is used in the path from } u \text{ to } v \\ & \text{in the direction from } i \text{ to } j \text{ within component } H_k \\ 0 & \text{otherwise} \end{cases}$$

$$X_{i,j}^k = \begin{cases} 1 & \text{if there is a path between } i \text{ and } j \text{ within component } H_k \\ 0 & \text{otherwise} \end{cases}$$

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

$$w_{i,j} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is a pendant edge used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$z_i = \begin{cases} 1 & \text{if pendant node } i \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

The mathematical programming formulation reads as follows:

$$min \sum_{k=1}^{m} (\sum_{i,j \in V} c_{ij}(x_{ij} - w_{ij}) + \sum_{i,j \in V} d_{ij} w_{ij}) \tag{5}$$

subject to:

$$\sum_{k=1}^{m} X_i^k + Y_i^k = 1 \qquad \forall i \in T \tag{6}$$

$$\sum_{k=1}^{m} X_d^k = m \tag{7}$$

$$\sum_{k=1}^{m} X_i^k \leq 1 \qquad \forall i \in W \tag{8}$$

$$Y_i^k = 0 \qquad \forall i \in W, \quad \forall k \in 1 \cdots m \tag{9}$$

$$Z_{ij}^k \leq x_{ij} \qquad \forall i \in T, \quad \forall j \in V, \quad \forall k \in 1 \cdots m \tag{10}$$

$$Y_i^k = \sum_{j \in Adj(i)} Y_{ij}^k \qquad \forall i \in T, \quad \forall k \in 1 \cdots m \tag{11}$$

$$\sum_{(u,j) \in E} y_{u,j}^{u,v,k} \geq 2X_{u,v}^k - Y_u^k \qquad \forall u, v \in \hat{T}, \quad u \neq v, \quad \forall k \in 1 \cdots m \tag{12}$$

$$\sum_{(i,v) \in E} y_{i,v}^{u,v,k} \geq 2X_{u,v}^k - Y_v^k \qquad \forall u, v \in \hat{T}, \quad v \neq u, \quad \forall k \in 1 \cdots m \tag{13}$$

$$\sum_{(i,p) \in E} y_{i,p}^{u,v,k} - \sum_{(p,i) \in E} y_{p,i}^{u,v,k} \geq 0 \quad \forall u, v \in \hat{T}, \quad \forall p \in V \setminus u, v, \quad \forall k \in 1 \cdots m \tag{14}$$

$$y_{i,j}^{u,v,k} + y_{j,i}^{u,v,k} \leq x_{i,j} \quad \forall u, v \in \hat{T}, \quad u \neq v, \quad \forall (i,j) \in E, \quad \forall k \in 1 \cdots m \tag{15}$$

$$\sum_{i \in T} (X_i^k + Y_i^k) \leq Q \qquad \forall k \in 1 \cdots m \tag{16}$$

$$X_i^k + X_j^k \leq 1 + X_{i,j}^k \qquad \forall i \in V, \quad \forall j \in V, \quad \forall k \in 1 \cdots m \tag{17}$$

$$X_i^k + Y_j^k \leq 1 + X_{i,j}^k \qquad \forall i \in V, \quad \forall j \in T, \quad \forall k \in 1 \cdots m \tag{18}$$

$$Y_i^k + Y_j^k \leq 1 + X_{i,j}^k \qquad \forall i \in T, \quad \forall j \in T, \quad \forall k \in 1 \cdots m \tag{19}$$

$$2X_{i,j}^k \leq X_i^k + X_j^k + Y_i^k + Y_j^k \qquad \forall i \in V, \quad \forall j \in V, \quad \forall k \in 1 \cdots m \tag{20}$$

$$\sum_{k=1}^{m} X_{i,j}^k \leq 1 \qquad \forall i, j \in V \tag{21}$$

$$\sum_{k=1}^{m} Y_i^k \leq z_i \qquad \forall i \in T \tag{22}$$

$$\sum_{j \in Adj(i)} x_{i,j} - 1 \leq M(1 - z_i) \qquad \forall i \in T \quad M \in \mathbb{Z}^+, M \geq max(\delta_i) \;\; i = 1 \cdots |V| \tag{23}$$

$$\sum_{k=1}^{m} Z_{i,j}^k = w_{i,j} \qquad \forall i \in T, \quad j \in Adj(i) \tag{24}$$

$$w_{i,j} \leq x_{i,j} \qquad \forall i \in T, \quad j \in Adj(i) \tag{25}$$

$$Z_{i,j}^k \leq X_j^k \qquad \forall i \in T, \quad \forall j \in Adj(i), \quad \forall k \in 1 \cdots m \tag{26}$$

$$\Big(\sum_{i \in Adj[j]} x_{j,i} - \sum_{i \in Adj[j]} Z_{i,j}^k\Big) \geq 2X_j^k \qquad \forall j \in V \setminus T, \quad \forall k \in 1 \cdots m \tag{27}$$

$$2y_{i,j}^{u,v,k} \leq X_{i,j}^k + X_{u,v}^k \quad \forall u, v \in \hat{T}, \quad \forall i, j \in V, \;\; u \neq v, \quad \forall k \in 1 \cdots m \tag{28}$$

Constraints (6)-(11) impose consistency on individual nodes and edges, while constraints (12)-(15) ensure connectivity between nodes, particularly 2-connectivity on 2-connected structures. Expressions (16)-(23) impose structural consistency, including capacity. Finally, inequalities (24)-(28) are needed for technical issues. This model is of integer linear nature with polynomial number of variables and constraints on the size of the graph. Small sized problem instances can be solved by applying this model, which is done in Section 5.

4. GRASP RESOLUTION

Given the nature of the problem and its complexity, we will address the resolution thereof by the GRASP (Greedy Randomized Adaptive Search Procedures) metaheuristic [5], an iterative process used with success in telecommunications [18]. GRASP comprises two phases: Construction and Local Search. In the first phase, a feasible solution is built by applying greediness (intensification) and randomization (diversification) using a RCL (Restricted Candidate List) to select elements to be added to the solution. In the second phase, this solution is improved by exploring neighbor solutions successively. The solution found by running independently both phases several times is taken as the best solution. A complete detail of generic GRASP characteristics can be read in [16].

4.1. Construction phase

The Construction Phase is the first milestone to produce a feasible solution. In our problem, we need to build m 2-node-connected components having the depot d as the common vertex. During the Construction Phase, components will be iteratively built. We describe below the stages of such phase of GRASP.

Algorithm 1 Selection of m initial nodes

```
1:  procedure Far
2:  input G, C, T, m, n
3:  bestfar ← φ
4:  maxdistance = 0
5:  for i=1 to n do
6:     far ← φ
7:     for i=1 to m do
8:        far[i] ← ExtractRandomNode(T)
9:     end for
10:    distance = 0
11:    for i=1 to m-1 do
12:       for j=i+1 to m do
13:          distance = distance + C_far[i],far[j]
14:       end for
15:       if distance > maxdistance then
16:          bestfar ← far
17:          maxdistance = distance
18:       end if
19:    end for
20: end for
21: return bestfar
```

- **Step 1**. We proceed to locate the first m terminal nodes to be included (one in each component). Algorithm 1 considers m random terminals and computes the sum of distances between them. This procedure is performed n times and the set of m nodes with the maximum sum of distances between them is chosen.

- **Step 2**. For each node of the set selected in Step 1, we consider the k node-disjoint shortest (respect to the routing costs) paths between the node under consideration and the depot, whose total cost is minimal. To obtain these k node-disjoint paths that meet this condition (minimum total cost), we use the algorithm developed by Bhandari [3]. The number of paths k is a parameter of the constructor ($k \geq 2$). From this list of k paths, we choose randomly exactly two paths, and we include them in the solution. This process is repeated m times, once for each set of k node-disjoint paths.

- **Step 3**. We add terminal nodes that are still not part of the solution under construction. Such terminals will be incorporated into each of the components as follows:

  A terminal node which does not belong to the solution under construction is selected randomly, and is connected to the solution generating a path to some of the m components. This operation preserves 2-node-connectivity since adding an independent path between two nodes to a 2-node-connected graph generates a new 2-node-connected graph [6]. We choose the component which connects the node using the criterion of fewer nodes present in this component. This approach is particularly useful for balancing the number of nodes in each of the m components without losing feasibility with respect to the capacity constraint Q. In this process, we try to keep a trade-off of connecting the node to an "inadequate" component as far as costs are concerned.

  To do this, we transform the component by adding a virtual node v' connected to all nodes of such component by zero cost edges, and likewise assigning the value 0 to the edges present in the component to be treated. Then, we define $\bar{C}_{(|V|+1)\times(|V|+1)}$ as the matrix of the transformed component.

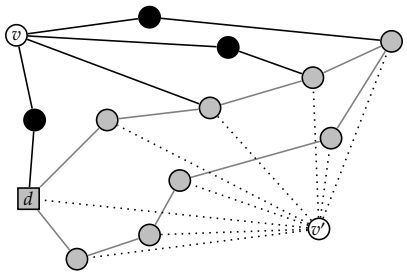

Figure 3: Including node v into a component.

Once we apply the transformation explained above, we proceed to get the k node-disjoint paths with minimum total cost (again using the algorithm of Bhandari) between the terminal node to include v and the virtual node v' (see Figure 3). Among

these k paths, we choose any two randomly, and we incorporate them in the solution under construction.

Algorithm 2 Construction of feasible solution

```
1: procedure Construct_Greedy_Randomize_Feasible_Solution
2: input G, C, ListSize, m, n, Q, T
3: G_Sol ← φ
4: component_nodes ← φ
5: not_assigned ← T
6: FarNodes ← Far(G, C, T, m, n)
7: for i=1 to m do
8:    node = ExtractRandomNode(FarNodes)
9:    minpaths = Bhandari(G, C, not_assigned, ListSize, depot, node)
10:   path_1 ← ExtractRandomPath(minpaths)
11:   path_2 ← ExtractRandomPath(minpaths)
12:   G_Sol ← add_path(G_Sol, path_1)
13:   G_Sol ← add_path(G_Sol, path_2)
14:   component_nodes[i] ← add_nodes(component_nodes[i], path_1)
15:   component_nodes[i] ← add_nodes(component_nodes[i], path_2)
16:   not_assigned ← subtract_nodes(not_assigned, path_1)
17:   not_assigned ← subtract_nodes(not_assigned, path_2)
18: end for
19: repeat
20:   node = ExtractRandomNode(not_assigned)
21:   comp = CompSelect(G_Sol)
22:   Ḡ = transform (G, C, C̄, G_Sol, comp, component_nodes) // Figure 3
23:   minpaths = Bhandari(Ḡ, C̄, not_assigned, ListSize, node, virtual)
24:   path_1 ← ExtractRandomPath(minpaths)
25:   path_2 ← ExtractRandomPath(minpaths)
26:   G_Sol ← add_path(G_Sol, path_1)
27:   G_Sol ← add_path(G_Sol, path_2)
28:   component_nodes[comp] ← add_nodes(component_nodes[comp], path_1)
29:   component_nodes[comp] ← add_nodes(component_nodes[comp], path_2)
30:   not_assigned ← subtract_nodes(not_assigned, path_1)
31:   not_assigned ← subtract_nodes(not_assigned, path_2)
32: until not_assigned = φ
33: return G_Sol
```

Algorithm 2, that describes the three steps that comprise the construction phase of GRASP, stops when all terminal nodes are included in some component using the procedure described above.

We remark that in the construction phase, the algorithm tries to build non-cyclical components using, if it improves costs, Steiner nodes. The pendant nodes are not considered at this stage, they appear in the solution when the local search is performed.

4.2. Local Search Phase

Once we build a feasible solution to the CmTNSSP, it must be improved to approach the global optimal solution. To do this, we use a combination of classical local searches and those based on exact integer linear programming models. There are different strategies for combining a process of building a feasible solution and a set of local searches. In this paper, for deploying local searches we use a variant of VNS (Variable Neighborhood Search) called VND (Variable Neighborhood Descent), whose generic algorithm is detailed in [11].

We have designed five neighborhoods corresponding to the five local searches that we develop below. These local searches are referred to as Extract Insert Nodes **(Extract-Insert)**, Swapping Nodes **(Swapping)**, Components Crossing **(Crossing)**, Best Path with Rays **(Best PWR)** and Best 2-Node-Connected Component **(Best 2NC)**, which are applied successively in this order.

4.2.1. Extract-Insert Nodes

This local search performs the extraction of all terminal nodes in a random order from their current positions in the solution, and relocate them to other positions (either in the same component or other) to improve the overall cost without losing feasibility. The extraction procedure is simple: A terminal node is extracted and the nodes adjacent to the extracted node are reconnected. To make the insertion of the extracted node, we consider the following definition:
Let $i \in T$ be a terminal node extracted and a neighborhood N defined as follows:

$$N(i) = \left\{ j \in T : j \begin{array}{l} \text{are the } k \text{ nodes } \textit{closer} \text{ to node } i \text{ taking into account routing} \\ \text{costs } c_{ij} \text{ defined in original graph } G \end{array} \right\} \quad (29)$$

The loop for each terminal node i, ends after having considered all possible insertions between k closest nodes, and selects the movement that produces the lowest total cost. The algorithm repeats the same procedure for all $i \in T$ not even considered, by examining $N(i)$ until finally selecting the movement that produces the lowest total cost.

4.2.2. Swapping Nodes

This local search selects two nodes and makes an exchange (swapping) between them. This process starts with a random selection of a terminal not pendant node and tests all possible ways to swap this node with another *close* node belonging to a 2-node-connected component (the same or other). To clarify the concept *close*, we define a neighborhood related to the considered node.

Again, we will appeal to the same definition of neighborhood that we use in the extract-insert local search, (detailed in 4.2.1), i.e. the neighborhood N of k nodes $j \in T$ closest to the node i. The algorithm begins by taking a random node i and considers the node j as the nearest node to i. If j is a pendant node, it does not perform any movement

and continues with the next node, i.e. takes a next j closest to i. Each time a swapping movement leads to improvement and keeps the feasibility, the current solution is updated, the possible swapping with other nodes j in descending order of distance are discarded and finally, the algorithm continues with the next non pendant terminal node i.

4.2.3. Crossing components

This local search (Algorithm 3) takes two *close* nodes (as defined in Section 4.2.2), each one in a different component, eliminates one of their adjacent edges (for each node) and connects each pair of nodes (in different component) by the edge that generates the best cost.

Algorithm 3 Crossing Components.

input G_{inic}, T, k
$G_{best} \leftarrow G_{inic}$
for ($i = 1$ **to** $|T|$) **do**
  if (i is not a pendant node) **then**
    Let K be the ordered set of k nodes closest to node i
    for ($u = 1$ **to** k) **do**
      Let $j = u^{th}$ node closest to node i
      remove an edge adjacent to node i
      remove an edge adjacent to node j
      Let i' be the opposite end of the edge incident to i
      Let j' be the opposite end of the edge incident to a j
      state_1=generate edges (i, j') and (i', j)
      state_2=generate edges (i, j) and (i', j')
      select the state that generates feasible solution with improved resulting cost
      $improve = update(G_{best})$
      if ($improve$) **then**
        breakfor
        {exit FOR loop, we do not consider next closer nodes}
      end if
    end for
  end if
end for
return G_{best}

4.2.4. Best path with pendants

This local search is based on an integer linear programming model. First we give a definition of structures used for this local search, that we call **path with pendant nodes** or, shortly, **path with pendants**.

Definition 6. ***Path with pendant nodes***. *Given an undirected graph $G = (V, E)$, we define a path with pendant nodes and endpoints a and $z \in V$ as the path (if exists) $p(a, z) \subseteq G$ that connects nodes a and z (that we call main path), and the following conditions are met:*

- *G is acyclic and connected.*
- *All nodes that do not belong to p are connected to some node of p through a simple edge.*

Given a feasible solution to the CmTNSSP, we should identify all simple cycles that exist in each component and we should explode them in paths, adding their pendants nodes. For each path with pendants, exact local search is applied to obtain the best solution with such topology. This algorithm is based on an integer linear programming model, it takes an input graph with two distinguished nodes a and z and returns the best path with pendants with the same endpoints a and z as optimal solution.

We consider the following definitions:

Let a and z be two distinguished terminal nodes such that $a \in \hat{T}$ and $z \in \hat{T}$.
Let $T = \hat{T} \setminus (\{a\} \cup \{z\})$ be the set of terminal nodes without a and z.

Let us now define the model variables specific to this local search. Note that some of them exhibit a similar meaning with respect to the formulation of the CmTNSSP.

$$X_i = \begin{cases} 1 & \text{if node } i \in \hat{T} \text{ belongs to main path} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_i = \begin{cases} 1 & \text{if node } i \in T \text{ is a pendant node} \\ 0 & \text{otherwise} \end{cases}$$

$$Z_{i,j} = \begin{cases} 1 & \text{if } i \in \hat{T} \text{ and } j \in V \text{ are connected, being } i \text{ a pendant node and } j \text{ a main path node} \\ 0 & \text{otherwise} \end{cases}$$

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

$$w_{i,j} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is a pendant edge and is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,j}^{u,v} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is used in path that goes from node } u \text{ to node } v \\ 0 & \text{otherwise} \end{cases}$$

The integer linear programming model is defined as follows:

$$min(\sum_{i,j \in V} c_{ij}(x_{ij} - w_{ij}) + \sum_{i,j \in V} d_{ij} w_{ij}) \tag{30}$$

subject to:

$$X_i + Y_i = 1 \qquad \forall i \in T \tag{31}$$

$$X_i = 1 \qquad \forall i \in (\{a\} \cup \{z\}) \tag{32}$$

$$Z_{ij} \leq X_j \qquad \forall i \in T \quad \forall j \in Adj(i) \tag{33}$$

$$Y_i = \sum_{j \in Adj(i)} Z_{ij} \qquad \forall i \in T \tag{34}$$

$$\sum_{j \in V} w_{i,j} \leq Y_i \qquad \forall i \in T \tag{35}$$

$$Z_{i,j} = w_{i,j} \qquad \forall i \in T \quad j \in Adj(i) \tag{36}$$

$$\sum_{j \in Adj(i)} x_{i,j} \leq M(1 - Y_i) + 1 \quad \forall i \in T \quad M \in \mathbb{Z}^+, M \geq max(\delta_i) \;\; i = 1 \cdots |V| \tag{37}$$

$$w_{i,j} \leq x_{i,j} \qquad \forall i \in T \quad j \in Adj(i) \tag{38}$$

$$\sum_{j \in Adj[u]} y^{u,v}_{u,j} = 1 \qquad \forall u, v \in \hat{T}, u \neq v, \tag{39}$$

$$\sum_{i \in Adj[v]} y^{u,v}_{i,v} = 1 \qquad \forall u, v \in \hat{T}, v \neq u, \tag{40}$$

$$\sum_{i \in Adj[p]} y^{u,v}_{i,p} - \sum_{i \in Adj[p]} y^{u,v}_{p,i} \geq 0 \quad \forall u, v \in \hat{T}, \quad \forall p \in V \setminus u, v \tag{41}$$

$$y^{u,v}_{i,j} + y^{u,v}_{j,i} \leq x_{i,j} \quad \forall u, v \in \hat{T}, u \neq v, \quad \forall (i,j) \in E \tag{42}$$

$$Y_i = 0 \qquad \forall i \in W \tag{43}$$

$$\sum_{j \in Adj(i)} Z_{i,j} = 0 \qquad \forall i \in W \tag{44}$$

$$(\sum_{i \in Adj[j]} Z_{i,j} + 2X_j - \sum_{i \in Adj[j]} x_{j,i} = 0) \qquad \forall j \in W \tag{45}$$

$$\sum_{i \in Adj[j]} (Z_{i,j} + Z_{j,i}) + 2X_j - \sum_{i \in Adj[j]} x_{j,i} = 0 \qquad \forall j \in T \tag{46}$$

$$\sum_{i \in Adj[j]} (Z_{i,j}) + X_j - \sum_{i \in Adj[j]} x_{j,i} = 0 \qquad \forall j \in (\{a\} \cup \{z\}) \tag{47}$$

Algorithm 4 Best path with pendant nodes.

```
1: input G_sol, G, C, D, T, MAX_PATH_LENGTH
2: G_best ← G_sol
3: q_cycles = cycles_count(G_sol) {Numbers of cycles of G_sol}
4: all_cycles ← cycles(G_sol) {Array with cycles of G_sol}
5: for (i = 1 to q_cycles) do
6:    path_long = min (length(all_cycles(i)), MAX_PATH_LENGTH ))
7:    begin_path = 1
8:    end_path =length(all_cycles(i))
9:    while (end_path ≤ length(all_cycles(i))) do
10:      end_path = begin_path + 3 + (rand() MOD (path_long - 2))
11:      P = path_with_rays(G_sol, all_cycles(i), begin_path, (end_path MOD
         length(all_cycles(i))
12:      H ← induced_graph_path(P, G, T)
13:      P_best = best_pwr(G_sol, G P C, D, H)
14:      G_best ← G_best - P + P_best
15:      begin_path = end_path
16:   end while
17: end for
18: return G_best
```

Algorithm 4 describes the local search which involves the replacement of a path with pendants by another path with the same nodes and endpoints whose total cost is lower (optimal). It begins by taking as input the graph G_{Sol}, feasible solution of CmTNSSP. For each m components of G_{Sol} we count its cycles, which are then identified and stored in the indexed list *all_cycles* (Lines 3 and 4). Next, each of the cycles identified in the previous steps are treated, running the operations defined in the scope of **for** (Lines 5 to 17) until examining all cycles. Each cycle is divided into a certain number of paths of variable length (*MAX_PATH_LENGTH* parameter). We set a start node and end node of the first path in the cycle (Lines 7 and 8).

Once initialized the path to process, we enter into a repetitive loop determined by the scope of (**while**) (Lines 9 to 16), which readjust the path length in a random way (Line 10). Each path obtained in the previous step is added with pendant nodes present in G_{Sol} (Line 11) obtaining a path with endpoints *begin_path*, *end_path* and pendant nodes, such we specify in Definition 6. In the next step, we generate the graph H induced by nodes

of the path with pendants P respect to the original graph G. (Line 12). The graph H thus generated is taken as input to process **best_pwr**, that gives us the best path with pendants and endpoints *begin_path*, *end_path* (Line 13). In line 14, we perform the substitution of the path with pendants P by the path with pendants P_{best}, obtaining a better solution G_{best}. Next, we reset the start and the end node in the cycle we are processing (Line 15) to generate a new path. After processing all paths within each cycle, we return the best cost solution G_{best} (Line 18).

4.2.5. Best 2-Connected Component

This local search is also based on integer linear programming. Just as in the previous local search, given a feasible solution to the problem, Algorithm 5 identifies all cycles that exist in each component. For each cycle we will now apply an exact algorithm getting the best replacement solution that changes a cycle by a 2-node-connected topology.

As we saw in Section 1, the best 2-node-connected solution covering a certain set of nodes is not necessarily a cycle, so this local search may include such topologies in our solution (see Figure 1). This algorithm takes as input the induced sub-graph of the original graph with nodes of the cycle and some Steiner nodes, and returns the best 2-connected sub-graph, i.e. it can potentially change a cycle for a structure that contains a Monma's graph, if such change improves solution costs.

To model this local search, we use a particular case of **GSP** (General Steiner Problem), where connectivity of all its terminal nodes is two. The model only considers the routing cost matrix because in this local search pending nodes generated so far, are not considered.

Let us define the model variables as follows:

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

$$y_{i,j}^{u,v} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used in a path from node } u \text{ to } v \\ 0 & \text{otherwise} \end{cases}$$

The integer linear programming model is defined as follows:

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

subject to:

$$\sum_{j \in Adj[u]} y_{u,j}^{u,v} = 2 \qquad \forall u, v \in \hat{T}, u \neq v,$$

$$\sum_{i \in Adj[v]} y_{i,v}^{u,v} = 2 \qquad \forall u, v \in \hat{T}, v \neq u,$$

$$\sum_{i \in Adj[p]} y_{i,p}^{u,v} - \sum_{i \in Adj[p]} y_{p,i}^{u,v} \geq 0 \quad \forall u, v \in \hat{T}, \quad \forall p \in V \setminus u, v$$

$$y_{i,j}^{u,v} + y_{j,i}^{u,v} \leq x_{i,j} \quad \forall u, v \in \hat{T}, u \neq v, \quad \forall (i,j) \in E$$

Algorithm 5 Best 2-node-connected component.

```
1: input G, G_sol, C, T
2: G_best ← G_sol
3: q_cycles = cycles_count(G_sol) {Number of cycles of G_sol}
4: all_cycles ← cycles(G_sol) {Array with cycles of G_sol}
5: for (i = 1 to q_cycles) do
6:     best = best_2nc(G_sol, G_orig, all_cycles(i))
7:     G_best ← G_best - all_cycles(i) + best_2nc
8: end for
9: return G_best
```

Analogous to Algorithm 4, Algorithm 5 counts and identifies the cycles present in G_{sol} (lines 3 and 4). For each of these cycles, the process **best_2nc** (line 6) returns the best 2-node-connected structure and performs substitution of a cycle by the best one (line 7).

5. COMPUTATIONAL RESULTS

To the best of our knowledge, exact resolution of the CmTNSSP does not exist in the literature, therefore, in principle we do not have a reference to compare the effectiveness of the metaheuristic developed in this work. Considering that the CmTNSSP is a relaxation of CmRSP and that any solution of CmRSP is also solution of CmTNSSP, we refer to the work on the CmRSP in [1]. In that paper, the vast majority of the problem instances used are solved to optimality and those that are unresolved have lower bounds that will guide us to measure the results generated by our application. Also, we compare against more recent results for CmRSP provided in [14].

The exact ILP model has been implemented in AMPL. The heuristic was coded in C, using the callable library of CPLEX. Our hardware platform consists of a computer with Intel I7 processor with 8 Gb. RAM and OS Fedora Core 20.

5.1. Exact resolution

The model has been implemented and executed on several small instances and we have selected one of them to show the results. We have defined a graph called *nut*30 and denoted $N = (V, E)$ with $V = T \cup W \cup \{d\}$, where: $T = \{1 \cdots 19\}$ is the set of terminal nodes of graph N, $W = \{20 \cdots 29\}$ is the set of Steiner nodes, and $d = \{0\}$ is the depot

node. The capacity is set as $Q = 2$ and the number of components as $m = 2$. The routing cost matrix C and the connection costs matrix D are identical, where their values are the euclidean distances between vertices of the graph N.

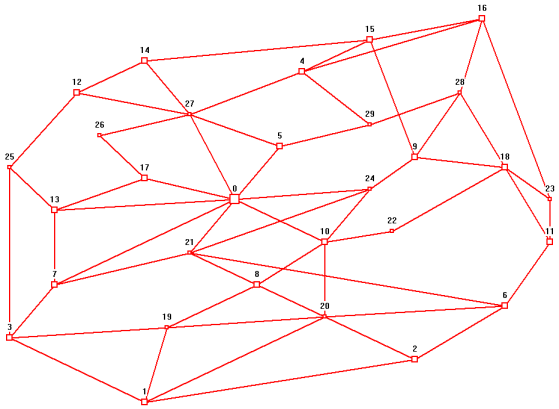

Figure 4: Initial graph (*nut*30) for testing ILP model of CmTNSSP.

In order to shorten the computational processing used in executing the solver CPLEX, we have not considered the complete graph, instead, we have generated only some edges of the graph N. Hence, the set E contains only the edges that can be seen in Figure 4. Still, given the complexity of the model, the transformation to an integer linear programming for this instance had 721,244 rows, 618,913 columns, and 629,149 non-zero values.

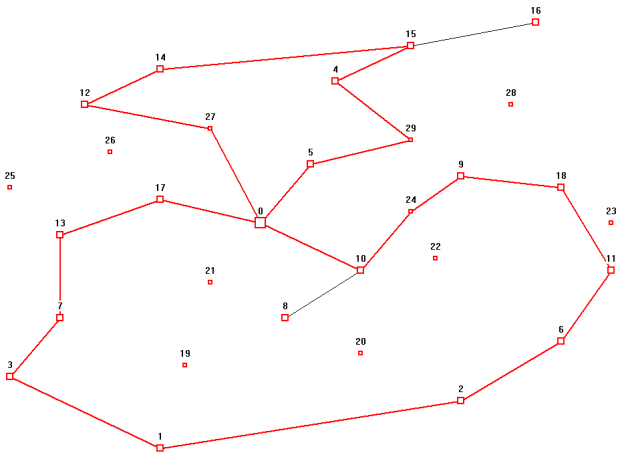

Figure 5: Global optimum of CmTNSSP for *nut*30, found using CPLEX solver.

After running the model, we obtain the exact solution of CmTNSSP for the instance defined above. We can observe its graphical representation in Figure 5. Note that even though we are solving the CmTNSSP, the optimal solution is also a solution of CmRSP, i.e. the connected components are exclusively cycles.

5.1.1. Resolution by GRASP

We use the test instances proposed by [1], which are divided into two classes, A and B. In class A, both routing and connection costs match. In class B, routing costs are greater than connection costs. For both classes of instances the graphs used are the same, the only difference is in the cost of the edges according to whether or not they are incident to a pendant node. These graphs are *eil*51, *eil*76 and *eil*101, obtained from the TSPLIB, the Traveling Salesman Problem Library [15]. Additionally, a new graph called *eil*26 is added and it is built with the first 26 vertices of *eil*51. Then, we set $n = \{26, 51, 76, 101\}$ as the number of vertices for each of the graphs defined in the previous paragraph. The first node of each of these graphs is tagged as depot. The remaining 25, 50, 75, and 100 nodes respectively, are divided into terminal and optional nodes according to a parameter $\alpha \in \{0.25, 0.5, 0.75, 1\}$, where U (set of terminal nodes) contains the first $\alpha(n-1)$ nodes and W (set of Steiner nodes) contains the remaining ones. For each of these combinations we generate instances with $m \in \{3, 4, 5\}$, and Q will be calculated for a percentage use of the components above the 90 % using the following formula:

$$Q = \left\lceil \frac{|U|}{0.9m} \right\rceil \tag{48}$$

The costs of instances from classes A and B are defined in the following way:

- **Class A**. Routing and connection costs are equal and correspond to the Euclidean distance $e_{i,j}$ between nodes (i, j). Thus $c_{i,j} = d_{i,j} = e_{i,j}$
- **Class B**. Routing costs $c_{i,j} = \lceil \beta\, e_{i,j} \rceil$, where β is an integer in the range [6,9]. Connection costs are $d_{i,j} = \lceil (10 - \beta) e_{i,j} \rceil$. For our Class B instances, we use $\beta = 7$.

In addition to the definitions specified in the preceding paragraphs, there is another constraint on connection costs. Each edge connecting nodes on a 2-node-connected component with a pendant node, cannot have a higher cost than a given bound:

$$d_{max} = 0.2 \times \frac{\sum_{(i,j)\in E} d_{ij}}{|E|} \tag{49}$$

This is in fact an additional problem constraint, which is also present in the studies used as reference for comparison in this work.

We can see in Table 1 the results of the solutions for Class A instances. The notations corresponding to each column are the following:
$|T|$ is the number of terminal nodes in the specified instance, CN is the number of nodes

INSTANCE	$\|T\|$	Q	*CN*	*PN*	*SN*	Z_{best}	$\bar{Z}_1$	$\bar{Z}_2$	*gap %*	*t(s)*
A01-n026-m03	12	5	12	0	1	242	242	242	0,000	1.61
A02-n026-m04	12	4	12	0	1	261	261	261	0,000	0.97
A03-n026-m05	12	3	12	0	1	292	292	292	0,000	13.77
A03-n026-m05	12	3	12	0	0	292	292	292	0,000	4.54
A04-n026-m03	18	7	18	0	0	301	301	301	0,000	34.29
A05-n026-m04	18	5	18	0	0	339	339	339	0,000	62.58
A05-n026-m04	18	5	18	0	1	339	339	339	0,000	9.34
A06-n026-m05	18	4	18	0	0	375	375	375	0,000	2.67
A07-n026-m03	25	10	24	1	0	325	325	325	0,000	14.06
A08-n026-m04	25	7	25	0	0	362	362	362	0,000	3.99
A10-n051-m03	12	5	12	0	0	242	242	242	0,000	20.09
A11-n051-m04	12	4	12	0	3	261	261	261	0,000	6.42
A12-n051-m05	12	3	11	1	2	286	286	286	0,000	37.69
A13-n051-m03	25	10	22	3	3	322	322	322	0,000	130.85
A14-n051-m04	25	7	24	1	1	360	360	360	0,000	49.75
A15-n051-m05	25	6	23	2	2	379	379	379	0,000	117.67
A16-n051-m03	37	14	33	4	1	373	373	373	0,000	296.60
A17-n051-m04	37	11	33	4	1	405	405	405	0,000	80.49
A18-n051-m05	37	9	33	4	1	432	432	432	0,000	2720.60
A19-n051-m03	50	19	45	5	0	458	458	458	0,000	1674.86
A20-n051-m04	50	14	48	2	0	490	490	490	0,000	3429.11
A21-n051-m05	50	12	43	7	0	520	520	520	0,000	6338.64
A22-n076-m03	18	7	17	1	5	330	330	330	0,000	36.13
A23-n076-m04	18	5	15	3	7	385	385	385	0,000	112.97
A24-n076-m05	18	4	17	1	4	448	448	448	0,000	109.91
A25-n076-m03	37	14	35	2	2	403	402	402	0,249	3624.35
A26-n076-m04	**37**	**11**	**36**	**1**	**3**	**456**	**460**	**457**	**-0,870**	**7200.00**
A27-n076-m05	37	9	36	1	4	483	479	479	0,835	7200.00
A28-n076-m03	56	21	48	8	1	474	471	471	0,637	7200.00
A29-n076-m04	**56**	**16**	**49**	**7**	**1**	**519**	**523**	**519**	**-0,765**	**7200,00**
A30-n076-m05	56	13	50	6	2	547	545	545	0,367	7200.00
A31-n076-m03	75	28	71	4	0	571	564	564	1,241	7200.00
A32-n076-m04	75	21	73	2	0	617	606	602	1,815	7200.00
A33-n076-m05	**75**	**17**	**68**	**7**	**0**	**651**	**654**	**640**	**-0,459**	**7200.00**
A34-n101-m03	25	10	21	4	7	363	363	363	0,000	199.27
A35-n101-m04	25	7	21	4	9	415	415	415	0,000	1023.84
A36-n101-m05	25	6	22	3	9	448	448	448	0,000	1264.62
A37-n101-m03	50	19	46	4	8	500	500	500	0,000	4020.65
A38-n101-m04	50	14	47	3	6	538	532	528	1,128	7200.00
A39-n101-m05	50	12	46	4	5	573	568	567	0,880	7200.00
A40-n101-m03	75	28	69	6	5	613	595	595	3,025	7200.00
A41-n101-m04	75	21	73	2	1	651	625	623	4,160	7200.00
A42-n101-m04	75	17	70	5	2	677	662	657	2,266	7200.00
A43-n101-m03	100	38	84	16	0	662	646	646	2,477	7200.00
A44-n101-m04	100	28	87	13	0	680	680	679	0,000	7200.00
A45-n101-m05	100	23	84	16	0	713	700	700	1,857	7200.00

Table 1: Best values found for instances Class A.

present in 2-node-connected structures, PN is the number of pendant nodes in the solution, SN is the number of Steiner nodes used in the solution, Z_{best} is the objective value found by GRASP, $\bar{Z}_1$ is the reference objective value obtained in [1], $\bar{Z}_2$ is the best value obtained in a recent work [14], and gap is the percentage difference of $\bar{Z}_1$ with respect to our solution, which is calculated as follows:

$$gap = \frac{Z_{best} - \bar{Z}_1}{\bar{Z}}$$

Finally, column $t(s)$ points the execution time of the instance in seconds. We have defined a limit of 7200 seconds of maximum runtime.

Table 1 reports the best Z_{best} found for CmTNSSP. Values in bold are those where the proposed GRASP based heuristic improves the solution found by the original work of [1]. Note that some of those values where later improved by [14]. In general terms, we can conclude that our proposed algorithm is successful in solving the CmRSP, a problem closely related to CmTNSSP. Also, some improvements in specific instances were found.

INSTANCE	$\|T\|$	Q	*CN*	*PN*	*SN*	Z_{best}	$\bar{Z}_1$	$\bar{Z}_2$	*gap %*	*t(s)*
B01-n026-m03	12	5	11	1	1	1684	1684	1684	0,000	3.09
B02-n026-m04	12	4	12	0	1	1827	1827	1827	0,000	1.09
B03-n026-m05	12	3	11	1	2	2041	2041	2041	0,000	10.68
B04-n026-m03	18	7	17	1	1	2104	2104	2104	0,000	24.90
B05-n026-m04	18	5	17	1	1	2370	2370	2370	0,000	78.21
B06-n026-m05	18	4	17	1	2	2615	2615	2615	0,000	47.01
B07-n026-m03	25	10	24	1	0	2251	2251	2251	0,000	35.13
B08-n026-m04	25	7	24	1	0	2510	2510	2510	0,000	51.65
B09-n026-m05	25	6	25	0	0	2674	2674	2674	0,000	150.31
B10-n051-m03	12	5	10	2	2	1681	1681	1681	0,000	2035.19
B11-n051-m04	12	4	10	2	3	1821	1821	1821	0,000	49.26
B12-n051-m05	12	3	10	2	2	1975	1972	1972	0,152	930.42
B13-n051-m03	25	10	21	4	3	2176	2176	2176	0,000	1724.28
B14-n051-m04	25	7	22	3	3	2470	2470	2470	0,000	626.97
B15-n051-m05	25	6	21	4	4	2579	2579	2579	0,000	92.66
B16-n051-m03	37	14	29	8	2	2490	2490	2490	0,000	3699.45
B17-n051-m04	37	11	29	8	2	2735	2721	2721	0,515	3605.47
B18-n051-m05	37	9	32	5	2	2908	2908	2908	0,000	197.51
B19-n051-m03	50	19	39	11	0	3015	3015	3015	0,000	871.33
B20-n051-m04	50	14	39	11	0	3267	3260	3260	0,215	7200,00
B21-n051-m05	50	12	38	12	0	3404	3404	3404	0,000	3773.22
B22-n076-m03	18	7	15	3	4	2253	2253	2253	0,000	186.10
B23-n076-m04	18	5	13	5	8	2620	2620	2620	0,000	90.78
B24-n076-m05	18	4	15	3	9	3155	3059	3059	3,138	7200,00
B25-n076-m03	37	14	32	5	6	2731	2720	2720	0,404	7200,00
B26-n076-m04	**37**	**11**	**34**	**3**	**4**	**3134**	**3138**	**3100**	**-0,127**	**7200.00**
B27-n076-m05	37	9	36	1	3	3329	3311	3284	0,544	7217.19
B28-n076-m03	**56**	**21**	**40**	**16**	**4**	**3044**	**3088**	**3044**	**-1,425**	**7200.00**
B29-n076-m04	**56**	**16**	**44**	**12**	**2**	**3439**	**3447**	**3415**	**-0,232**	**7200.00**
B30-n076-m05	**56**	**13**	**44**	**12**	**2**	**3635**	**3648**	**3632**	**-0,356**	**3797.03**
B31-n076-m03	**75**	**28**	**55**	**20**	**0**	**3724**	**3740**	**3652**	**-0,428**	**2112.23**
B32-n076-m04	75	21	57	18	0	4096	4026	3964	1,739	7200,00
B33-n076-m05	75	17	58	17	0	4489	4288	4217	4,688	7200,00
B34-n101-m04	25	7	19	6	9	2445	2434	2434	0,369	7200,00
B35-n101-m04	25	7	19	6	6	2795	2782	2782	0,467	7200,00
B36-n101-m05	25	6	18	7	4	3009	3009	3009	0,000	597.71
B37-n101-m03	**50**	**19**	**40**	**10**	**8**	**3331**	**3332**	**3322**	**-0,030**	**7200,00**
B38-n101-m04	50	14	38	12	8	3560	3533	3533	0,764	7200,00
B39-n101-m05	50	12	41	9	8	3873	3872	3834	0,026	7200,00
B40-n101-m03	75	28	68	7	5	3931	3923	3887	0,204	7200,00
B41-n101-m04	75	21	68	7	6	4332	4125	4082	5,018	7200,00
B42-n101-m05	75	17	69	6	6	4494	4458	4358	0,808	7200,00
B43-n101-m03	100	38	96	4	0	4403	4110	4110	7,129	7200,00
B44-n101-m04	100	28	95	5	0	4526	4506	4355	0,444	7200,00
B45-n101-m05	100	23	96	4	0	4639	4632	4565	0,151	7200,00

Table 2: Best values found for instances Class B

Similarly, in Table 2 we can see the best objective values generated by our algorithm for Class B instances. We can observe even more improvements with respect to the original work of [1] and similar relationship with results of [14]. The same conclusions already stated for Class A, also hold for Class B instances. Other results about this work and more detailed procedures with other instances can be read in [2].

It is worth mentioning that, due to lack of references for comparison, we are comparing against results produced by algorithms which were not conceived to solve the problem introduced in this work. Nevertheless, our results are competitive when compared with the ones produced by the authors who introduced the CmRSP. The comparison against more recent results gives less chances to succeed in terms of improvements on CmRSP instances, since newer heuristic solving methods are very much specialized. Actually, the best known results for the CmRSP have been published very recently in [20], a work which is contemporary with this one.

5.2. CmTNSSP with non cyclical 2-node-connected components

In the results displayed in Tables 1 and 2, despite the local search applied which induces the use of non-cyclical 2-node-connected components if these are optimal (see Section 4.2.5), we didn't find such structures for the tested instances. To verify that the proposed algorithm finds such solutions, we generate an additional test case based on a graph comprising 36 nodes, which are distributed in the following way:

$$d = \{0\}, \quad T = \{1 \cdots 27\}, \quad W = \{28 \cdots 35\}$$

The set of vertices V are located on a planar coordinate system (x, y) with the following values:

0 (11,9)	6 (5,9)	12 (14,12)	18 (20,6)	24 (25,9)	30 (3,10)
1 (9,13)	7 (3,7)	13 (14,6)	19 (21,17)	25 (28,12)	31 (16,5)
2 (7,11)	8 (8,8)	14 (16,9)	20 (21,12)	26 (28,6)	32 (21,10)
3 (6,13)	9 (7,6)	15 (18,12)	21 (22,9)	27 (30,9)	33 (22,14)
4 (3,12)	10 (4,4)	16 (19,9)	22 (24,6)	28 (7,9)	34 (25,5)
5 (1,9)	11 (13,15)	17 (17,6)	23 (25,12)	29 (9,4)	35 (28,9)

Cost matrices $C = \{c_{ij}\}_{i,j \in V}$ and $D = \{d_{ij}\}_{i,j \in V}$ are both defined by Euclidian distances between vertices i, j multiplied by a factor 10, except for a set of edges $E' \subseteq E$ to which the following costs are assigned:

$c_{0,11} = c_{11,0} = d_{0,11} = d_{11,0} = 1$
$c_{12,15} = c_{15,12} = d_{12,15} = d_{15,12} = 5$
$c_{0,14} = c_{14,0} = d_{0,14} = d_{14,0} = 1$
$c_{16,14} = c_{14,16} = d_{16,14} = d_{14,16} = 1$
$c_{0,13} = c_{13,0} = d_{0,12} = d_{12,0} = 1$
$c_{0,13} = c_{13,0} = d_{0,13} = d_{13,0} = 1$
$c_{0=15,20} = c_{20,15} = d_{15,20} = d_{20,15} = 1$

$c_{22,26} = c_{26,22} = d_{22,26} = d_{26,22} = 1$
$c_{20,23} = c_{23,20} = d_{20,23} = d_{23,20} = 1$
$c_{18,22} = c_{22,18} = d_{18,22} = d_{22,18} = 1$
$c_{14,17} = c_{17,14} = d_{14,17} = d_{17,14} = 80$
$c_{18,17} = c_{17,18} = d_{18,17} = d_{17,18} = 1$
$c_{24,27} = c_{27,24} = d_{24,27} = d_{27,24} = 5$

The constructor parameters are the following:

$m = 2$; $Q = 18$; $ListSize = 4$; $k = 7$; $p = 11$; $MAX\_PATH\_LENGTH = 4$

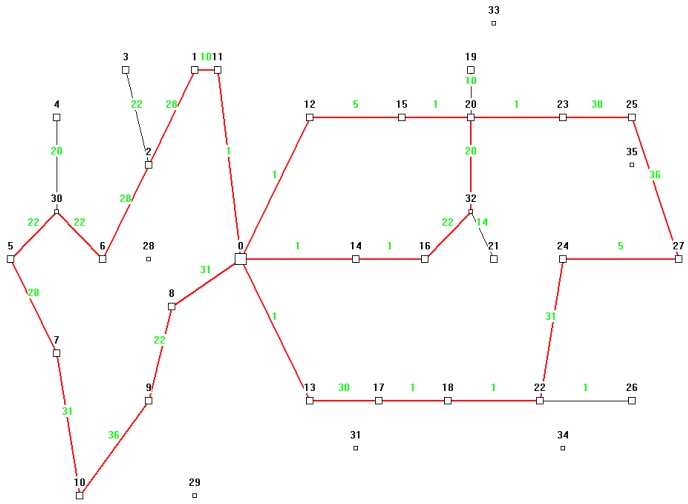

Figure 6: Topology of non-cyclical 2-node-connected component found.

For the values specified above, the GRASP-VND algorithm found an optimal (local to our knowledge) feasible solution with a non-cyclical structure in one of its components (Figure 6). These results show that the designed GRASP-VND metaheuristic is able to obtain the best solution (local optimum) with non-cyclical structures.

6. CONCLUSIONS AND FUTURE WORKS

The Capacitated m Two-Node Survivable Star Problem (CmTNSSP) has been introduced. As far as we know, it has not been studied in prior literature. The need for redundancy and cheaper costs in network deployment is remarkable. Inspired by theoretical results and the related problem CmRSP, we propose an alternative problem where rings are replaced by arbitrary two-node connected components. Both problems are computationally intractable. Therefore, heuristics are suitable for large case scenarios. The CmTNSSP has been modeled by an ILP formulation and heuristically addressed following a GRASP metaheuristic enriched with a Variable Neighborhood Descent (VND) and exact local searches. Numerical results validated both the exact formulation and the heuristic approach. Results from the literature concerning CmRSP were taken as reference for comparison. In all cases, the components obtained were cycles instead of other two-connected topologies. We found that a particular cost structure lead to non-cyclical solutions. Further research is needed in order to understand the nature of problem instances which influence these results. In this paper we have seen that the CmTNSSP as

a slight variation of CmRSP. However, delay-sensitive applications can increase the relevance of CmTNSSP with respect to CmRSP. To achieve this goal, diameter constraints should be introduced to ensure connectivity of any pair of nodes by a limited number of hops. Obviously, there will be a trade-off when this constraint is added to the problem. Two-node-connected components (not purely cycles) can meet this objective from a topological point of view. Adding diameter constraints become CmTNSSP in a more sophisticated problem, covering other network requirements such as quality of service (QoS). Authors are actually researching this line of work. As a future work, we also wish to apply these techniques to the design of real-life networks.

References

[1] Baldacci, R., Dell'Amico, M., and González, J. J. S. "The capacitated m-ring-star problem." *Operations Research*, 55(6) (2007) 1147–1162.

[2] Bayá, G. *Diseño Topológico de Redes. Caso de Estudio: Capacitated* m *Two-Node Survivable Star Problem*. Master's thesis, Universidad de la República. Pedeciba Informática, Montevideo, Uruguay (2014).

[3] Bhandari, R. "Optimal physical diversity algorithms and survivable networks." In "Computers and Communications, 1997. Proceedings., Second IEEE Symposium on," 433–441 (1997).

[4] Dantzig, G., Fulkerson, R., and Johnson, S. "Solution of a large-scale traveling-salesman problem." *Journal of the operations research society of America*, 2(4) (1954) 393–410.

[5] Feo, T. A. and Resende, M. "Greedy randomized adaptive search procedures." *Journal of Global Optimization*, 6(2) (1995) 109–133.

[6] Frederickson, G. N. and Ja'Ja', J. "Approximation algorithms for several graph augmentation problems." *SIAM Journal on Computing*, 10(2) (1981) 270–283.

[7] Hoshino, E. A. and De Souza, C. C. "A branch-and-cut-and-price approach for the capacitated m-ring-star problem." *Discrete Appl. Math.*, 160(18) (2012) 2728–2741.

[8] Labbé, M., Laporte, G., Martín, I. R., and González, J. J. S. "The ring star problem: Polyhedral analysis and exact algorithm." *Networks*, 43(3) (2004) 177–189.

[9] Labbé, M., Laporte, G., Martín, I. R., and González, J. J. S. "Locating median cycles in networks." *European Journal of Operational Research*, 160(2) (2005) 457–470.

[10] Mauttone, A., Nesmachnow, S., Olivera, A., and Robledo, F. "Solving a ring star problem generalization." In "Conference: 2008 International Conferences on Computational Intelligence for Modelling, Control and Automation (CIMCA 2008), Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC 2008), Innovation in Software Engineering (ISE 2008).", 981–986 (2008).

[11] Mladenovic, N. and Hansen, P. "Variable neighborhood search." *Computers & Operations Research*, 24(11) (1997) 1097–1100.

[12] Monma, C. L., Munson, B. S., and Pulleyblank, W. "Minimum-weight two-connected spanning networks." *Mathematical Programming*, 46(1-3) (1990) 153–171.

[13] Naji-Azimi, Z., Salari, M., and Toth, P. "A heuristic procedure for the capacitated m-ring-star problem." *European Journal of Operational Research*, 207(3) (2010) 1227–1234.

[14] Naji-Azimi, Z., Salari, M., and Toth, P. "An integer linear programming based heuristic for the capacitated m-ring-star problem." *European Journal of Operational Research*, 217(1) (2012) 17–25.

[15] Reinelt, G. "TSPLIB - A t.s.p. library." Technical Report 250, Universität Augsburg, Institut für Mathematik, Augsburg (1990).

[16] Resende, M. G. "Greedy randomized adaptive search procedures." In C. A. Floudas and P. M. Pardalos, editors, "Encyclopedia of Optimization," 1460–1469. Springer US (2009).

[17] Richey, M. B. "Optimal location of a path or tree on a network with cycles." *Networks*, 20(4) (1990) 391–407.

[18] Robledo, F. *GRASP heuristics for Wide Area Network design*. Ph.D. thesis, INRIA (2005).

[19] Stoer, M. *Design of Survivable Networks (Lecture Notes in Mathematics)*. Springer (1992).

[20] Zhang, Z., Qin, H., and Lim, A. "A memetic algorithm for the capacitated m-ring-star problem." *Appl. Intell.*, 40(2) (2014) 305–321.

[21] Zorpette, G. "Keeping the phone lines open." *Spectrum, IEEE*, 26(6) (1989) 32–36.

Part II

Diameter Constrained Reliability in Networks Design

Chapter 5

Capacitated m Ring Star Problem under Diameter Constrained Reliability

Here we go one step further designing a network that supports delay sensitive applications and quality of services. Therefore diameter constraint and minimum realibility is introduced.

Capacitated m Ring Star Problem under Diameter Constrained Reliability

Gabriel Bayá, Antonio Mauttone, Franco Robledo [1]

Dpto. de Inv. Operativa. Universidad de la República. Montevideo, Uruguay

Pablo Romero, Gerardo Rubino [2]

Inria Rennes Bretagne-Atlantique. Campus de Beaulieu 35042 Rennes, France

Abstract

We add random link failures into a celebrated robust network design problem, called Capacitated m-Ring Star Problem (CmRSP), where m rings should connect terminal nodes to a source node at minimum cost. The result is a novel $\mathcal{NP}$-Hard network optimization problem, called Capacitated m-Ring Star Problem Diameter Constrained Reliability, or CmRSP-DCR for short. The hardness of the CmRSP-DCR is formally proved. Then, we heuristically address a GRASP resolution, discuss results and trends for future work.

Keywords: Network Optimization, Network Reliability, CmRSP, GRASP.

[1] Email: (gbaya,mauttone,frobledo)@fing.edu.uy
[2] Email: (pablo.romero,gerardo.rubino)@inria.fr.
This work has been partially supported by the Stic AmSud project "AMMA" 2013-2014

1 Motivation

A hot-topic in the fiber optic field is to design an IP/MPLS network over a resilient DWDM physical network, meeting traffic and capacity constraints. The operator must address several requirements, such as resilience under a single-failure point (i.e., 2-node connectivity), the delicate mapping from logical into physical layer (trading high-connectivity and bandwidth resources), among many others. In practice, a shorter routing implies bandwidth savings. Moreover, a single failure will affect a reduced number of applications.

We are given a perfect telephone exchange, called *depot*, several terminal nodes and optional nodes. In urban optical telecommunication networks, two-node connectivity is usually expensive for terminals that are far away from the depot. An elegant cost-effective solution is provided by Roberto Baldacci, Mauro Dell Amico and José Luis Salazar [1]. In order to connect all terminals, the authors propose to find the cheapest m rings joined precisely in the depot, while some terminals can be pending on some node of a ring. The number of nodes within a ring must not exceed the depot capacity, and the cost of pending nodes is different than the cost of the connections within the rings. The minimum-cost design of the m-rings is called Capacitated m Ring Star Problem, termed here CmRSP for short.

However, delay-sensitive applications demand a limited number of hops. Inspired in delay sensitive applications, Héctor Cancela and Louis Petingi introduced a new reliability measure, called diameter-constrained reliability (DCR). If we are given a graph, a terminal set (i.e., a node subset), and a positive integer d (called diameter), we want all pairs to be connected by d hops or less, in a hostile environment where link failures occur. We invite the reader to see [4] for a rich discussion on diameter-constrained reliability and its applications, ranging from FTTH to peer-to-peer networks and flooding-based systems.

Inspired in delay sensitive applications over a FTTH deployment, we consider a mixed CmRSP with DCR constraint. This article is organized in the following manner. The CmRSP is formally defined in Section 2, while the DCR is formally defined in Section 3. The Capacitated m Ring Star Problem with Diameter-Constrained Reliability (CmRSP-DCR for short) is introduced in Section 4. Since CmRSP-DCR belongs to the class of $\mathcal{NP}$-Hard problems, a GRASP methodology is here developed for its resolution. Empirical results are presented in Section 6. Concluding remarks and trends for future work are discussed in Section 7.

2 Capacitated m-Ring Star Problem

We are given a graph $G = (V, E)$ assumed to be simple, a positive integer m, and a tri-partition $V = \{s\} \cup V_S \cup V_T$, being s the depot, V_S optional Steiner nodes and V_T terminal nodes. The source s has a capacity q_s, and there are two classes of connections with different costs: ring-connections are given by a cost-matrix $R = (r_{i,j})$ such that $r_{i,j}$ is the cost of ring-connection between arbitrary nodes $v_i, v_j \in V$; pending-connections are given by another cost-matrix $C = (c_{i,j})$ such that $c_{i,j}$ is the cost between a non-source node $v_i \in V - \{s\}$ and a terminal node $v_j \in V_T$. In the CmRSP, the goal is to choose a minimum cost spanning subgraph $H = \cup_{i=1}^{m} C_{l_i} \cup S_i$, where the C_{l_i}s are cycles that only meet on the source node $s \in C_{l_i}$ and have length l_i, and S_i are some links connected to nodes from C_{l_i}. The capacity constraint implies that $|S_i| + l_i \leq q_s$ for all $i \in \{1, \ldots, m\}$. The CmRSP belongs to the class $\mathcal{NP}$-Hard, since the Traveling Salesman Problem is included in CmRSP (choose $m = 1$ and a matrix C with infinite costs [1]). Therefore, the problem has been heuristically addressed in several opportunities [6,9,15].

3 Diameter Constrained Reliability

We are given a simple graph $G = (V, E)$, a *terminal* set $K \subseteq V$ and a *diameter*, d. Further, let us assume that nodes do not fail, but each link $e \in E$ can fail stochastically and independently, with a certain probability $p_e \in [0, 1]$. We want to find the probability of the event "all pair of nodes from the terminal set K are joined by some path with length d or less". The probability of this event is denoted $R^d_{K,G}$, and is called diameter-constrained reliability (DCR for short). Since the DCR subsumes the probability that a random graph is connected, the exact DCR computation belongs to the class of $\mathcal{NP}$-Hard problems [12]. Indeed, The DCR remains $\mathcal{NP}$-Hard even in a two-terminal scenario with diameter three [5]. Here, we cite a special family of graphs that accept efficient DCR computation. Once the hardness of the DCR is known, several approximation algorithms were developed, as well as exact DCR computation for special families of graphs [3].

Definition 3.1 Let $G = (V, E)$ a simple graph, $K \subseteq V$ and d a positive integer. A subgraph $G' = (V, E')$ is d-K connected if $d(u, v) \leq d$, $\forall u, v \in K$.

Definition 3.2 Let $G = (V, E)$ a simple graph, $K \subseteq V$ and d a positive integer. The graph G is d-K-r weak if for every set $U \subseteq E$ with $|U| \geq r$, the resulting subgraph $G - U$ is not d-K connected.

Proposition 3.3 *Let $G = (V, E)$ a d-K-r weak graph, for some r independent of $n = |V|$. Then, the DCR can be found in polynomial time in n.*

Proof. The number of subgraphs $G' = G - U$ with $|U| < r$ is $\sum_{i=0}^{r-1} \binom{|E|}{i} \sim |E|^{r-1} \leq n^{2r-2} = p(n)$, bounded by the polynomial $p(n)$. Those subgraphs can either be d-K-connected or not. The d-K condition can be checked for those subgraphs in polynomial time, using Breadth First Search (BFS). □

Corollary 3.4 *Any feasible solution for the CmRSP accepts an exact DCR computation, in polynomial time with the number of nodes.*

Proof. Any feasible solution for the CmRSP is d-V-$m + 1$ weak. Since the number of rings m does not depend on n, we are done. □

Furthermore, under identical link reliabilities and high diameter, if $H = \cup_{i=1}^{m} C_{l_i} \cup S_i$ is a feasible solution for CmRSP and $S = \cup_{i=1}^{m} S_i$ is the set of pending links, then the reliability equals the product of ring-reliabilities and pending link reliabilities:

$$R^d_{V,G_H} = p^{|S|} \prod_{i=1}^{m} [p^{l_i} + l_i p^{l_i - 1}(1 - p)]. \tag{1}$$

Equation (1) will be useful to test whether the DCR condition holds for particular networks.

4 Main Problem

The problem we address in this paper is the Capacitated m Ring-Star Problem with Diameter Constrained Reliability, or CmRSP-DCR. The goal is to find a minimum-cost spanning graph consisting of m rings (with the source s as a common node), that respect both the capacity constraint q_s and reliability constraint $R^d_{V_T,G} \geq R_{min}$.

Proposition 4.1 *The CmRSP-DCR belongs to the $\mathcal{NP}$-Hard class.*

Proof. CmRSP is included in CmRSP-DCR. Indeed, we simplify the DCR constraint choosing $R_{min} = 0$. □

5 GRASP Resolution

Greedy Randomized Adaptive Search Procedure (GRASP) is a powerful multi-start or iterative process, with great success in telecommunications [11]. In

GRASP, feasible solutions are produced in a first phase, and neighbor solutions are explored in a second phase. The best overall solution is returned as the result. We invite the reader to consult [10] for a comprehensive study of this metaheuristic. Here, we will sketch the main ingredients of our particular GRASP design, to know, Construction Phase and Local Search Phase.

5.1 Construction Phase

Pick m terminal nodes uniformly at random, and apply Ramesh Bhandari's algorithm [2] in order to find the cheapest pair of node-disjoint paths between the depot and each terminal (the reader can find other efficient minimum-sum path-disjoint construction in [13,14]). Other terminal nodes are greedily added to the cycles, meeting the capacity and diameter constraints.

5.2 Local search phase

The following 5 movements are applied sequentially (several times, until no improvement is possible) whenever both the cost is reduced and feasibility is preserved. They are $TwoOpt$, that swaps any two terminal nodes (in the same or in different cycle), $MoveCycle$ moves one node from one cycle to another, $Reconnect$ deletes one link from each cycle and greedily re-connects their ends, $AddDelete(k)$ that adds k links between nodes in a certain component, and deletes links from nodes with degree higher than two, and $BestPathwPN(p)$, the most sophisticated movement, which replaces a simple path with pendant nodes p, by the best of them (with the same ends), using an exact algorithm based on ILP. These movements are explained in more detail in the thesis [7].

6 Empirical Results

For the sake of simplicity, we work with identical link reliabilities (in practice, this assumption holds unless there is additional information of distinguished links). Therefore, the DCR can be found for all feasible topologies using Equation (1). Observe that CmRSP is a relaxation of CmRSP-DCR, and the cost of feasible solutions for the CmRSP-DCR are lower-bounded by the optimal solution of CmRSP. In order to highlight the main challenges of the new problem and the gap offered by our GRASP methodology, we will contrast against optimal solutions for the CmRSP, choosing instances developed by Roberto Baldacci, Mauro Dell' Amico and José Luis Salazar González [1]. The authors considered instances from TSPLIB, with no Steiner nodes. Instances are divided in two classes (A and B) ranging from 26 to 101 nodes. Both

classes have the same topology, but edge costs are different. In class A, the cost of each link equals the Euclidean distance $r_{i,j} = c_{i,j} = d_{i,j}$, while in class B, $r_{i,j} = \lceil 7d_{i,j} \rceil$ and $c_{i,j} = \lceil 3d_{i,j} \rceil$. We used $m \in \{3, 4\}$, diameter $d \in \{q_s, 9/10 \times q_s, (9/10)^2 \times q_s\}$ and elementary link reliabilities $p_e = 0,99$. Table 1 presents a contrast between the optimum solution for the CmRSP (Z_{best}) and cost achieved meeting the DCR condition ($\bar{Z}$) from a total of 18 instances. The acronym PN stands for the number of pending nodes in the solution, and R^d_{V,G_H} is the DCR found by our GRASP resolution (which is greater than R_{min}). The reader can observe that in this trade-off, a gain of diameter constrained reliability is achieved by an additional cost of 3% or less, for 13 out-of 18 instances. The added cost is never greater than 20% (see instance 12). From a topological viewpoint, pending links penalize the DCR by a factor $p_e = 0,99$. Therefore, the new DCR constraint tries to reduce the number of pending links. For instances marked with an asterisk (3 and 12), our GRASP methodology could not find feasible instances, since the network was stressed by an aggressive diameter constraint. In those cases, a chord has been added to some cycles, and the resulting components could achieve the required diameter specification. In all cases, the new topology is more robust, from both connectivity and reliability aspects. Indeed, the system is ready for both a single failures and random link failures.

7 Concluding Remarks and Future Work

In this article, we explore the interplay between network reliability and topological network design. Specifically, the Capacitated m-Ring Star Problem (CmRSP) is linked with a diameter-constrained reliability (DCR) requirement, resulting another $\mathcal{NP}$-Hard problem, to know, CmRSP-DCR. This combines 2-connected blocks (i.e., rings) with a diameter requirement, suitable for delay sensitive applications. Once the hardness of CmRSP-DCR is established, the problem has been addressed heuristically using GRASP methodology, a celebrated heuristic widely applied in several telecommunication problems.

As a future work, we would like to relax the topology of rings, using arbitrary 2-connected blocks, as we presented in Section 6. Indeed, Clyde Monma proved that the minimum-cost 2-node connected network can be 4/3 cheaper that the cheapest cycle in metric graphs [8]. The reader can find the Capacitated m Two-Node Survivable Star Problem in the thesis [7]. However, an exact (or efficient) computation of DCR in 2-connected networks is still an open problem. Sometimes, the desired diameter forces not to use cycles, but other 2-node-connected structures instead, improving reliability. This is

N	***INSTANCE***	***Q***	d_{max}	R_{min}	***PN***	$\bar{Z}$	Z_{best}	***GAP***	R^d_{V,G_H}	d
1	A09-n026-m04	6	6	0,97	0	382	382	0,000	0,991900	6
2	A09-n026-m04	6	5	0,97	2	407	382	6,545	0,973900	5
3	A09-n026-m04	6	4	0,97	2	450	382	17,801	0,973633(*)	4
4	A19-n051-m03	19	19	0,92	4	458	458	0,000	0,927859	17
5	A19-n051-m03	19	17	0,92	4	458	458	0,000	0,927859	17
6	A19-n051-m03	19	15	0,92	4	476	458	3,930	0,920139	15
7	A31-n076-m03	28	28	0,86	6	570	564	1,064	0,875291	25
8	A31-n076-m03	28	25	0,86	6	570	564	1,064	0,875291	25
9	A31-n076-m03	28	22	0,86	7	606	564	7,447	0,860948	22
10	B09-n026-m04	6	6	0,97	0	2.674	2.674	0,000	0,991801	6
11	B09-n026-m04	6	5	0,97	2	2.808	2.674	5,011	0,973810	5
12	B09-n026-m04	6	4	0,97	2	3.144	2.674	17,577	0,973812(*)	4
13	B19-n051-m03	19	19	0,87	4	3.058	3.015	1,426	0,927859	17
14	B19-n051-m03	19	17	0,87	4	3.058	3.015	1,426	0,927859	17
15	B19-n051-m03	19	15	0,87	11	3.029	3.015	0,464	0,872479	15
16	B31-n076-m03	28	28	0,77	21	3.845	3.740	2,807	0,772797	25
17	B31-n076-m03	28	25	0,77	21	3.845	3.740	2,807	0,772797	25
18	B31-n076-m03	28	22	0,77	21	3.793	3.740	1,417	0,772797	20

Table 1
Optimal solution for CmRSP versus GRASP solution for CmRSP-DCR.

reinforced with Monma prediction of possible savings as well. We have two hints that the generalization is in the right way.

Additionally, we wish to apply these techniques to the design of real-life networks. The knowledge in diameter-constrained reliability provides an insight of cost-reliability trade-off. We encourage the scientific community to combine both reliability analysis with connectivity properties in the design of the physical layer of FTTH systems.

References

[1] Roberto Baldacci, Mauro Dell'Amico, and Juan José Salazar González. The capacitated m-ring-star problem. *Operations Research*, 55(6):1147–1162, 2007.

[2] Ramesh Bhandari. Optimal physical diversity algorithms and survivable networks. In *Computers and Communications, 1997. Proceedings., Second IEEE Symposium on*, pages 433–441, 1997.

[3] Eduardo Canale, Franco Robledo, Pablo Romero, and Pablo Sartor. Monte carlo methods in diameter-constrained reliability. *Optical Switching and Networking*, 14, Part 2(0):134 – 148, 2014. Special Issue on RNDM 2013.

[4] Héctor Cancela, Mohamed El Khadiri, and Louis Petingi. Polynomial-time topological reductions that preserve the diameter constrained reliability of a communication network. *IEEE Transactions on Reliability*, 60:845–851, dec 2011.

[5] Héctor Cancela and Louis Petingi. Reliability of communication networks with delay constraints: computational complexity and complete topologies. *International Journal of Mathematics and Mathematical Sciences*, 2004:1551–1562, 2004.

[6] E. A. Hoshino and C. C. De Souza. A branch-and-cut-and-price approach for the capacitated m-ring-star problem. *Discrete Appl. Math.*, 160(18):2728–2741, December 2012.

[7] Gabriel Bayá Mantani. Diseño Topológico de Redes. Caso de Estudio: Capacitated m Two-Node Survivable Star Problem. Master's thesis, Universidad de la República. Pedeciba Informática, Montevideo, Uruguay, 2014.

[8] Clyde Monma, Beth Spellman Munson, and William R. Pulleyblank. Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46(1-3):153–171, 1990.

[9] Z. Naji-Azimi, M. Salari, and P. Toth. A heuristic procedure for the capacitated m-ring-star problem. *European Journal of Operational Research*, 207(3):1227 – 1234, 2010.

[10] Mauricio Resende and Celso Ribeiro. *Greedy Randomized Adaptive Search Procedures.* In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers, 2003.

[11] Franco Robledo. *GRASP heuristics for Wide Area Network design.* PhD thesis, INRIA/IRISA, Université de Rennes I, Rennes, France, 2005.

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

[13] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974.

[14] J. W. Suurballe and Robert Endre Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984.

[15] Zizhen Zhang, Hu Qin, and Andrew Lim. A memetic algorithm for the capacitated m-ring-star problem. *Appl. Intell.*, 40(2):305–321, 2014.

Part III

Generalizing other Topological Network Design Problem

Chapter 6

The Capacitated Two-Node Survivable Tree Problem

A new combinatorial optimization problem is introduced, called Capacitated Two-Node Survivable Tree Problem. The CTNSTP is a relaxation of the CRTP where cycles can be replaced by 2-node-connected structures if the total cost of network improves.

Capacitated Two-Node Survivable Tree Problem

Gabriel Bayá, Antonio Mauttone, Franco Robledo and Pablo Romero

Departamento de Investigación Operativa
Facultad de Ingeniería - Universidad de la República
Montevideo Uruguay
Email:(gbaya,mauttone,frobledo,promero)@fing.edu.uy

***Abstract*—The object under study is a novel mathematical optimization problem, inspired in the evolution of fiber-optics communication. Real-life implementations must address a cost-robustness tradeoff. As corollary, real topologies are hierarchically organized in backbone and access networks. The backbone is two-node connected, while the access network usually considers either leaf nodes or elementary paths, directly connected to the backbone.**

In this paper, the Capacitated Two-Node Survivable Tree Problem is introduced (CTNSTP for short). The backbone consists of m two-node-connected blocks with a perfect depot as a common node. The access network consists of trees directly connected to the backbone.

The decision version for the CTNSTP belongs to the class of NP-Complete computational problems. As a consequence, a GRASP heuristic enriched with a Variable Neighborhood Descent (VND) is developed. A smart neighborhood of our VND includes the best replacement using integer Linear Programming formulations. A fair comparison among recent works in the field confirm remarkable savings with the novel proposal.

***Keywords*—Network Survivability, CTNSTP, GRASP**

I. Motivation

Availability has been the major cause of concern in telephonic services. A minimally connected topology provides availability, but it is not robust under single point failures. In optic fiber based communication, robustness is essential, so, two-node-connected topologies are considered. A natural approach to reach two-node connectivity is to connect all terminals in a ring or cycle in an economic way. In this scenario a node is connected to another one by two independent paths. This problem is called Traveling Salesman Problem or TSP, and it is widely studied in the scientific literature [1].

A cornerstone in the field of structural network design is authored by Clyde Monma et. al [2]. They study the Minimum-weight Two-Connected Spanning Problem (MW2CSP), briefly, how to connect terminals in the cheapest way, with a resulting two-node connected topology. They prove that the corresponding decision version for the MW2CSP belongs to the set of $\mathscr{NP}$-Complete decision problems. Furthermore, the cheapest Hamiltonian Tour (i.e., a ring that meets all the nodes) is not necessarily a global optimal solution. Specifically, the cost of the cheapest ring is upper-bounded by $4/3 \times opt$, being opt the cost of the best two-node-connected structure.

Inspired by optic fiber design, Martín Labbé et. al. introduce the Ring Star Problem, or RSP for short [3]. In that work the core is a ring, and the remaining terminals are linked to the ring as leaf-nodes. The goal is to find the minimum-cost topology meeting the previous constraints, given costs in the ring-connections and leaf-links. A further generalization, the Capacitated Ring Star Problem (CmRSP) is introduced by Roberto Baldacci et. al. pressed by realistic solutions, where customers are geographically distributed [4]. The authors consider m blocks with the depot as the only common node. The blocks are rings again; the main difference with the RSP is the presence of m rings instead of one. Both optimization problems belong to the $\mathscr{NP}$-Hard class, since they generalize the Hamiltonian Tour [5].Therefore, the CmRSP has been heuristically addressed in several opportunities [6], [7]. A trade-off between cost and robustness is proposed by Alessandro Hill et. al [8] where the core is a ring again, but there are nodes from a secondary class, that are connected to the ring by trees. The result is the Capacitated Ring-Tree Problem, or CRTP for short.

Recent works in structural network design replace rings by arbitrary two-connected components, inspired in the savings predicted by Clyde Monma et al. For instance, Gabriel Bayá et. al. introduce the Capacitated m Two-Node Survivable Star Problem, or CmTNSSP [9] where the m rings of the CmRSP are replaced by two-connected components. Analogously, Rodrigo Recoba et. al. introduce the Two-Node Connected Star Problem (TNCSP), which is precisely the RSP but with a two-node-connected core that replaces the ring [10].

In this paper, a natural extension for both the CRTP and CmRSP is introduced, where m two-connected blocks are considered, and the secondary nodes from the access network includes trees connected to the blocks. The goal is to achieve flexibility and savings simultaneously.
The main contributions of this paper are the following:

- The Capacitated Two-Node Survivable Tree Problem (CTNSTP) is introduced.
- Given its intractability, a heuristic resolution is developed. We adopted a GRASP approach enriched with a Variable Neighborhood Descent, or GRASP-VND.
- A fair comparison with prior works in the field is presented in order to highlight the benefits of this new proposal.

This article is organized in the following manner. The formal definition of the CTNSTP is presented in Section II. A

Greedy Randomized Adaptive Search Procedure (GRASP) is developed for its resolution in Section III. The experimental analysis is conducted in Section IV. Concluding remarks and trends for future work are discussed in Section V.

II. Capacitated Two-Node Survivable Tree Problem

The cost-robustness trade-off is a major engineering challenge to develop physical communication systems. Ideally, the underlying topology should be flexible enough to produce savings, but resilient to simple node/link failures in the backbone. Here we describe the closest works from a topological point of view. In fact, we present a topological extension of the CRTP, gaining on both flexibility and savings.

Stephan Voß and Alessandro Hill recently introduced the CRTP [8]. They consider a simple undirected graph $G = (V,E)$, a positive integer m and a partition $V = \{s\} \cup V_{T_1} \cup V_{T_2} \cup V_S$, being s the depot, V_{T_1} the type-1 terminal nodes, V_{T_2} the type-2 terminal nodes and V_S the optional or Steiner nodes. Steiner nodes can be present in the solution if they improve total cost. The source s has a capacity q_s, and there is a cost-matrix $C = (c_{i,j})$, $v_i, v_j \in V$.

In the CRTP [8], the goal is to choose a minimum cost spanning subgraph $H = \cup_{i=1}^{k} R_{l_i}$, wherein the R_{l_i} are ring-trees (i.e., rings with arborescences) that only meet on the depot $s \in R_{l_i}$ and have a length l_i. Every node from the set V_{T_1} belongs to precisely one ring-tree, while nodes from V_{T_2} belong to exactly one ring. Steiner nodes can either be included or not in the solution. The capacity constraint implies that $l_i \leq q_s$ for all $i \in \{1,\ldots,m_p\}$, with $m_p \leq m$, being m the maximum number of ring-trees allowed. Here, we consider a further extension, where rings are replaced by arbitrary 2-node-connected components, then we obtain the Capacitated Two-Node Survivable Tree Problem (CTNSTP) (Figure 1). The CTNSTP also belongs to the $\mathscr{NP}$-Hard class, since the design of a single component ($m = 1$, $q_s = |V|$, $V_{T_1} = V_S = \emptyset$) is the minimum-cost 2-node-connected spanning network problem (MW2NCSN), which is $\mathscr{NP}$-Hard [2].

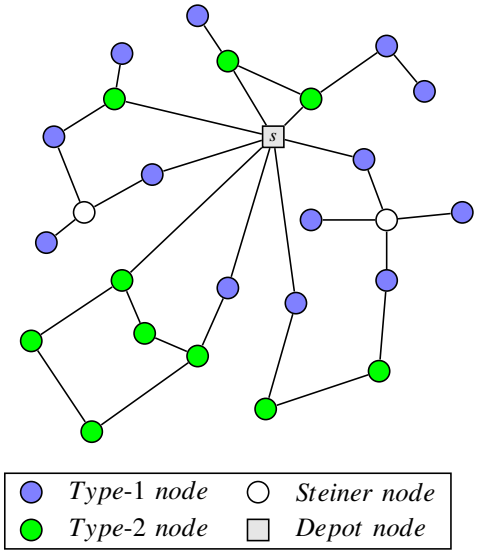

Fig. 1. A feasible solution for the CTNSTP

III. GRASP Resolution

Greedy Randomized Adaptive Search Procedures (GRASP) is a powerful multi-start or iterative process, with great success in telecommunications [11]. In GRASP, feasible solutions are produced in a first phase, while neighbor solutions are explored in a second phase. The best overall solution is returned as the result. There is a trade-off between greediness (intensification) and randomization (diversification), by means of a restricted candidate list. We invite the reader to consult [12] for a comprehensive study of this metaheuristic. Here, we sketch the main ingredients of our particular GRASP design, namely, Construction Phase and Local Search Phase.

A. Construction Phase

During the Construction Phase, components will be iteratively built. The goal is to produce a feasible solution that includes type-2 terminal nodes in 2-node-connected structures, and type-1 terminal nodes in both 2-node-connected structures and trees. Let us consider an arbitrary instance for the CTNSTP, a positive integer k and a maximum number of iterations $MaxIter$. In order to define our construction phase, the following four functions will be used:

1 $Pick(m,G,R,MaxIter)$: returns m terminal nodes $v_1,\ldots,v_m$ from different components with $v_i \in V_{T_1} \cup V_{T_2}$.
2 $Connect(G,C,s,node,k,non\_connected)$: connects the source-node s to every node v_i with k node-disjoint paths.
3 $ChooseTwo(C)$: chooses 2 paths out of m uniformly at random.
4 $Insert(non\_connected,G,C)$: inserts type-2 nodes in the backbone and type-1 nodes in the backbone or in a tree.

Algorithm 1 Construction Phase

```
1: input G, C, k, m, Maxiter
2: G_Sol ← ∅
3: component_nodes ← ∅
4: non_connected ← V_T1 ∪ V_T2
5: {v_1,...,v_m} ← Pick(m,G,R,Maxiter)
6: for i=1 to m do
7:     node = Random(v_1,...,v_m)
8:     C ← Connect(G,C,s,node,k,non_connected)
9:     C_i ← ChooseTwo(C)
10:    G_Sol ← G_Sol ∪ C_i
11:    component_nodes[i] ← component_nodes[i] ∪ C_i
12:    non_connected ← non_connected − C_i
13: end for
14: G_Sol ← G_Sol ∪ Insert(non_connected,G,C)
15: return G_Sol
```

The previous functions will be called sequentially. *Pick* runs $MaxIter$ independent random sets of m terminal nodes. It returns the set with minimum global cost between all the pairs of the set. Once the set $v_1,\ldots,v_m$ is obtained, $Connect(s,v_i,k)$

is called for each node v_i. It applies Ramesh Bhandari's algorithm [13] in order to find the cheapest set of k node-disjoint paths between the depot and terminal v_i (type-1 or type-2). Function *ChooseTwo* just chooses uniformly at random two disjoint paths out of k. Finally, in *Insert*, non-connected type-1 and type-2 nodes are randomly chosen and iteratively added to the smallest component, meeting feasibility. In this way, the capacity constraint is met during the construction phase. Consider an isolated node v and a component C (see Figure.2). All links that belong to other components will be deleted, and the costs of all links from C are set to 0. An artificial node v' is connected to every node from C. Bhandari's algorithm is applied in order to find k (or possibly less) node-disjoint paths between v and v' in the resulting network. Only two disjoint paths between v and v' will be chosen. Finally, the resulting links that connect v with C are added to the solution. Type-1 nodes can either be inserted into an existing tree, or a new tree can be built for that specific purpose.

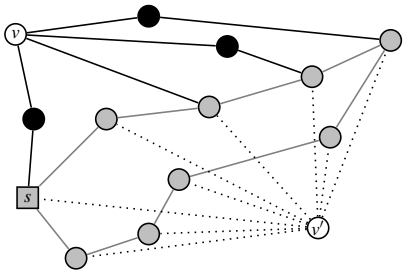

Fig. 2. Including node v into component $\mathscr{C}$.

B. Local search phase

The seven following functions determine different neighborhood structures, which are applied following a variable neighborhood descent (sequential execution; if there is an improvement we return to the first function again):

- *Swap-Nodes*: takes a random terminal node and swaps it with its closest possible terminal node (the possibility means that the cost is decreased),
- *Move-Node*: removes a node, reconnects their neighbors and inserts the node into a tree or 2-node-connected structure,
- *Crossing-Components*: Finds two close terminal nodes from different 2-node-connected structures, deletes adjacent links of them and reconnects the components in the best manner.
- *Add-Links*: random links are added into a fixed 2-node-connected structure, and finally removed meeting feasibility.
- *Tree-Convert*: removes a random number of type-1 nodes in the 2-node-connected structure of a component, and re-insert them in a tree,
- *Move-Steiner* removes Steiner nodes in the solution when this movement improves the cost. Subsequently, this function inserts Steiner nodes in the same way as long as the cost of the solution is improved.
- *Best Component*: replaces each cycle in the solution by its best 2-node connected component, using an exact ILP based algorithm.

In order not to stuck in local optima, a perturbation process takes place. Function *Shake* randomly disconnects a percentage p of terminal nodes and reconnects them in another way. *Shake* is called whenever the previous seven functions are stucked in a solution and do not have activity (i.e., they do not produce better solutions). In the following paragraphs, the seven functions will be explained in full detail.

1) Swap-Nodes: This local search selects two nodes and makes an exchange (swapping) between them. This process starts with a random selection of a type-1 or type-2 terminal node and tests all possible ways to swap this node with another *close* node belonging to a 2-node-connected component (the same or other) or belonging to a tree. To clarify the concept *close* we define a neighborhood related to the considered node. Again we will appeal to the same definition of neighborhood we use in Move-Node local search, (detailed in III-B2), i.e. the neighborhood N of k nodes $j \in T$ closest to the node i. It should be noted that to apply the movement, both nodes must belong to the 2-node-connected structure of the component, or they must belong to different trees.
The algorithm picks a random node i and proceeds as follows. Consider its closest node j. If j belongs to a tree (i belongs to a different tree to allow move) we exchange the nodes between trees removing each of them and inserting in the other tree using a Minimum Spanning Tree algorithm [14]. If j is a node that belongs to a 2-node-connected structure, this function connects adjacent nodes of j to node i and adjacent nodes of i to node j. Each time a swapping movement leads to improvement and keeps the feasibility, the current solution is updated, the possible swapping with other nodes j in descending order of distance are discarded and finally the algorithm continues with the next terminal node i.

2) Move-Node: This local search performs the extraction of all terminal nodes in a random order from their current positions in the solution, and relocates them to another positions either in the 2-node-connected structure of component or in a tree, improving the overall cost without losing feasibility. The extraction procedure is simple, we extract a terminal node and we reconnect the adjacents to the extracted node. To make the insertion of the extracted node we consider the following definition:
Let $i \in T$ be a node extracted with T the set of terminal nodes of the graph and a neighborhood N defined as follows:

$$N(i) = \left\{ j \in V_{T_1} \cup V_{T_2} : j \begin{array}{l} \text{are the } k \text{ nodes } \textit{closer} \text{ to node } i \\ \text{taking into account costs } c_{ij} \\ \text{defined in original graph } G \end{array} \right\} \quad (1)$$

The loop for each terminal node i, ends after having considered all possible insertions between k closest nodes, and

selects the movement that produces the lowest total cost. The algorithm repeats the same procedure for all $i \in V_{T_1} \cup V_{T_2}$ not even considered, by examining $N(i)$ until finally selecting the movement that produces the lowest total cost.

3) Crossing components: This local search (Algorithm 2) takes two *close* nodes (as defined in Section III-B1), each one in different 2-node-connected structure of a component, eliminates one of their adjacent edges (for each node) and connects each pair of nodes (in different component) by the edge that generates the best cost.

Algorithm 2 Crossing picks two close nodes, deletes incident edges and the components are crossed by adding two new edges.

```
1: input G_inic, T, k
2: G_best ← G_inic
3: for (i = 1 to |T_V1 ∪ T_V2|) do
4:    if (i is not in tree) then
5:       Let K be the ordered set of k nodes closest to node i
6:       for (u = 1 to k) do
7:          Let j = u^th node closest to node i
8:          remove an edge adjacent to node i
9:          remove an edge adjacent to node j
10:         Let i' be the opposite end of the edge incident to i
11:         Let j' be the opposite end of the edge incident to
            a j
12:         state_1=generate edges (i, j') and (i', j)
13:         state_2=generate edges (i, j) and (i', j')
14:         select the state that generates feasible solution with
            improved resulting cost
15:         improve = update(G_best)
16:         if (improve) then
17:            breakfor
               {exit FOR loop, we do not consider next closer
               nodes}
18:         end if
19:      end for
20:   end if
21: end for
22: return G_best
```

4) Add-Links: This local search (Algorithm 3) inserts k edges in a 2-node-connected structure of a selected component. Afterwards the function considers all nodes of degree 3 or greater of the component, and removes one incident edge until leaving the node degree in 2, without losing feasibility. This process is performed several times in each component.

5) Tree-Convert: In this local search k, type-1 terminal nodes belonging to a 2-node-connected structure of a component are removed, then they are reinserted in the best positioned tree (if there are any) or a new tree is generated with the removed node and the best positioned node of the component.

6) Move-Steiner: This local search works by deleting and inserting Steiner nodes in the component when they are present. The first stage of this local search considers all Steiner nodes belonging to the solution, and tries to remove them if the total cost improves. Next, the function selects Steiner nodes that are not yet in solution one by one and attempts to re-insert them. Algorithm 5 shows the stages of this local search.

Algorithm 3 In this algorithm, k edges are inserted in each component and other edges are removed in nodes of degree 3 or greater.

```
1: input G_inic, k, maxiter
2: G_best ← G_inic
3: for (h = 1 to maxiter) do
4:    for (i = 1 to m) do
5:       for (j = 1 to k) do
6:          Let u, v ∈ component i, selected randomly
7:          Add edge (u, v) to component i
8:       end for
9:    end for
10:   for each (node u / δ(u) > 2) do
11:      while (G_best is feasible) do
12:         Remove an incident edge to node u
13:      end while
14:   end for
15: end for
16: return G_best
```

Algorithm 4 In this Algorithm, k type-1 nodes are removed in each 2-node connected structures of the component and they are reinserted in a tree.

```
1: input G_inic, k
2: G_best ← G_inic
3: for (i = 1 to m) do
4:    Remove randomly k type-1 nodes ∈ component i
5:    for each (node u removed) do
6:       if (exists tree T ∈ component i) then
7:          insert node u in tree T
8:       else
9:          Create new tree T' linking the node u
            to component i
10:      end if
11:   end for
12: end for
13: return G_best
```

7) Best-Component: This local search is based on Integer Linear Programming. Further information about the model used in this local search can be found in [15]. Given a feasible

solution to the problem, Algorithm 6 identifies all cycles that exist in each component. For each cycle we apply an exact algorithm getting the best replacement solution that changes a cycle by a 2-node-connected topology. As stated in Section I, the best 2-node-connected solution covering a certain set of nodes is not necessarily a cycle, so this local search may include such topologies in our solution. This algorithm takes as input the induced sub-graph of the original graph with nodes of the cycle and some Steiner nodes, and returns the best 2-node-connected sub-graph, i.e it can potentially change a cycle by a 2-node-connected topology if such change improves solution costs. In order to model this local search we used a particular case of **GSP** (Generalized Steiner Problem) [16] wherein connectivity of all its terminal nodes is two.

Algorithm 5 In this algorithm Steiner nodes are removed and reinserted improving the cost of solution.

1: **input** G_{inic}
2: $G_{best} \leftarrow G_{inic}$
3: **for each** (Steiner node $w \in G_{best}$) **do**
4: remove w if $cost(G_{best}) < cost(G_{best} - w)$
5: **end for**
6: **for each** (Steiner node $w \notin G_{best}$) **do**
7: add w in G_{best} if $cost(G_{best} + w) < cost(G_{best})$
8: **end for**
9: **return** G_{best}

Algorithm 6 In this algorithm cycles are replaced by the best 2-node-connected-component.

1: input G, G_{sol}
2: $G_{best} \leftarrow G_{sol}$
3: $q\_cycles = cycles\_count(G_{sol})$ {Number of cycles of G_{sol}}
4: $all\_cycles \leftarrow cycles(G_{sol})$ {Array with cycles of G_{sol}}
5: **for** ($i = 1$ **to** q_cycles) **do**
6: $best =$ **best_component**(G_{best}, G,R, $all\_cycles(i)$)
7: $G_{best} \leftarrow G_{best}$ - $all\_cycles(i)$ + $best$
8: **end for**
9: $improve$=(Cost(G_{best}) < Cost(G_{sol}))
10: **return** $improve$, G_{best}

Algorithm 6 counts and identifies the cycles present in G_{sol} (lines 3 and 4). For each of these cycles the stage **best_component** (line 6) returns the best 2-node-connected structure and the cycle is replaced by the latter (performed in line 7). The function **best_component** resolves the ILP model. It should be noted that neighbor solutions are feasible, so feasibility is preserved during the local search phase.

IV. Experimental analysis

As far as we know, the closest work is the Capacitated Ring-Tree Problem or CRTP. In fact, the CTNSTP is a topological relaxation of the CRTP, and every feasible solution of the latter is feasible in the former. We refer to the work on the CRTP in [8]. In that paper, a considerable number of problem instances used are solved to optimality and those that are unresolved have lower bounds that will guide us to measure the results generated by our application.

We use the test instances provided by Alessandro Hill and reported in [8] and [17] These instances are originated in the Class A instances of CmRSP in [4]. For each Class A instance, a partition of terminal nodes in type-1 and type-2 was made with different distribution of such kind of nodes, summarizing 5 instances for [0, 0.25, 0.5, 0.75, 1] percentage of type-1 nodes.

P	r_1	$\lvert V_{T_2}\rvert$	$\lvert V_{T_1}\rvert$	$\lvert V_S\rvert$	m	q	l_b	u_b	u_{b0}	u_{b1}	Δ	$t(s)$
Q-1	1	0	12	13	3	5	157	**157**	**157**	157	0,000	600
	0.75	3	9				210	**210**	215	**211**	**-1,860**	600
	0.5	6	6				227	**227**	**227**	227	0,000	600
	0.25	9	3				236	**236**	**236**	236	0,000	600
	0	12	0				242	**242**	**242**	242	0,000	600
Q-2	1	0	12	13	4	4	163	**163**	164	166	1,220	600
	0.75	3	9				207	**207**	**207**	207	0,000	600
	0.5	6	6				240	**240**	**240**	240	0,000	600
	0.25	9	3				249	**249**	**249**	249	0,000	600
	0	12	0				251	**251**	**251**	251	0,000	600
Q-3	1	0	12	13	5	3	170	**170**	173	175	1,156	600
	0.75	3	9				242	**242**	244	244	0,000	600
	0.5	6	6				251	**251**	**251**	253	0,797	600
	0.25	9	3				279	**279**	**279**	279	0,000	600
	0	12	0				279	**279**	**279**	279	0,000	600
Q-4	1	0	18	7	3	7	207	**207**	**207**	208	0,483	600
	0.75	4	14				256	**256**	**256**	256	0,000	600
	0.5	9	9				274	**274**	**274**	274	0,000	600
	0.25	13	5				292	**292**	**292**	292	0,000	600
	0	18	0				301	**301**	305	**301**	**-1,311**	600
Q-5	1	0	18	7	4	5	217	**217**	220	223	1,364	600
	0.75	4	14				285	**285**	**285**	288	1,053	600
	0.5	9	9				313	**313**	318	320	0,629	600
	0.25	13	5				334	**334**	**334**	334	0,000	600
	0	18	0				339	**339**	**339**	339	0,000	600
Q-6	1	0	18	7	5	4	227	**227**	231	232	0,433	600
	0.75	4	14				278	**278**	**278**	280	0,719	600
	0.5	9	9				336	**336**	**336**	336	0,000	600
	0.25	13	5				361	**361**	**361**	361	0,000	600
	0	18	0				375	**375**	**375**	375	0,000	600
Q-7	1	0	25	0	3	10	245	**245**	248	248	0,000	600
	0.75	6	19				294	**294**	**294**	296	0,680	600
	0.5	13	12				313	**313**	**313**	313	0,000	600
	0.25	18	7				327	**327**	**327**	327	0,000	600
	0	25	0				328	**328**	**328**	328	0,000	600
Q-8	1	0	25	0	4	7	252	**252**	267	268	0,375	600
	0.75	6	19				311	**311**	315	319	1,270	600
	0.5	13	12				345	**345**	**345**	347	0,580	600
	0.25	18	7				357	**357**	**357**	357	0,000	600
	0	25	0				362	**362**	**362**	362	0,000	600
Q-9	1	0	25	0	5	6	254	**254**	262	268	2,290	600
	0.75	6	19				319	**319**	322	326	1,242	600
	0.5	13	12				369	**369**	372	372	0,000	600
	0.25	18	7				378	**378**	379	**378**	**-0,264**	600
	0	25	0				396	**396**	397	**396**	**-0,252**	600

TABLE I
VALUES FOUND FOR INSTANCES WITH 26 NODES.

Tables I to IV presents a contrast between the optimum solution (when it was reached, otherwise the lower bound) for the CRTP. The acronyms are the following: P is the identifier of the instance, r_1 is the percentage of type-1 nodes, $|V_{T_2}|$, $|V_{T_1}|$ and $|V_S|$ are the number of type-2, type-1 and Steiner nodes of the instance respectively, l_b and u_b the lower and upper bound in the exact resolution method [8], u_{b_0} is the cost of solution using the approximate method in [17] and u_{b_1} the optimum obtained for our metaheuristic.

The parameter Δ is a measure of our GRASP-VND effectiveness, we compare the results obtained in our metaheuristic

with the results obtained in [17]. Parameter Δ is defined as follows:

$$\Delta = \frac{u_{b_1} - u_{b_0}}{u_{b_0}} \quad (2)$$

From 225 instances, we obtained the global optimum in 55 of them, better results were obtained in 73 instances, and the average gap was 0.099.

P	r_1	$\|V_{T_2}\|$	$\|V_{T_1}\|$	$\|V_S\|$	m	q	l_b	u_b	u_{b0}	Z	Δ	$t(s)$
Q-10	1	0	12	38	3	5	156	**156**	**156**	156	0,000	3600
	0.75	3	9				190	**190**	196	196	0,000	3600
	0.5	6	6				213	**213**	215	217	0,922	3600
	0.25	9	3				222	**222**	**222**	222	0,000	3600
	0	12	0				242	**242**	**242**	242	0,000	3600
Q-11	1	0	12	38	4	4	159	**159**	163	166	1,807	3600
	0.75	3	9				209	**209**	**209**	209	0,000	3600
	0.5	6	6				230	**230**	**230**	230	0,000	3600
	0.25	9	3				238	**238**	**238**	238	0,000	3600
	0	12	0				251	**251**	**251**	251	0,000	3600
Q-12	1	0	12	38	5	3	170	**170**	172	173	0,578	3600
	0.75	3	9				203	**203**	**203**	203	0,000	3600
	0.5	6	6				251	**251**	**251**	253	0,791	3600
	0.25	9	3				278	**278**	**278**	278	0,000	3600
	0	12	0				279	**279**	**279**	279	0,000	3600
Q-13	1	0	25	25	3	10	245	**245**	248	248	0,000	3600
	0.75	6	19				293	302	305	306	0,327	3600
	0.5	12	13				311	**311**	312	312	0,000	3600
	0.25	18	7				322	**322**	**322**	322	0,000	3600
	0	25	0				328	**328**	**328**	328	0,000	3600
Q-14	1	0	25	25	4	7	252	**252**	267	269	0,743	3600
	0.75	6	19				304	**304**	321	321	0,000	3600
	0.5	12	13				341	352	352	355	0,845	3600
	0.25	18	7				357	**357**	**357**	357	0,000	3600
	0	25	0				362	**362**	**362**	362	0,000	3600
Q-15	1	0	25	25	5	6	254	**254**	262	267	1,873	3600
	0.75	6	19				331	335	339	337	**-0,593**	3600
	0.5	12	13				359	370	372	372	0,000	3600
	0.25	18	7				372	387	387	**385**	**-0,519**	3600
	0	25	0				390	**390**	397	**392**	**-1,276**	3600
Q-16	1	0	37	13	3	14	304	**304**	**304**	304	0,000	3600
	0.75	9	28				350	375	375	377	0,531	3600
	0.5	18	19				364	376	378	**376**	**-0,532**	3600
	0.25	27	10				379	**379**	380	**379**	**-0,264**	3600
	0	37	0				380	**380**	381	**380**	**-0,263**	3600
Q-17	1	0	37	13	4	11	308	**308**	309	310	0,323	3600
	0.75	9	28				363	**363**	369	376	1,862	3600
	0.5	18	19				384	399	399	403	0,993	3600
	0.25	27	10				396	404	404	404	0,000	3600
	0	37	0				410	**410**	418	**412**	**-1,456**	3600
Q-18	1	0	37	13	5	9	314	**314**	**314**	314	0,000	3600
	0.75	9	28				374	408	408	412	0,971	3600
	0.5	18	19				401	431	431	435	0,920	3600
	0.25	27	10				417	436	436	**433**	**-0,693**	3600
	0	37	0				446	**446**	452	**446**	**-1,345**	3600
Q-19	1	0	50	0	3	19	376	**376**	377	380	0,789	3600
	0.75	12	38				418	427	436	438	0,457	3600
	0.5	25	25				435	445	447	450	0,667	3600
	0.25	37	13				451	**451**	454	454	0,000	3600
	0	50	0				462	**462**	473	**465**	**-1,720**	3600
Q-20	1	0	50	0	4	14	384	**384**	386	392	1,531	3600
	0.75	12	38				423	458	458	**456**	**-0,439**	3600
	0.5	25	25				448	493	493	496	0,605	3600
	0.25	37	13				471	502	502	**496**	**-1,210**	3600
	0	50	0				493	**493**	513	**499**	**-2,806**	3600
Q-21	1	0	50	0	5	12	390	**390**	392	396	1,010	3600
	0.75	12	38				447	491	501	506	0,988	3600
	0.5	25	25				478	526	526	531	0,942	3600
	0.25	37	13				497	525	525	**523**	**-0,382**	3600
	0	50	0				522	526	541	**526**	**-2,852**	3600

TABLE II
VALUES FOUND FOR INSTANCES WITH 51 NODES.

P	r_1	$\|V_{T_2}\|$	$\|V_{T_1}\|$	$\|V_S\|$	m	q	l_b	u_b	u_{b0}	u_{b1}	Δ	$t(s)$
Q-22	1	0	18	57	3	7	213	**213**	214	216	0,935	3600
	0.75	4	14				272	**272**	**272**	276	1,471	3600
	0.5	9	9				288	318	318	318	0,000	3600
	0.25	13	5				303	318	318	318	0,000	3600
	0	18	0				331	**331**	332	331	**-0,301**	3600
Q-23	1	0	18	57	4	5	232	**232**	235	236	0,426	3600
	0.75	4	14				302	309	312	314	0,641	3600
	0.5	9	9				336	**336**	**336**	336	0,000	3600
	0.25	13	5				359	369	369	**367**	**-0,542**	3600
	0	18	0				386	**386**	390	**386**	**-1,026**	3600
Q-24	1	0	18	57	5	4	257	**257**	259	**265**	2,317	3600
	0.75	4	14				325	**325**	**325**	326	0,308	3600
	0.5	9	9				368	379	379	379	0,000	3600
	0.25	13	5				397	**397**	**397**	397	0,000	3600
	0	18	0				448	**448**	451	**448**	**-0,665**	3600
Q-25	1	0	37	38	3	14	320	**320**	**320**	320	0,000	3600
	0.75	9	28				363	390	390	396	1,538	3600
	0.5	18	19				372	402	402	405	0,746	3600
	0.25	27	10				390	403	403	406	0,744	3600
	0	37	0				409	**409**	413	**409**	**-0,969**	3600
Q-26	1	0	37	38	4	11	326	**326**	336	339	0,893	3600
	0.75	9	28				382	402	402	408	1,493	3600
	0.5	18	19				410	455	455	459	0,879	3600
	0.25	27	10				418	460	460	**458**	**-0,435**	3600
	0	37	0				446	458	458	**454**	**-0,873**	3600
Q-27	1	0	37	38	5	9	340	**340**	343	350	2,041	3600
	0.75	9	28				407	446	446	**442**	**-0,897**	3600
	0.5	18	19				426	473	473	474	0,211	3600
	0.25	27	10				443	497	497	**485**	**-2,414**	3600
	0	37	0				477	506	506	**502**	**-0,791**	3600
Q-28	1	0	56	19	3	21	383	**383**	395	398	0,759	3600
	0.75	14	42				427	462	462	469	1,515	3600
	0.5	28	28				438	477	477	480	0,629	3600
	0.25	42	14				461	465	472	474	0,424	3600
	0	56	0				476	**476**	495	**480**	**-3,030**	3600
Q-29	1	0	56	19	4	16	389	**389**	402	406	0,995	3600
	0.75	14	42				441	488	488	489	0,205	3600
	0.5	28	28				466	520	520	525	0,962	3600
	0.25	42	14				492	532	532	**530**	**-0,376**	3600
	0	56	0				514	535	543	**536**	**-1,289**	3600
Q-30	1	0	56	19	5	13	399	**399**	414	420	1,449	3600
	0.75	14	42				469	533	533	536	0,563	3600
	0.5	28	28				493	554	554	554	0,000	3600
	0.25	42	14				512	558	558	549	**-1,613**	3600
	0	56	0				546	557	561	**554**	**-1,248**	3600
Q-31	1	0	75	0	3	28	473	**473**	478	483	1,046	3600
	0.75	18	57				516	551	551	566	2,722	3600
	0.5	37	38				537	564	564	566	0,355	3600
	0.25	56	19				554	564	573	**568**	**-0,873**	3600
	0	75	0				572	**572**	584	**575**	**-1,541**	3600
Q-32	1	0	75	0	4	21	482	**482**	494	500	1,215	3600
	0.75	18	57				531	573	573	575	0,349	3600
	0.5	37	38				552	612	612	614	0,327	3600
	0.25	56	19				586	618	618	**616**	**-0,324**	3600
	0	75	0				603	626	626	**620**	**-0,958**	3600
Q-33	1	0	75	0	5	17	488	**488**	495	501	1,212	3600
	0.75	18	57				552	623	623	630	1,124	3600
	0.5	37	38				585	623	623	625	0,321	3600
	0.25	56	19				608	656	656	**650**	**-0,915**	3600
	0	75	0				641	674	674	**667**	**-1,039**	3600

TABLE III
VALUES FOUND FOR INSTANCES WITH 76 NODES.

V. CONCLUSIONS AND FUTURE WORKS

The Capacitated Two-Node Survivable Tree Problem (CTNSTP) has been introduced. As far as we know, it has not been studied in prior literature. The need for redundancy and cheaper costs in network deployment is remarkable. Inspired by theoretical results and the related problem CmRSP, we propose an alternative problem where rings are replaced by arbitrary two-node connected components. Both problems are computationally intractable. Therefore, heuristics are suitable for large case scenarios. The CTNSTP has been heuristically addressed using a GRASP metaheuristic enriched with a Variable Neighborhood Descent (VND) and one exact local search. Results from the literature concerning CRTP were taken as reference for comparison. In all cases, the components obtained were cycles instead of other two-connected topologies. Further research is needed in order to understand the nature of problem instances which influence these results. The problem could by extended in a suitable way to model delay-sensitive applications. To meet this goal, diameter constraints should be introduced to ensure connectivity between terminal

P	r_1	$\|V_{T_2}\|$	$\|V_{T_1}\|$	$\|V_S\|$	m	q	l_b	u_b	u_{b0}	u_{b1}	Δ	$t(s)$
Q-34	1	0	25	75	3	10	274	**274**	282	282	0,000	7200
	0.75	6	19				314	**314**	327	**325**	**-0,612**	7200
	0.5	12	13				337	353	353	**350**	**-0,850**	7200
	0.25	18	7				356	363	363	363	0,000	7200
	0	25	0				366	**366**	**366**	366	0,000	7200
Q-35	1	0	25	75	4	7	289	**289**	293	293	0,000	7200
	0.75	19	6				344	367	367	367	0,000	7200
	0.5	12	13				367	405	405	**404**	**-0,247**	7200
	0.25	18	7				385	416	416	418	0,481	7200
	0	25	0				409	425	425	**423**	**-0,471**	7200
Q-36	1	0	25	75	5	6	299	**299**	**299**	299	0,000	7200
	0.75	19	6				361	393	393	**390**	**-0,763**	7200
	0.5	12	13				378	403	403	**401**	**-0,496**	7200
	0.25	18	7				407	429	429	432	0,699	7200
	0	25	0				440	452	452	**450**	**-0,442**	7200
Q-37	1	0	50	50	3	19	411	**411**	**411**	411	0,000	7200
	0.75	12	38				457	492	492	**490**	**-0,407**	7200
	0.5	25	25				473	499	499	**496**	**-0,601**	7200
	0.25	37	13				483	503	503	**499**	**-0,795**	7200
	0	50	0				493	508	523	**516**	**-1,338**	7200
Q-38	1	0	50	50	4	14	415	**415**	420	423	0,714	7200
	0.75	12	38				460	480	480	481	0,208	7200
	0.5	25	25				484	517	517	**512**	**-0,967**	7200
	0.25	37	13				501	531	531	**528**	**-0,565**	7200
	0	50	0				525	537	537	**532**	**-0,931**	7200
Q-39	1	0	50	50	5	12	426	**426**	443	445	0,451	7200
	0.75	12	38				481	505	505	505	0,000	7200
	0.5	25	25				495	527	527	524	**-0,569**	7200
	0.25	37	13				523	564	564	**556**	**-1,418**	7200
	0	50	0				553	574	574	**570**	**-0,697**	7200
Q-40	1	0	75	25	3	28	511	**511**	516	517	0,194	7200
	0.75	18	57				555	594	594	**588**	**-1,010**	7200
	0.5	37	38				570	592	592	596	0,676	7200
	0.25	56	19				588	612	612	**610**	**-0,327**	7200
	0	75	0				606	**606**	622	**613**	**-1,447**	7200
Q-41	1	0	75	25	4	21	516	**516**	519	521	0,385	7200
	0.75	18	57				559	595	595	597	0,336	7200
	0.5	37	38				582	607	607	**603**	**-0,659**	7200
	0.25	56	19				603	619	619	**612**	**-1,131**	7200
	0	75	0				624	639	642	**632**	**-1,558**	7200
Q-42	1	0	75	25	5	17	522	**522**	529	531	0,378	7200
	0.75	18	57				584	653	653	654	0,153	7200
	0.5	37	38				598	645	645	**644**	**-0,155**	7200
	0.25	56	19				622	670	670	**662**	**-1,194**	7200
	0	75	0				649	689	689	**686**	**-0,435**	7200
Q-43	1	0	100	0	3	38	555	**555**	**555**	556	0,180	7200
	0.75	25	75				611	652	652	654	0,307	7200
	0.5	50	50				624	657	660	660	0,000	7200
	0.25	75	25				644	648	656	**652**	**-0,610**	7200
	0	100	0				663	**663**	683	**677**	**-0,878**	7200
Q-44	1	0	100	0	4	28	564	**564**	568	568	0,000	7200
	0.75	25	75				624	663	663	666	0,452	7200
	0.5	50	50				644	690	690	**682**	**-1,159**	7200
	0.25	75	25				665	683	691	**684**	**-1,013**	7200
	0	100	0				684	700	700	**692**	**-1,143**	7200
Q-45	1	0	100	0	5	23	570	**570**	576	580	0,694	7200
	0.75	25	75				629	695	695	698	0,432	7200
	0.5	50	50				674	717	717	722	0,697	7200
	0.25	75	25				689	730	730	**714**	**-2,192**	7200
	0	100	0				709	743	743	**733**	**-1,346**	7200

TABLE IV
VALUES FOUND FOR INSTANCES WITH 101 NODES.

nodes by a limited number of hops. Naturally, there exists a cost-reliability trade-off when this constraint is added to the problem. Two-node-connected components (not purely cycles) can meet this objective from a topological point of view. The addition of a diameter constraint introduces structural complexity, covering other network requirements such as quality of service (QoS). We are currently studying diameter-constrained scenarios for this problem, as well as the Capacitated Two-Node Survivable Star Problem previously introduced in [15].

ACKNOWLEDGMENT

The authors wish to thank Alessandro Hill for the contribution of the test cases of his work.

REFERENCES

[1] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[2] C. Monma, B. S. Munson, and W. R. Pulleyblank, "Minimum-weight two-connected spanning networks," *Mathematical Programming*, vol. 46, no. 1-3, pp. 153–171, 1990.

[3] M. Labbé, G. Laporte, I. R. Martín, and J. J. S. González, "The ring star problem: Polyhedral analysis and exact algorithm," *Networks*, vol. 43, no. 3, pp. 177–189, 2004.

[4] R. Baldacci, M. Dell'Amico, and J. J. S. González, "The capacitated m-ring-star problem," *Operations Research*, vol. 55, no. 6, pp. 1147–1162, 2007.

[5] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[6] E. A. Hoshino and C. C. de Souza, "A branch-and-cut-and-price approach for the capacitated m-ring-star problem," *Discrete Appliee Mathematics*, vol. 160, no. 18, pp. 2728–2741, Dec. 2012.

[7] Z. Zhang, H. Qin, and A. Lim, "A memetic algorithm for the capacitated m-ring-star problem," *Applied Intelligence*, vol. 40, no. 2, pp. 305–321, 2014.

[8] A. Hill and S. Voß, "Optimal capacitated ring trees," *EURO Journal on Computational Optimization*, vol. 4, no. 2, pp. 137–166, 2016.

[9] G. Bayá, A. Mauttone, F. Robledo, and P. Romero, "Capacitated m two-node survivable star problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 253 – 260, 2016, {INOC} 2015 7th International Network Optimization Conference.

[10] R. Recoba, F. Robledo, P. Romero, and O. Viera, "Two-node-connected star problem," *International Transactions in Operational Research*, pp. n/a–n/a, 2016. [Online]. Available: http://dx.doi.org/10.1111/itor.12362

[11] F. Robledo, "GRASP heuristics for Wide Area Network design," Ph.D. dissertation, INRIA/IRISA, Université de Rennes I, Rennes, France, 2005.

[12] M. Resende and C. Ribeiro, "Grasp: Greedy randomized adaptive search procedures," in *Search Methodologies*, E. K. Burke and G. Kendall, Eds. Springer US, 2014, pp. 287–312.

[13] R. Bhandari, "Optimal physical diversity algorithms and survivable networks," in *Computers and Communications, 1997. Proceedings., Second IEEE Symposium on*. IEEE, 1997, pp. 433–441.

[14] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[15] G. Bayá, A. Mauttone, and F. Robledo, "The capacitated m two node survivable star problem," *Yugoslav Journal of Operations Research*, vol. 27, no. 2, 2017.

[16] S. E. Dreyfus and R. A. Wagner, "The steiner problem in graphs," *Networks*, vol. 1, no. 3, pp. 195–207, 1971. [Online]. Available: http://dx.doi.org/10.1002/net.3230010302

[17] A. Hill, "Multi-exchange neighborhoods for the capacitated ring tree problem," in *International Conference on Numerical Methods and Applications*. Springer, 2014, pp. 85–94.