THESIS

to be presented at

Universidad de La República, UdelaR

in order to obtain the title of

MAGISTER EN INGENIERÍA MATEMÁTICA

for

Eng. Javier PEREIRA LUCAS

Research Institute : LPE - IMERL
University Components :
UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

Thesis title :

Routing cost optimization in Multi Overlay Robust Networks

to be presented on May 2013 to the Comitee of Examiners

Dr. Pablo	BELZARENA	Academic Director
Dr. Alvaro	MARTIN	Thesis Director
Dr. Franco	ROBLEDO	Thesis Director
Dr. Alejandra	BEGHELLI	President
Dr. Gerardo	RUBINO	
Dr. Martín	VARELA RICO	

Acknowledgements

This work is dedicated not only to those who helped me to deal with all the challenges I faced in order to complete my thesis but also to those who gave me emotional support throughout this time.

I would specially like to thank to my Thesis Directors, Dr. Franco Robledo and Dr. Alvaro Martín, for the original idea of the thesis and for all the support I had to solve this problem.

I would also like to thank ANTEL, Uruguay's biggest telecommunications service provider, not only for supporting this work and the local research team with valuable information, but also for helping in the development of many research groups in different research areas all over the country.

Last but not least I would like to thank to the ones who are always present when I need support, my family and friends.

Contents

Summary

In the present work we solve the problem of data flow routing in Multi-Overlay Robust Networks (MORN) while aiming to minimize its routing cost. This kind of networks are typically IP/MPLS Data Network deployed over an SDH/DWDM transport infrastructure.

Through the IP/MPLS Multi-Layer Data Network different kinds of services having a wide variety of quality of service requirements are delivered. Those services are being transported by an SDH/DWDM Transport Network which has different transport capacities. In this network, routing cost depends not only on the assigned transport capacity but also in the technology that it uses.

Our problem seeks not only to route data flows through Data and Transport Networks but also to optimize routing costs and the reliability of the network. The inputs of our problem are the topology of the Data and Transport networks as well as the budget that the network operator has in order to improve its network routing costs and reliability. We will assume that the operator can only use that budget for installing new links between existing transport nodes. The output of the problem is the data flow routing in the Data and Transport Networks and its associated cost. Routing in the Transport Network is calculated not only in the nominal scenario - when all the Transport Network links are up and running - but also in each single transport link failure case.

Resumen

En el presente trabajo se resuelve el problema de rutear flujos de datos en una Red Multi-Capa Robusta (MORN por sus siglas en inglés), mientras que se trata de minimizar el costo asociado a su ruteo. Este tipo de redes son generalmente redes de datos IP/MPLS desplegadas sobre una infraestructura de transporte SDH/DWDM.

Sobre la red de datos IP/MPLS se cursan distintos servicios con diferentes requerimientos de calidad de servicio (QoS). Los servicios de la Red de Datos son transportados por la red SDH/DWDM la cual tiene distintas capacidades de transporte. En éste tipo de redes el costo asociado al transporte depende no solo de la capacidad asignada para el transporte sino que también depende de la tecncología utilizada para transportar dicha capacidad.

En el problema no sólo se busca enrutar flujos de datos a través de las Redes de Datos y Transporte sino que también se busca optimizar los costos de ruteo y la confiabilidad de la red. Como punto de partida, el problema toma como información la topología de las Redes de Datos y Transporte así como cierto presupuesto que el operador de la red posee para poder mejorar los costos de ruteo y la confiabilidad de su red. Asumiremos que dicho presupuesto solo puede ser utilizado para instalar nuevos enlaces entre los nodos existentes en la Red de Transporte. La salida del problema es el ruteo de los flujos de datos tanto en la Red de Datos como en la de Transporte, así como el costo asociado a dicho ruteo. El ruteo en la Red de Transporte se calcula no solo en el escenario nominal - cuando todos los enlaces de la Red de Transporte están funcionales - sino que también en cada escenario de falla simple en sus enlaces.

Part I

INTRODUCTION

Chapter 1

Introduction

1.1 Introduction

Telecommunication networks play an extremely important role in our communities, allowing to share information in a wide range of activities. These activities span from entretainment (online gaming, social networking, etc) to all kind of critical activities such as science, education or health.

In last years most telecommunication companies started deploying optical fiber networks. Throughout this work, this optical network will be refered to as the *physical network*. To guarantee service availability, these networks were designed in a way that several independent paths are available between each pair of nodes. As optimizing budget allocation is a key requirement for operators, several algorithms were developed to address this point. Many research groups in works as [23, 25, 30] have been working for many time in this area.

The exponential growth of Internet traffic volume led to the deployment of Dense Wavelength Division Multiplexing (DWDM) technology [5]. This technology allows multiplexing several connections over one single optical fiber using different wavelengths, and rapidly became very popular with telecommunications companies because it allowed them to expand the capacity of their networks without laying more optical fiber. A DWDM link is a logical connection between two nodes that have an optical fiber connection. A DWDM path is a set of links that connects, from a logical point of view, two remote nodes. Nowadays, DWDM is the most popular network technology in high-capacity optical backbone networks. Optical repeaters must be placed at regular intervals for compensating the loss in optical power while the signal travels along the fiber. Therefore, the cost of a DWDM path is proportional to its length over the physical network.

DWDM supports a set of standard high-capacity interfaces (e.g. 1, 2.5, 10 or 40 Gbps). The cost of a connection also depends on the capacity but not proportionally. Due to economies of scale, the higher the bit-rate the lower is the ratio cost per capacity. DWDM nodes and paths form a so-called *transport network*. This network runs on top of the physical one.

As traffic grows, the number of DWDM links per physical connection must increase. This

may cause multiple logical link failures from a single physical link failure. This scenario is shown in Figure 1.1.

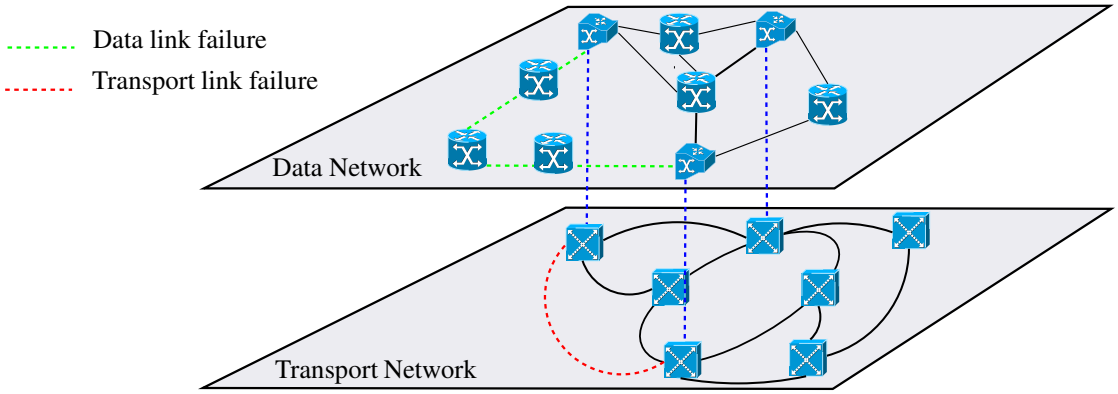

Figure 1.1 – Example of multiple logical link failure from a single physical link failure.

This issue led to the development of new multi-layer models aware of all network layers. Most of these models share in common the 1+1 protection mechanism. This means that each traffic demand required between two DWDM nodes must be routed through two independent paths, so in case of any single physical link failure, at least one of them survives. Routing optimization of a two-layered-network is significantly more complex than single-layered ones. Our concept of route optimization refers to the duty of finding minimum-cost routes. So, having a definition of routing costs (for instance its length), the routes to be found for all traffic requirements must be the ones that produce the lowest cost for the operator. This topic has been widely studied and some examples can be found in [13, 14, 26, 36, 37, 40, 41].

Another widely used transport layer technology is Synchronous Digital Hierachy (SDH) [3, 6–8] as it has 1+1 protection as its native protection mechanism.

The transport layer builds the Transport Network that is built using either SDH or DWDM as transport technologies. We will consider the exchanged traffic between transport nodes as static. Once a traffic demand is established between two nodes of the network, a path with enough capacity is built in order to route such demand. That capacity can not be used by other traffic demands so it is consumed entirely regardless it is effectively used or not.

During many years the connections of IP networks were implemented over SDH. As a consequence IP networks rarely suffered from unplanned topology changes. Most recently, Multi-Protocol Label Switching (MPLS) [39], traffic engineering extensions for dynamic routing protocols (e.g. OSPF-TE [24]), fast reroute algorithms (FRR [31]) and other new features were added to traditional IP routers. This new technology bundle known as IP/MPLS, opens a competitive alternative against traditional protection mechanisms based on SDH.

Data Network is built on top of the Transport Network. It is made up of “virtual links” and Data Network elements such as IP/MPLS routers. Data links are called virtual as they are not associated to any static physical path. The Data Network uses the services provided by

the Transport Network to route its virtual links. Figure 1.2 shows an example of virtual links mapping into a Transport Network.

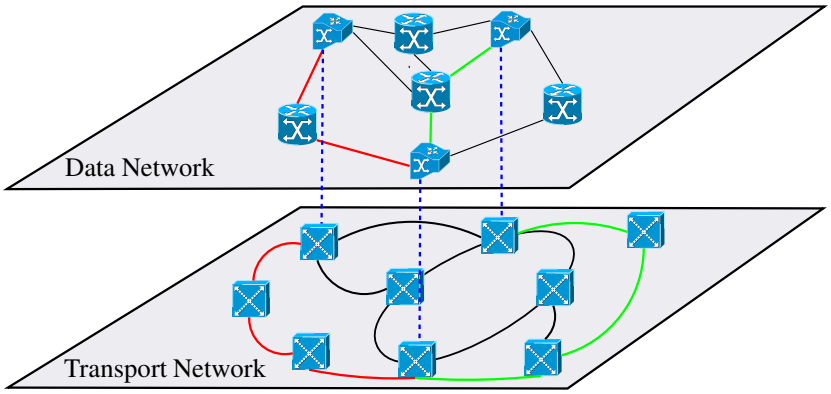

Figure 1.2 – Example of virtual links mapping into a Transport Network.

In order to cope with data traffic demands, "virtual capacities" are created in the Data Network. These capacities can be estimated as explained in [38] and will be considered as given. Virtual capacities define demands for the Transport Network, leading to the creation of capacities in the physical layer. These capacities are created using idle capacity or by installing new optical fibers. Virtual links and virtual capacities builds the "virtual layer" in the Data Network.

As we will see later on, our problem presents many traffic demands between Data Network users. Such demands are usually called commodities, so we are dealing with a class of multi-commodity flow problems. This kind of problems have been widely treated in literature such as in [9, 16, 22, 28, 35].

One remarkable characteristic of the data flows in the Data Network that we will work with is that they are unsplittable, so data traffic belonging to a given flow must follow the same path. This characteristic is not always assumed in literature, some examples in which data flows can be splitted are [9, 16, 21, 26, 28, 29, 34, 35, 41]. On the other hand, unsplittable flow problems have been addressed in [10, 42]. However, they only consider a single-layer network. An illustration of the differences between splitted and unsplitted data flows in a Data Network is shown in Figure 1.3. In that Figure it can be observed how, in Data Network B, the traffic between two endpoints goes through two different paths, so we say that the traffic is splitted between both paths. On the other hand, in Data Network A, all the traffic goes through the same path, so we say it is unsplitted.

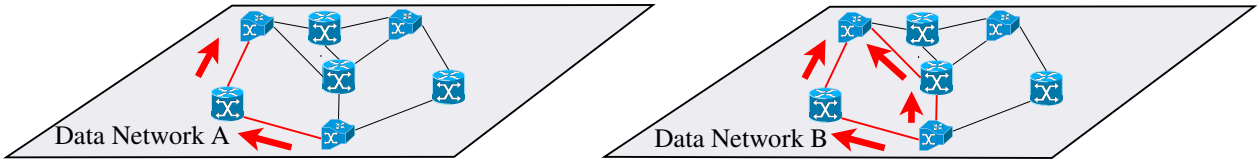

Figure 1.3 – Illustration of a Data Network implementing unsplitted (A) and splitted (B) data flows.

IP/MPLS networks are usually represented in a stack diagram like the Open System Interconnection (OSI) basic reference model that is defined in the ITU-T X.200 recommendation [4]. Figure 1.4 shows the interaction of different layers in a stack diagram. Figure 1.5 shows a network diagram of a multilayer structure. Layer 1 represents the Physical network, Layer 2 represents the Transport Network and Layer 3 represents the Data Network.

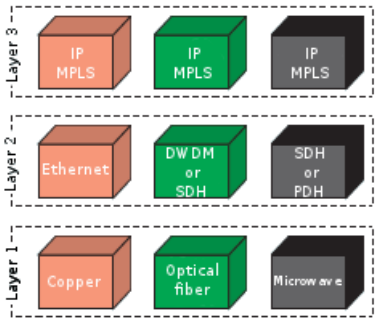

Figure 1.4 – Examples of different technologies interacting in a stack structure.

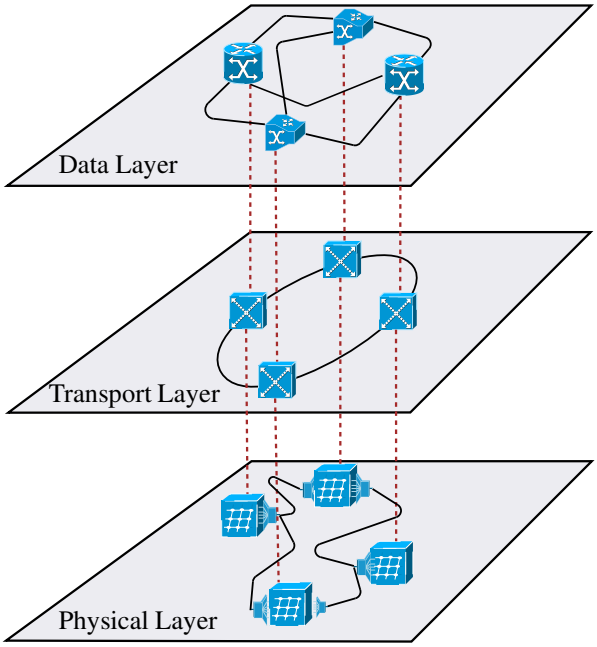

Figure 1.5 – Example of a multilayer network.

1.2 Problem Overview and Related work

In the present work we address the problem of finding the minimum cost-configuration of a logical topology over a physical layer. That physical layer can be extended according to pre-esablished budget. We will refer to it as the installation budget.

Our goal is to develop an algorithm that helps telecommunication companies to design and plan the expansion of its telecommunication network considering current and expected data traffic demands constrained on the installation budget.

Considering as an input the traffic requirements of the Data Network (data flows), the solution of the problem is presented as a routing in the Data Network and Transport Network. The routing is performed minimizing its cost and must be resilient to simple failures in the Transport Network. This means that when one link of the Transport Network fails, the traffic can still be routed. In order to achieve this, the solution can span over the initial Transport Network or eventually use additional transport links that must be installed. The number and type of transport links that can be installed is constrained on the installation budget. We will refer to this problem as the Multi-Overlay Budget-Constrained Robust Network problem or simply MOBCRN.

The inputs of the problem are:

1. Data flow requirements between the Data Network terminals.

2. Data Network and Transport Network topologies.
3. The budget to install new links.
4. The cost of installing each potential new link in the Transport Network.

The Data Network spans over a network that can be modeled using a complete graph. This means that every data node can establish a link with any other data node in the Data Network. This can be justified by the fact that Data Network nodes are connected through virtual links that are established dinamically once a data flow needs to be routed.

As the Transport Network usually has a topology structured in rings, we can model it as the composition of one or more cycles.

As we mentioned earlier, to solve our problem we must route data flows in the Data Network and Transport Network targeting to optimize the routing costs. In case it is necessary and possible, this is achieved by expanding the Transport Network installing new links.

Some related problems have been studied in the past. In [33], Bley et al. present a model-based optimization for the design of multi-layer networks. Based on reference networks from the German research project EIBONE [2], they investigate the influence of various planning alternatives on the total design cost. These alternatives include a comparison of point-to-point versus transparent optical layer architectures, different traffic distributions and the use of SDH vs. Ethernet [1] interfaces.

In [27], Koster et al. address a planning problem in the design of SDH/DWDM multi-layer telecommunication networks. Their goal is to find a minimum cost installation of links and nodes in both network layers such that traffic demands can be realized. The problem is solved using mixed-integer programming techniques that takes into account constraints as node or link failures.

In [38], Risso address the problem of deploying a data network, tipically IP over MPLS, relying on the services of a lower layer using transport technologies such as SDH or DWDM. That goal is achieved implementing routing robustness at single-failure links and at optimum cost. The problem is solved applying meta-heuristics methods based on Greedy Randomized Adaptive Search Procedure (GRASP) [20].

In [32], Parodi studies the problem of designing a Data Network using an existing Transport Network in a robust way and at minimum cost. Network design involves decisions about the network topology, link capacities and traffic routing. This is achieved by using different binary integer programming models implemented in CPLEX.

In [18], Despaux study the optimization of a multi-overlay network implementing heuristics (tabu-search-based).

In [17], Corez address the problem of multi-overlay network planning applying a variable neighbourhood search approach.

The design of multilayer networks has also been studied in [11–13].

1.3 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2: *Model Entities*, introduces the network elements and mathematical concepts that compose the MOBCRN problem.

Chapter 3: *Formal Definition of the Problem*, explains the abstract model and the structure of the problem.

Chapter 4: *An Algorithm for the MOBCRN*, explains step-by-step how the algorithm works.

Chapter 5: *Test Cases Description*, describes all the test cases implemented in order to validate and explore the functionalities of the algorithm designed.

Chapter 6: *Results of the Test Cases*, shows the results obtained when running the set of test cases explained in the aforementioned chapter.

Chapter 7: *Conclusions*, summarizes the project and presents the conclusions.

Chapter 8: *Open Problems and Future Work*, presents some open problems that can be studied taking the present work as base.

Part II

PROBLEM DESCRIPTION AND FORMAL MODEL

Chapter 2

Model Entities

2.1 Introduction

This chapter introduce the network elements that are part of the MOBCRN problem. We define the Data and Transport Network with their different elements and characteristics and how we model the costs. We also explain the relation between Data and Transport Networks and the routing within each network. Finally, we define the problem formally.

2.2 The Data Network

Definition 2.2.1 *A Data Network graph is a simple undirected graph, $G_D = (V_D, E_D)$, where V_D represents the set of data nodes and E_D represents the set of edges.*

$\mathbf{G_D = (V_D, E_D)}$ **topology:** The Data Network graph can be either a simple graph or a multi-graph. For convinience we will assume that the Data Network graph is a simple graph whose edges can eventually represent as much parallel edges as necessary.

We will also consider that the graph is undirected. This means that given any edge $e_d \in E_D$ that connects two data nodes v_a, $v_b \in V_D$, the communication between both nodes is bi-directional and full duplex. This is the same as having two uni-directional edges between both nodes compacted into just one node.

Finally, we also must consider the different alternatives for the connection between data nodes. We will consider that every node $v_d \in V_D$ can be directly connected to each other.

Figure 2.1 shows an example of a Data Network.

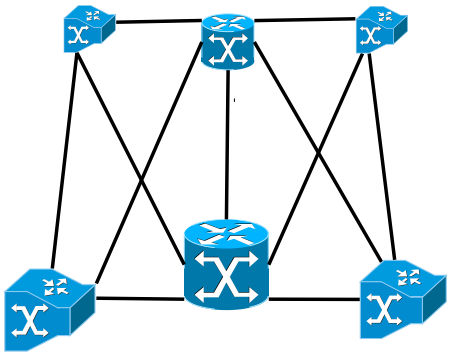

Figure 2.1 – Example of a Data Network.

2.2.1 Data Traffic

Data traffic will be generated or terminated by data nodes and it is usually variable in time. In order to model data traffic we can consider two different types of traffic: committed and excess.

Committed Traffic: This type of traffic must be carried by the Transport Network even if a simple failure [1] occurs. This traffic must be transported entirely and all the time, also considering QoS parameters such as *delay* or *jitter*. This type of traffic is usually related to real time multimedia data as voice over IP (VoIP) or video on demand (VoD).

Excess Traffic: This type of traffic will be available only a portion of time no matter if there is a simple failure or not. For example, excess traffic can be carried if there is enough capacity available in the installed routes. This type of traffic is usually related to best effort traffic such as Internet traffic.

Committed and excess traffic between each pair of data nodes $v_i, v_j \in V_D$ will be assumed to be known. We will denote them as $\dot{m}_{ij}$ and $\ddot{m}_{ij}$, respectively. We will refer to $\vec{m}_{ij} = (\dot{m}_{ij}, \ddot{m}_{ij})$ as the traffic demand vector. Its components are the committed and excess traffic. It represents the traffic that would be carried by the network in case it would have infinite transport capacity.

1. A simple failure occurs when one and only one link is unavailable for some reason.

Definition 2.2.2 *Function f, $f : \mathbb{R}^2 \to \mathbb{R}$, maps the traffic demand vector, $\vec{m}_{ij} = (\dot{m}_{ij}, \ddot{m}_{ij})$, in a single real value, $\bar{m}_{ij}$ which is the estimated traffic demand. More details about how this function operates can be found in [38].*

Definition 2.2.3 *We will refer to $\bar{M} = [\bar{m}_{ij}]_{1 \leq i,j \leq |V_D|}$ as the demand matrix of the problem. It is a matrix whose elements $\bar{m}_{ij} = f(\dot{m}_{ij}, \ddot{m}_{ij})$ represent the demand between v_i and v_j.*

2.3 The Transport Network

The Transport Network will be represented by a simple, non-directed, planar, 2-vertex-connected graph. We can assume that the graph is planar as the optical fiber canalizations are. In case links must be overlapped we can install a network station on the overlapping point.

Definition 2.3.1 *A Transport Network graph is a graph, $G_T = (V_T, E_T)$, where V_T and E_T represents the set of nodes and links of a Transport Network, respectively.*

$\mathbf{G_T = (V_T, E_T)}$ topology: The Transport Network graph can be designed with or without link protection. As per "link protection" we understand the possibility of route a flow by two edge-disjoint paths in G_T. This enables a transport node to switch between these paths if a link failure occurs in one of them. We will consider that the topology of the Transport Network is such that for every two nodes v_r, $v_s \in V_T$ there are at least two edge-disjoint paths that connect them. The most typical configuration that represents this scenario is a multi-ring topology. This is the most common topology used in SDH networks as SDH network elements has the capability of switching the traffic from one side of the ring to another in a few miliseconds (tipically less than 50 ms). We can say that the Transport Network is 2-vertex-connected as it is built upon the transport "rings" concatenation which always have at least two nodes in common.

Definition 2.3.2 *In same situations the telecommunication network will share a Data Network and a Transport Network node in the same building. Function tns, $tns : V_D \to V_T$, is such that $tns(v_d) = v_t$ when this situation happen. This function returns the transport node v_t located in the same Network Station as the data node v_d.*

Figure 2.2 shows an example of a Transport Network.

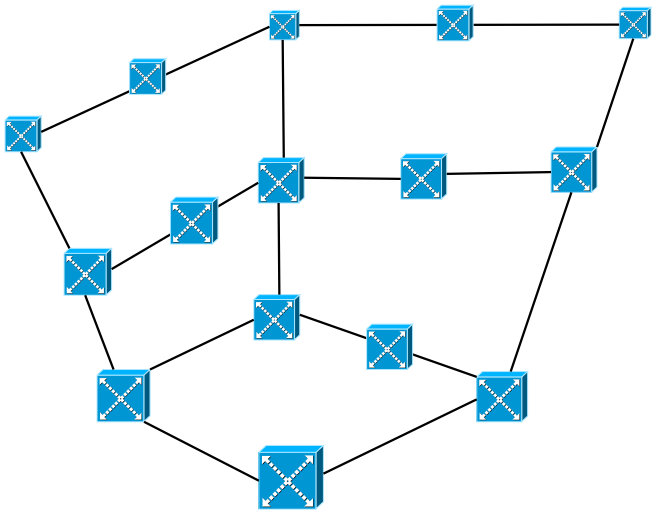

Figure 2.2 – Example of a Transport Network.

2.3.1 Transport Links

Transport links does not have any capacity limit. This means that we must install as much "transport resources" as needed to cope with Data Network traffic demand. Transport resources can be optical fibers, processing or switching boards and even a whole new transport equipment.

Another consideration is that, according to our modeling, a failure in a transport link affects all the connections that are being transported through that link. Although it is possible that not all the optical fibers corresponding to the same link fail at the same time, in general, optical fiber failures are caused by external factors such as canalization damages that usually affects all optical fibers corresponding to that canalization.

2.3.2 Transport Network Traffic

Transport Network traffic is completely static. Once a traffic demand is established between two endpoints - a traffic flow - we need to find a path between these endpoints. That path must have enough capacity to cope with the traffic demand and once it is established the required bandwidth is removed from the capacity of the links that compose that path. As we stated before, in order to achieve this we will install as many links as needed to match traffic demand needs.

The same happens with routing. Once a route flow is established it remains static. When some link fails, all flows that are being carried through that link are out of service until that link

becomes available again.

2.3.3 Transmission Costs

In Time Division Multiplex (TDM) technologies, such as SDH, the transmission cost of a flow is proportional to the product of the length of the route and the bandwidth of the flow. The traffic is transported in containers which have fixed bandwidths. In the case of optical technologies, such as DWDM, the assigned resource to transport traffic is not a container but a wavelength over which a wide variety of speeds can be modulated. In this case, costs are only proportional to the length of the route.

SDH and DWDM cost per kilometer are represented in Figure 2.3 in red and blue, respectively. The green line represents the minimum cost per kilometer for a given speed. This is the model that we will assume for costs in the Transport Network.

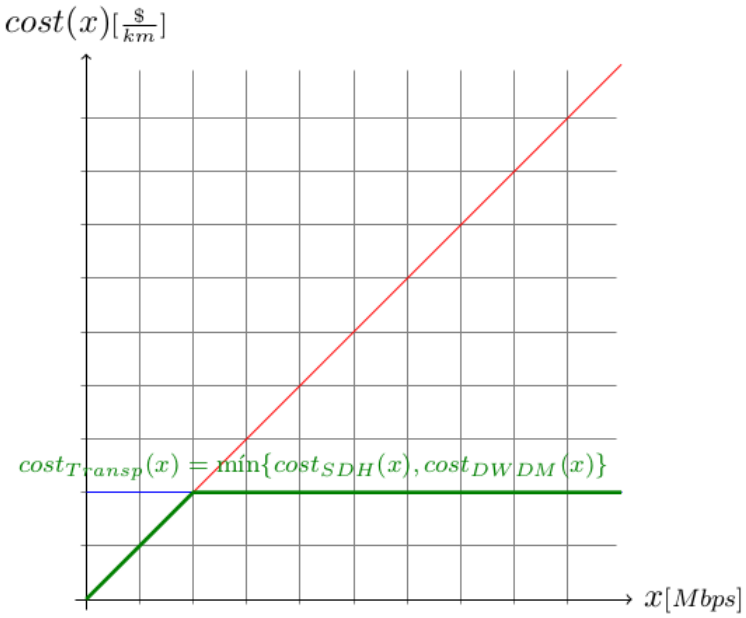

Figure 2.3 – Cost per transport technology.

Definition 2.3.3 *We will consider $\hat{B} = \{\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_B\}$ as the set of capacities supported by the connections of the Transport Network. By definition we will suppose that $\hat{b}_0 = 0$.*

Definition 2.3.4 *The cost function, $T : \hat{B} \to \mathbb{R}$, maps each capacity of $\hat{B}$ to the corresponding cost.*

Definition 2.3.5 *The distance function, $r : E_T \to \mathbb{R}$, maps each edge $e_t \in E_T$ to its length (say, in kilometers). We extend r to paths over G_T by defining $r(\rho_T^{ij})$ as the sum of $r(e_t)$ over all edges e_t of the path ρ_T^{ij}.*

Definition 2.3.6 *The cost in the Transport Network of a flow over a path ρ_T^{ij} with bandwidth b_{ij} is $cost(\rho_T^{ij}, b_{ij}) = r(\rho_T^{ij}) \times T(b_{ij})$.*

2.3.4 Installation Cost

As part of the construction of a routing strategy we allow, under a budget constrain, the installation of new transport links.

Definition 2.3.7 *Let $\bar{E_T}$ be the set of potential edges to be added to the Transport Network. We have $\bar{E_T} = K_{|V_T|} \setminus E_T$.*

Definition 2.3.8 *Let $\hat{E_T}$ be the expanded set of edges. $\hat{E_T} = E_T \cup \{\bar{e_i}\}$ where $\bar{e}_{i=0\ldots|\bar{E_T}|}$ are the links that are added when expanding the Transport Network.*

We will assume that installation cost of a new transport link depends linearly on its length. This is because most of installation costs can be associated to the cost of the optical fiber and the canalization cost.

Definition 2.3.9 *The installation cost matrix $\hat{C} = \{\hat{c_{ij}}\}_{(i,j)\in\bar{E_T}}$, is a positive-real-cost matrix associated to the arcs of $\bar{E_T} = \{K_{|V_T|} \setminus E_T\}$ that models the cost of installing a link between two different sites of V_T.*

2.3.5 Installation Budget

We will consider that the operator has a budget that can be used to expand its Transport Network aiming to reduce and optimize its routing costs.

Definition 2.3.10 *We will refer to I as the installation budget. It is the budget that the operator can invest on its Transport Network.*

2.4 Traffic Routing

2.4.1 Routing in the Data Network

The problem of routing in the Data Network is to find how to exchange traffic between any pair of nodes. That is, which are the links in the Data Network that we will use to route the traffic. In an MPLS network these links builds a tunnel.

Notation 2.4.1 *We denote by P_D be the set of all possible paths in G_D.*

Definition 2.4.2 *A routing scenario ρ_D is any subset of P_D.*

Definition 2.4.3 *A routing scenario ρ_D is a set of paths over G_D, each connecting a different pair of data nodes, v_i, v_j, with $(v_i, v_j) \in E_D$. We define for $(v_i, v_j) \in E_D$, ρ_D^{ij} as the unique path in ρ_D that connects v_i with v_j, if such a path exists. We write $\rho_D^{ij} = \emptyset$ otherwise.*

Let's the following example illustrate these definitions.

Example: Consider a Data Network as shown in Figure 2.4 where V_D and E_D are the following:

$$V_D = \{v_1, v_2, v_3, v_4, v_5\}.$$
$$E_D = \{(v_1, v_2), (v_1, v_5), (v_2, v_3), (v_2, v_5), (v_3, v_4), (v_3, v_5), (v_4, v_5)\}.$$

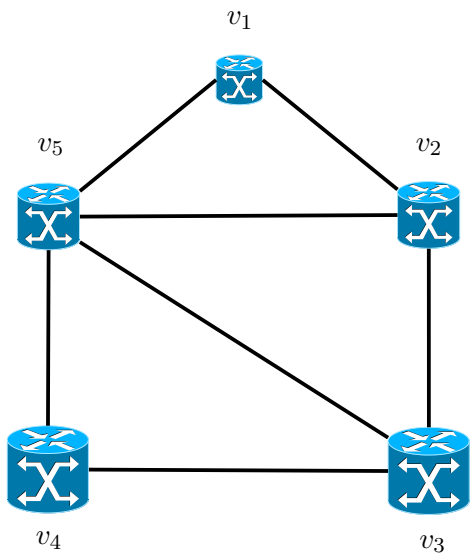

Figure 2.4 – Data Network example.

A possible routing scenario ρ_D is:

$$\rho_D = \{\{(v_1, v_2)\}, \{(v_2, v_3), (v_3, v_5)\}, \{(v_1, v_5), (v_5, v_4)\}, \{(v_4, v_5), (v_5, v_2)\}, \{(v_1, v_2), (v_2, v_3), (v_3, v_4)\}\}.$$

Under this routing scenario the possible routes from v_1 to v_4 are:

$\rho_{D1}^{14} = \{(v_1, v_5), (v_5, v_4)\}$.
$\rho_{D2}^{14} = \{(v_1, v_2), (v_2, v_3), (v_3, v_4)\}$.

These routes are coloured in Figure 2.5.

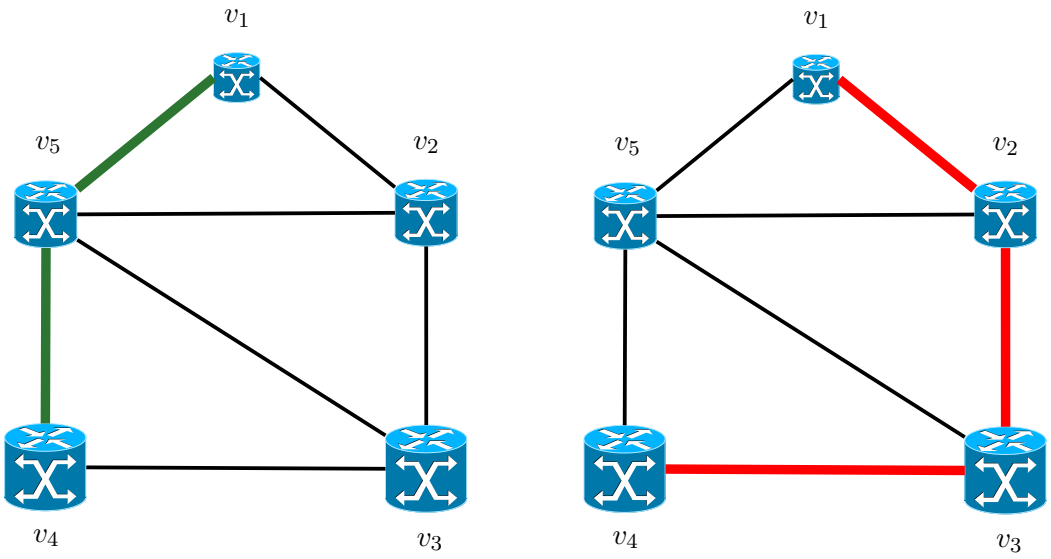

Figure 2.5 – Routes from v_1 to v_4 under ρ_D.

We next relate Transport and Data Network routing through the following definitions.

Definition 2.4.4 *A routing scenario function, $\Phi : (E_T \cup \emptyset) \rightarrow 2^{P_D}$, assigns a routing scenario for each edge $e_t \in E_T$, as well as the empty set. For each $e_t \in E_T$, $\Phi(e_t)$ represents a routing scenario in case of failure of the link represented by e_t. We refer to $\Phi(\emptyset)$ as the nominal routing scenario; which represents a routing scenario in case all transport links are operative.*

Notation 2.4.5 *We denote by $\rho^{ij}_{D_t} \in \Phi(e_t)$ the routing in the Data Network between data nodes v_i and v_j when the transport link represented by e_t is not operative.*

2.4.2 Routing in the Transport Network

The problem of routing in the Transport Network is to set up the links that will provide the transport service to the Data Network.

Notation 2.4.6 *We denote by P_T the set of all possible paths in G_T.*

Definition 2.4.7 *A flow configuration ρ_T is a set of paths over G_T, each connecting a different pair of nodes, t_i, t_j, $(t_i, t_j) \in E_T$. For a data edge $e_d = (v_i, v_j)$, we denote by ρ^{ij}_T the unique path in ρ_T that connects $tns(v_i)$ with $tns(v_j)$, if it exists, and we denote $\rho^{ij}_T = \emptyset$ otherwise.*

Definition 2.4.8 *A flow configuration function, $\Psi(v_i, v_j) = \rho^{ij}_T$, assigns a flow configuration to each edge in the Data Network.*

The following example illustrates these concepts.

Example: Let $G_D = (V_D, E_D)$ and $G_T = (V_T, E_T)$ be a Data and a Transport Network of Figure 2.6.

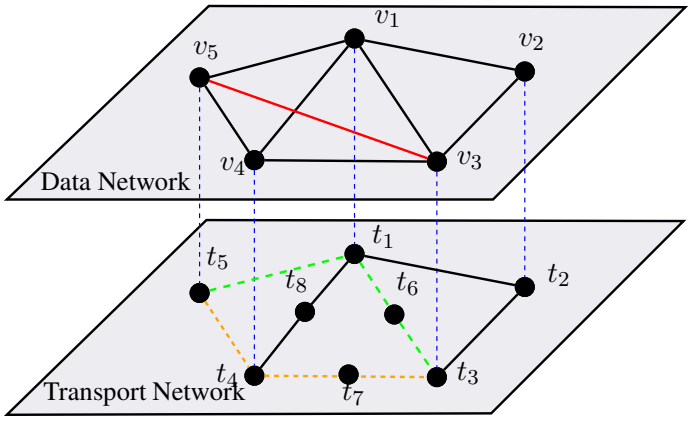

Figure 2.6 – Routing in the Transport Network.

We have:

$V_D = \{v_1, v_2, v_3, v_4, v_5\}$.
$E_D = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_3, v_5), (v_3, v_1), (v_4, v_5), (v_4, v_1), (v_5, v_1)\}$.

$V_T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$.
$E_T = \{(t_1, t_2), (t_2, t_3), (t_3, t_7), (t_3, t_6), (t_7, t_4), (t_4, t_8), (t_4, t_5), (t_5, t_1), (t_6, t_1), (t_8, t_1)\}$.

$tns(v_i) = t_i \quad i = 1 \ldots 5$.

A possible flow configuration is:

$\rho_T = \{\{(t_1, t_6), (t_6, t_3)\}, \{(t_1, t_2)\}, \{(t_1, t_8), (t_8, t_4)\}, \{(t_5, t_1), (t_1, t_6), (t_6, t_3)\}, \{(t_1, t_2), (t_2, t_3)\}, \{(t_5, t_4), (t_4, t_7), (t_7, t_3)\}\}$.

For example, under this flow configuration, ρ_T^{53} can be either $\{(t_5, t_1), (t_1, t_6), (t_6, t_3)\}$ or $\{(t_5, t_4), (t_4, t_7), (t_7, t_3)\}$.

Chapter 3

Formal Definition of the Problem

3.1 Structure of the Problem

The structure of the problem is the following:

Inputs:

1. **Data Network :**
   - $\mathbf{G_D} = (\mathbf{V_D}, \mathbf{E_D})$: Data Network graph.
   - $\bar{\mathbf{M}}$: Traffic matrix.
2. **Transport Network :**
   - $\mathbf{G_T} = (\mathbf{V_T}, \mathbf{E_T})$: Transport Network graph.
   - $\hat{\mathbf{B}}$: Available capacities.
   - $\mathbf{T}$: Technology cost function.
   - $\mathbf{r}$: Distance function.
   - $\mathbf{tns}$: Transport Network station function.
   - $\hat{\mathbf{C}}$: Installation costs matrix.
   - $\mathbf{I}$: Installation budget.

Outputs:

1. $\hat{\mathbf{G}}_\mathbf{T} = (\mathbf{V_T}, \hat{\mathbf{E}}_\mathbf{T})$: The expanded Transport Network.
2. $\mathbf{B} = \{\mathbf{b_{ij}}\}_{\mathbf{(i,j)} \in \mathbf{E_D}}$: Capacities assigned to the data links.
3. $\mathbf{\Phi} = \{\rho^{\mathbf{ij}}_{\mathbf{D_t}}\}_{\mathbf{i,j} \in \mathbf{V_D}, \mathbf{t} \in \{\emptyset \cup \hat{\mathbf{E_T}}\}}$: Data routes for each failure scenario as well as the nominal scenario.
4. $\mathbf{\Psi} = \{\rho^{\mathbf{ij}}_{\mathbf{T}}\}_{\mathbf{(i,j)} \in \mathbf{E_D}, \mathbf{b_{ij}} \neq \mathbf{0}}$: Transport routes for each data link in which a capacity is assigned.

As we stated before a major goal of this problem is to minimize the cost of the solution. The cost of a candidate solution, given by the outputs $\{\hat{G_T}, B, \Phi, \Psi\}$, is defined as:

$$\mathbf{Cost(\hat{G_T}, B, \Phi, \Psi) = \sum_{e=(v_i,v_j)\in E_D} r(\rho_T^{ij}) T(b_e)} \tag{3.1}$$

3.2 Formal Definition of the Problem

We refer to MOBCRN as the following optimization problem:

$$\begin{cases} \min\limits_{(\hat{G_T},B,\Phi,\Psi)} \quad Cost(\hat{G_T}, B, \Phi, \Psi) & & (3.2) \\ \rho_T^{ij} \neq \emptyset & \forall e\in E_D, e=(v_i,v_j), b_{ij}\neq 0, \rho_T=\Psi(e). & (3.3) \\ \rho_{D_t}^{ij} \neq \emptyset & \forall t\in\{\emptyset\cup\hat{E_T}\}, \bar{m}_{ij}\neq 0, \rho_{D_t}\in\Phi(t). & (3.4) \\ |\rho_{D_t}^{pq} \cap (\bigcup\limits_{\substack{e=(v_i,v_j)\\ e\in E_D \\ |\rho_T^{ij}\cap t|\neq\emptyset}} e)| = 0 & \forall t \in E_T, \forall v_i, v_j, v_p, v_q \in V_D & (3.5) \\ b_{pq} \geq \sum\limits_{v_i,v_j\in V_D} (\bar{m}_{ij} | \rho_{D_t}^{ij} \cap e|) & \forall e\in E_D, e=(v_p,v_q), \forall t\in\{\emptyset\cup\hat{E_T}\}, \bar{m}_{pq}\neq 0, \rho_{D_t}=\Phi(t). & (3.6) \\ \sum\limits_{(v_i,v_j)\in\hat{E_T}\setminus E_T} \hat{c_{ij}} \leq I & & (3.7) \end{cases}$$

The meaning of each component of the problem is the following:

Goal:

▲ Equation 3.2 is the one to be optimized. We seek to find three functions (B, Φ, Ψ) that minimize the summatory of transport cost for each data edge $e \in E_D$ that has an associated flow. On each case we add the product of the route of the flow $r(\rho_{ij})$ times the corresponding cost per kilometer of the capacity that was selected $T(b_{ij})$.

Constraints:

▼ Equation 3.3 indicates that each edge $e \in E_D$ that belongs to the solution ($b_{ij} \neq 0$) must have an associated flow in G_T.

▼ Equation 3.4 indicates that on each failure scenario as well as on the nominal scenario ($t \in \{\emptyset \cup E_T\}$) the correspondent routing scenario ($\rho_{D_t} = \Phi(t)$) must include a tunnel between any pair of data nodes: v_i, $v_j \in V_D$ that have a traffic demand between them.

▼ Equation 3.5 shows that for every transport link failure scenario ($\forall t \in E_T$), every tunnel within the Data Network for this state ($\rho_{D_t}^{pq}$), must avoid the usage of affected links (links $e \in E_D$, $e = (v_i, v_j)$ where $|\rho_T^{ij} \cap t| \neq \emptyset$).

▼ Equation 3.6 shows that the edge that connects a pair of data nodes $e = (v_p, v_q) \in E_D$, v_p, $v_q \in V_D$ must have enough capacity to cope with the summatory of the traffic demands that are routed through that edge. The idea is that when a tunnel for a given scenario ($\rho^{ij}_{D_t}$) is routed through edge e, we assign to it the correspondent traffic demands.
▼ Finally, equation 3.7 means that the sum of the installation costs of the edges that are installed ($\bar{e} \in \{\hat{E_T} \setminus E_T\}$) must be lower than the installation budget (I).

Summarizing, the problem consists of finding:

- A set of transport edges $\hat{E_T} \supseteq E_T$ such that the cost of installing the new transport edges, if any, must not exceed the installation budget I.
- A simple route in $\hat{G_T}$ for each "effective edge" of G_D. An effective edge is an edge of the Data Network that has been included in any data routing scenario, ρ_{D_t}, $t \in \{\emptyset \cup \hat{E_T}\}$.
- Transport link failure routing scenarios.
- A data tunnel for each couple of data nodes that have a traffic demand on every Transport Network scenario (nominal or single-failure).
- A data-capacity dimensioning in E_D that guarantees the demand requirements for the selected tunnel configuration.
- All of the above must be achieved while minimizing the routing cost in $\hat{G_T}$.

Part III

ALGORITHM AND RELEVANT PROPERTIES

Chapter 4

An Algorithm for the MOBCRN

4.1 Design Alternatives

In this section we explain how we address the MOBCRN problem. As we stated in Section 1.1, similar problems have been addressed using either meta-heuristics [38] or integer programming models [32].

In our case, we developed an ad-hoc heuristic to obtain an approximate solution. If we consider the case in which there is not any installation budget to consider, $I = 0$, then the MOBCRN problem is the same as the one described in [32]. As [32] is known to be **NP-Hard**, then we can conclude that the MOBCRN is also **NP-Hard**. This means that obtaining an exact solution may be infeasible for moderately large problem instances. At this point it is important to note that it is possible that the algorithm is unable to find a feasible solution, even if it exists. This is deeply explained in Section 4.2.3. However for the test cases that were proposed for this study this was not the case, as showed in Chapter 6. There exist a tradeoff between time and memory consumption and the quality of the approximate solution. This tradeoff will be studied in Chapter 6.

4.2 Algorithm Description

This section explains the algorithm that finds a solution for the MOBCRN problem. First, we describe the algorithm at a high level so the reader can understand it roughly from an end-to-end perspective. After that, we describe its structure step-by-step explaining its most important aspects.

4.2.1 High Level Description

In order to find a solution for the MOBCRN problem the algorithm proceeds as follows:

- **Step 1:** For each data flow, define its routing on the Transport Network, creating a Transport Network path.

- **Step 2:** After all data flows are routed in the Transport Network, build a graph Cz composed by the edges that are used to create the paths of the previous point. Note that Cz is not necessarily a connected graph. During this step the algorithm also identifies the connected subgraphs on Cz, cz_h.
- **Step 3:** For each connected subgraph cz_h in the Transport Network add as many edges as needed to transform cz_h into a 2-connected subgraph, $2 - cz_h$. Note that we build 2-connected subgraphs because the algorithm must route the data flows under a single-link failure scenario on the Transport Network . In order to route that data flow in the Transport Network for each single-link failure on the Transport Network we work with 2-connected subgraphs on it. Once we finish, we obtain the graph $2Cz$ which is the union of the 2-connected subgraphs, that is $2Cz = \bigcup_h cz_h$.
- **Step 4:** Simulate a single-failure link scenario for each edge t in $2Cz$ and re-route all data flows that were routed in the Transport Network using that edge. Once we finish, we have the routing in the Transport Network not only for the nominal scenario but also for each single-link failure scenario.
- **Step 5:** For each Transport Network path, find a routing in the Data Network. Once this point is concluded we obtain the routing in the Data Network for the nominal routing scenario as well as for each single-failure link scenario on the Transport Network.

Figure 4.1 represents a flow chart that describes how each step connects to each other.

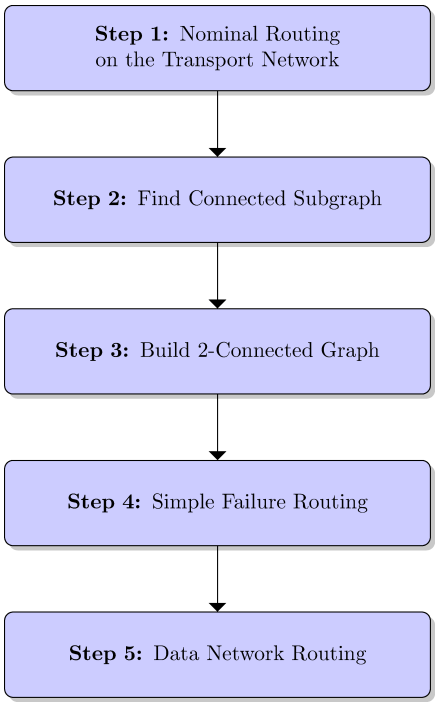

Figure 4.1 – Flow chart of the implementation of the MOBCRN.

Now that we have described the algorithm from a high level perspective we proceed to explain each step in detail.

4.2.2 Low Level Description

4.2.2.1 Step 1: Nominal Routing on the Transport Network

This part of the algorithm finds a nominal routing scenario on the Transport Network for each data flow defined on the Data Network. In order to route the data flows in the Transport Network the algorithm must find the *terminal nodes* of the Transport Network. A terminal node is just a Transport Network node that generates o terminates a data flow in the Transport Network.

Traffic flows are generated and terminated at the Data Network. However, the Data Network uses the services provided by the Transport Network to carry the information from one side to another. In order to achieve that, Data Network nodes that generates traffic delivers that traffic to a Transport Network node. That node is, from the Transport Network perspective, the one that generates the traffic, so it is a terminal node. In the same way the last transport node delivers the data to the Data Network node which effectively terminates the flow. However, that transport node is, from the Transport Network perspective, the one that terminates the traffic, so it is also a terminal node. Terminal and data nodes that generates and terminates traffic are related by the tns function. An example of this situation is showed in Figure 4.2.

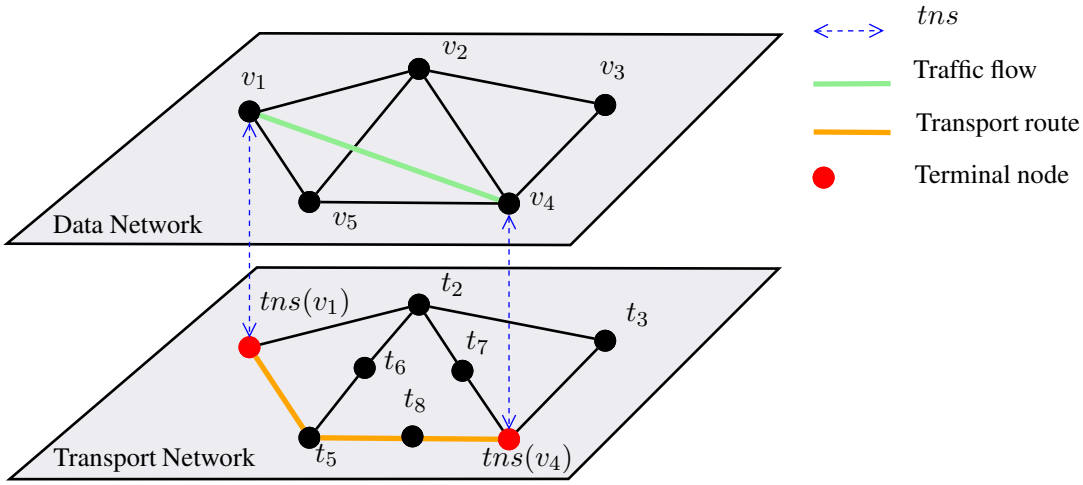

Figure 4.2 – Transport Network with terminal nodes.

After we have identified the terminal nodes we proceed to find a nominal routing scenario. This is, to select the Transport Network links that are going to be used to route the traffic flows when all transport links are active. The following aspects must be considered in order to select these links:

Length of the routes. As we explained before, the routing cost depends linearly on the length of the routes. Thus, finding the shortest path for each route is critical in order to achieve the goal of minimizing the routing cost. In order to achieve this the algorithm not only seeks for the shortest path on the existing transport links but also searches for the possibility of installing additional transport links. If the shortest path is achieved installing additional transport links and the cost of installing them fits within the available installation budget I, then these new links are added to the network. Otherwise, the shortest path is searched within the original Transport Network. The set of nodes and edges that builds the shortest path for each traffic flow will be called the *Minimum Cost Transport Network, (MCTN)*. Our shortest path algorithm (hereafter SPA), based on Dijkstra's implementation [19], is described in detail in section 4.2.2.6.

This scenario adds some complexity to the problem. In case that the MCTN is achieved installing additional links whose total installation cost exceeds the installation budget, the algorithm must decide which links to drop from that network in order to achieve the minimum cost constrained according to I.

In addition to that, the algorithm must take special care of the possibility of *re-using links* for more than one traffic flow. This is because of how the Transport Network works. Once we have assigned a capacity on a transport link that capacity is statically reserved on the link, no matter the portion of the capacity that is effectively used by the traffic flow. This scenario is illustrated in the following example.

Example: Consider a part of a Transport Network as shown in Figure 4.3.

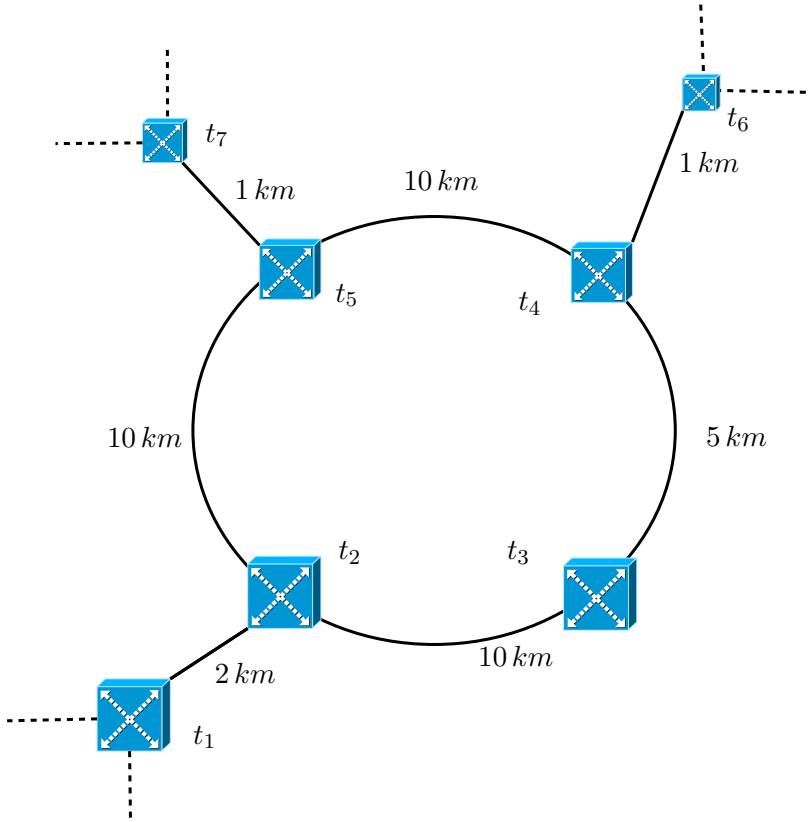

Figure 4.3 – Transport Network of the example .

Suppose that $V_T = \{t_1, \ldots, t_7\}$, $tns(v_i) = t_i$, $i = 1 \ldots, 7$ and we have the following traffic demand matrix:

- $m_{16} = 20Mbps$
- $m_{17} = 40Mbps$

The shortest path for each flow are the following:

- $\rho_T^{16} = \{(t_1, t_2), (t_2, t_3), (t_3, t_4), (t_4, t_6)\}$
- $\rho_T^{17} = \{(t_1, t_2), (t_2, t_5), (t_5, t_7)\}$

Where ρ_T^{16} is 18 km long, and $\rho_T^{17}$is 13 km long.

Assume that we have two different capacities we can use to transport the data: $\hat{B} = \{\hat{b}_1, \hat{b}_2\} = \{48, 150\}$ and that the cost depends linearly on its transport capacity.

The assigned capacity for the data links B is:

- $b_{12} = 150$
- $b_{23} = 48$
- $b_{34} = 48$
- $b_{46} = 48$
- $b_{25} = 48$
- $b_{57} = 48$

The technology cost function T is:

- $T(\hat{b}_1) = 10$
- $T(\hat{b}_2) = 30$

Then we have the following routing costs:

- $cost(\rho_T^{16}, T(B)) = 30 \times 2 + 10 \times (10 + 5 + 1) = 220$
- $cost(\rho_T^{17}, T(B)) = 30 \times 2 + 10 \times (10 + 1) = 170$

Despite we have that $cost(\rho_T^{16}, T(B)) = 220$ and $cost(\rho_T^{17}, T(B)) = 170$ the total cost is not 390. This is because the first link that is used in the routing, $\{t_1, t_2\}$, is shared between both flows. As the link is used by the first flow, the cost of using that link for the second flow is 0. Thus the total cost for the routing is 330.

Number of edges to be used. In our problem, we must find a routing for each single-link failure scenario. Thus, the less is the number of edges selected in the nominal routing scenario the less is the number of failure scenarios that will need to be considered. Moreover, since the larger the number of edges the larger the risk of experiencing a failure, the number of edges used in the nominal scenario also impacts in the robustness of the network. For these two reasons, in case that the algorithm finds more than one routing path with the same cost it will select the one that contains the less number of edges.

The pseudo-code of the algorithm that determines the nominal routing is showed below. Its inputs are the traffic demand matrix, M, the installation budget, I, the distance matrix, D, the installation cost matrix, C and the expanded Transport Network $\hat{G}_T$. Its outputs are the transport routes, Ψ and the total cost of the nominal routing scenario, identified by the variable $Cost$.

Pseudo-code 4.1 Nominal Routing.

Inputs: M, G_T, I, D, C
Outputs: $\Psi, Cost, \hat{G}_T$

$\delta = 0$; // A variable to measure how many budget is remaining throughout the process.
repeat
  for $(u, v \in, E_D)$ **do**
    if $M(u, v) \neq 0$ **then**
      $src = tns(u)$; // If $M(u, v)$ is not zero it is because there is a traffic demand from u to v, so we find the transport node that originates the data flow.
      $dst = tns(v)$; // If $M(u, v)$ is not zero it is because there is a traffic demand from u to v, so we find the transport node that terminates the data flow.
      $[sp(i), spcost(i), \hat{G}_T, Inv] = SPA[u, v, G_T, D, C, I]$; // We call SPA. Its inputs are the pair of nodes for which we would like to find the shortest path, u and v, the Transport Network G_T, the distance matrix, D, the const installation matrix, C and the installation budget I. Its outputs are the shortest path between u and v, the array sp the cost associated to that path, $spcost$, the expanded Transport Network $\hat{G}_T$ and the remaining budget Inv.
    end if
  end for
  for $(i = 1, \ldots, length(sp))$ **do**
    Find links $e_j \in sp$ that are shared by more than one path, if any.
    if $(e_j \cap sp(i) \neq \emptyset)$ **then**
      Save $sp(i)$ in ρ_T;
      $Cost = Cost + spcost(i)$; // Find the total cost of this routing scenario.
    end if
  end for
  $\delta = |Cost - \delta|$;
until $\delta = 0$;

4.2.2.2 Step 2: Find Connected Subgraph

In this step we build a connected graph Cz composed by the edges that are used to create the paths of the previous point. As Cz is not necessarily a connected graph we also identify its connected subgraphs, cz_h.

Consider a nominal routing scenario, ρ_T in the expanded graph $\hat{G}_T$.

Definition 4.2.1 *We define the graph Cz as $Cz = \{ \bigcup_{ij,\, i,j \in V_D} \rho_T^{ij} \} \subseteq \hat{G}_T$. We will identify on*

Cz its 2-connex components and refer to its n-th component as cz_n.

This step of the algorithm not only identifies the connected subgraphs but also its degree-1 nodes.

Definition 4.2.2 *A transport node t of a connected subgraph cz_n is a degree-1 node if it satisfies the following conditions:*

1. *t is either $tns(i)$ or $tns(j)$, a terminal node of a transport routing ρ_T^{ij}, such that $\rho_T^{ij} \subseteq cz_n$; $i, j \in V_D$.*
2. *There is not another transport routing $\rho_T^{kl} \subseteq cz_n$; $k, l \in V_D$ such that t is not a terminal node and $\{t \cap \rho_T^{kl}\} \neq \emptyset$.*

Let's review the following example in order to clarify these ideas.

Example: Consider the Transport Network and the nominal routing scenario shown in Figure 4.4 asuming $t_i = tns(v_i)$ with $i = 1 \ldots 14$. $t_i \in V_T$, $v_i \in V_D$.

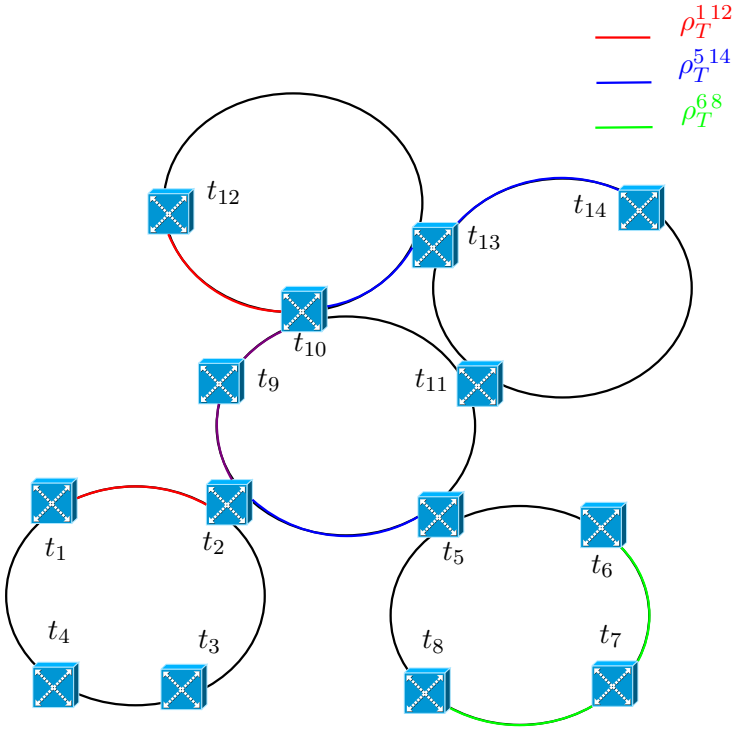

Figure 4.4 – Transport Network of the Example.

We have the following transport routes:
- $\rho_T^{1\,12} = \{(t_1, t_2); (t_2, t_9); (t_9, t_{10}); (t_{10}, t_{12})\}$
- $\rho_T^{5\,14} = \{(t_5, t_2); (t_2, t_9); (t_9, t_{10}); (t_{10}, t_{13}); (t_{13}, t_{14})\}$
- $\rho_T^{6\,8} = \{(t_6, t_7); (t_7, t_8)\}$

In this scenario we have the following:
- $\rho_T^{1\,12} \cap \rho_T^{5\,14} = \{(t_2, t_9); (t_9, t_{10})\}$
- $\rho_T^{1\,12} \cap \rho_T^{6\,8} = \{\emptyset\}$
- $\rho_T^{5\,14} \cap \rho_T^{6\,8} = \{\emptyset\}$

So we have the following connected subgraphs:
- $cz_1 = \{\rho_T^{1\,12} \cup \rho_T^{5\,14}\} = \{(t_1, t_2); (t_2, t_9); (t_9, t_{10}); (t_{10}, t_{12}); (t_5, t_2); (t_{10}, t_{13}); (t_{13}, t_{14})\}$
- $cz_2 = \{\rho_T^{6\,8}\} = \{(t_6, t_7); (t_7, t_8)\}$

So $Cz = \{cz_1, cz_2\}$ and we have the following set of degree-1 nodes: $\{t_1, t_5, t_6, t_8, t_{12}, t_{14}\}$. This can be observed in Figure 4.5.

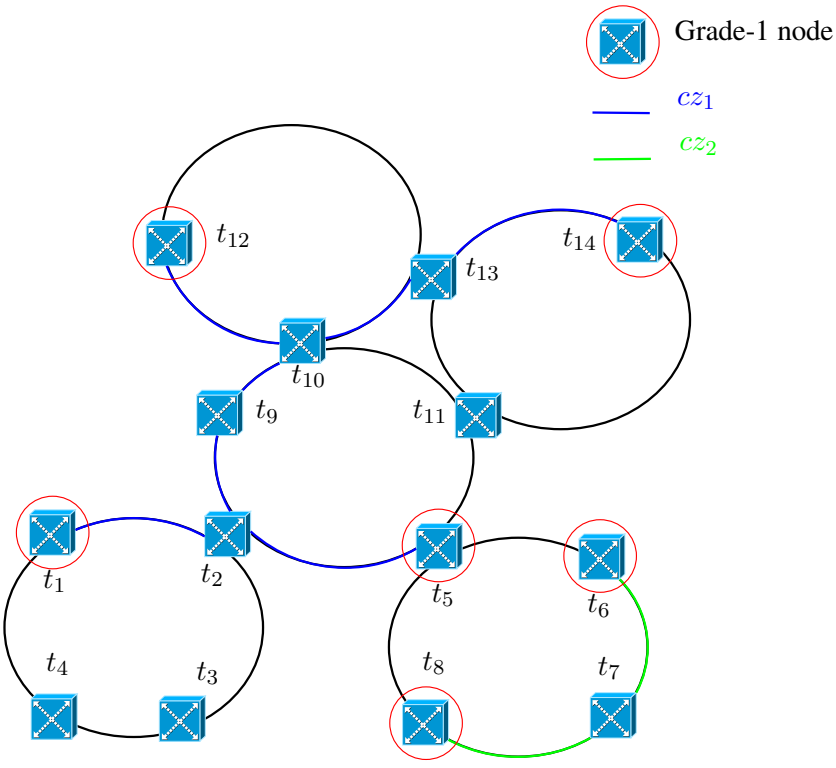

Figure 4.5 – Connected subgraphs of the Example.

This part of the algorithm is implemented based in deep-first search (DFS) algorithm [44].

4.2.2.3 Step 3: Build 2-Connected Graph

Once the algorithm has identified the connected subgraphs, the next step is to build the 2-connected graphs. As we explained in the previous subsection the connected subgraphs identified are delimited by degree-1 nodes. A 2-connected graph is a connected subgraph in $\hat{G}_T$ whose nodes are connected at least by two edge-disjoint paths.

This step takes as inputs the connected subgraphs and its degree-1 nodes and adds links to them in order to increase its connectivity degree at least by one.

One of the requirements of the problem is that the algorithm must find routes to every single failure scenario. Having degree-1 nodes in our solution will not satisfy that requirement. If the link that connects a degree-1 node to the rest of the network fails, then it will be isolated from the rest of the Transport Network so the incoming or outgoing traffic from that node will not be able to be routed.

Considering this, the goal of this step is to build 2-connected graphs so in case of a single failure scenario all the traffic routed within that connected subgraph can still be routed there.

For each degree-1 node t this step runs the SPA for t and any other transport node that belongs to a connected subgraph. In order to avoid selecting links that are already on the connected subgraphs these links are tagged with ∞ distance. When the algorithm finds the shortest path between t and any other transport node that belongs to a connected subgraph it saves the one that has shortest length.

Once the algorithm iterates over all degree-1 nodes the 2-connected graphs are built. The following example illustrates this.

Example: Consider the connected subgraphs of the previous example, shown in Figure 4.5.

We have the following degree-1 nodes:

- $\{t_1, t_5, t_6, t_8, t_{12}, t_{14}\}$

So the algorithm iterates over them running the SPA to increase its degree and build 2-connected graphs. Figure 4.6 shows a possible 2-connected network of this example.

In this case we have two 2-connex graphs:

- $2 - cz_1 = cz_1 \cup \{(t_1, t_4); (t_4, t_3); (t_3, t_2); (t_{12}, t_{13}); (t_{11}, t_{14})\}$
- $2 - cz_2 = cz_2 \cup \{(t_5, t_6); (t_5, t_8)\}$

Pseudo-code 4.2 shows the pseudo-code of the implementation of this step. The inputs for this step are the graph Cz, a matrix that contains the degree-1 nodes of Cz, $deg1Nodes$, the expanded Transport Network, $\hat{G}_T$, the distance matrix D, the cost matrix C, and the installation budget I. On the other hand its outputs is the 2-Connected Graph $2Cz$.

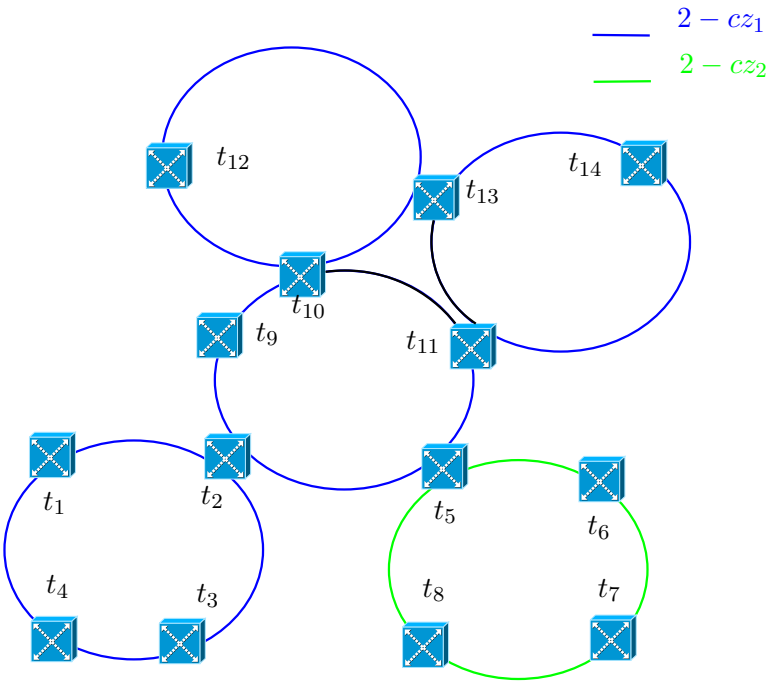

Figure 4.6 – 2-connected graphs of the Example.

Pseudo-code 4.2 Build 2-Connected Graph.

Inputs: Cz, $deg1Nodes$, $\hat{G}_T$, D, C, I.
Outputs: $2Cz$

1: **for** (All edges e in Cz) **do**
2: $C(e) = \infty$; // Set the cost of the edges of the connected subgraphs to ∞ so the SPA would not run over them.
3: **end for**
4: **for** (Each connected subgraph cz_h in Cz) **do**
5: **for** (Each degree-1 node i in cz_h) **do**
6: **for** (Each node j in cz_h) **do**
7: $[sp(i), spcost(i), \hat{G}_T, Inv] = SPA[i, j, G_T, D, C, I]$;
// For each degree-1 node call the SPA to find the shortest path to every node j in cz_h.
8: **end for**
9: Add $\min(sp(i))$ to the connected subgraph to build the 2-Connected Graph, $2Cz$.
10: **end for**
11: **end for**

In order to construct the 2-connex graph we could also have used algotithms such as [15, 43, 45]. However, our solution using the SPA performed accordingly when we executed the algorithm in the scenarios we run.

4.2.2.4 Step 4: Simple Failure Routing

An important criteria considered to design the Transport Network is that it must be tolerant to single-link failures. In this step we simulate a single-link failure for each link that is part of the 2-connected graph. In order to simulate the failure, the algorithm sets the length of the link to infinite. After that, it looks for the data flows that are transported by that link and re-route them within the 2-connected graph using the SPA.

Once the iteration over all links that belongs to the 2-connected graph the simulation is finished, having all the single-link failure routing scenarios.

The following example illustrates this.

Example: Consider the 2-connected graph of the previous example and the following information

Suppose we have the following traffic demand matrix:

- $m_{1\,12} = 200\,Mbps$
- $m_{3\,10} = 600\,Mbps$

Suppose that given the technology and link-length information the shortest path given by the SPA for each flow are the following:

- $\rho_T^{1\,12} = \{(t_1, t_2); (t_2, t_9); (t_9, t_{10}); (t_{10}, t_{12})\}$
- $\rho_T^{3\,10} = \{(t_3, t_2); (t_2, t_9); (t_9, t_{10})\}$

These routes are showed in dotted lines in Figure 4.7.

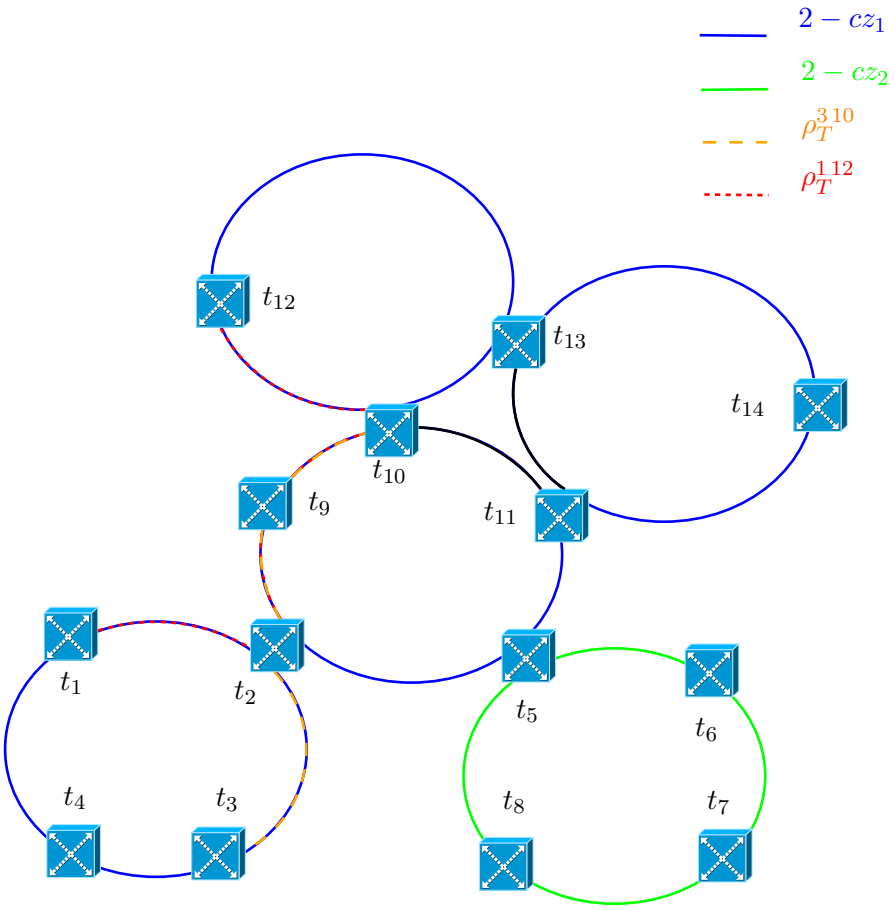

Figure 4.7 – Transport routes in the 2-connected graph.

Now suppose that link $e = (t_2, t_9)$ fails. In that case both routes are affected so they must be re-routed to transport the data flows in that scenario. A possible routing in that failure scenario is the following:

- $\rho_{Te}^{1\,12} = \{(t_1, t_2); (t_2, t_5); (t_5, t_{11}); (t_{11}, t_{14}); (t_{14}, t_{13}); (t_{13}, t_{10}); (t_{10}, t_{12})\}$
- $\rho_{Te}^{3\,10} = \{(t_3, t_2); (t_2, t_5); (t_5, t_{11}); (t_{11}, t_{14}); (t_{14}, t_{13}); (t_{13}, t_{10})\}$

This is illustrated in Figure 4.8.

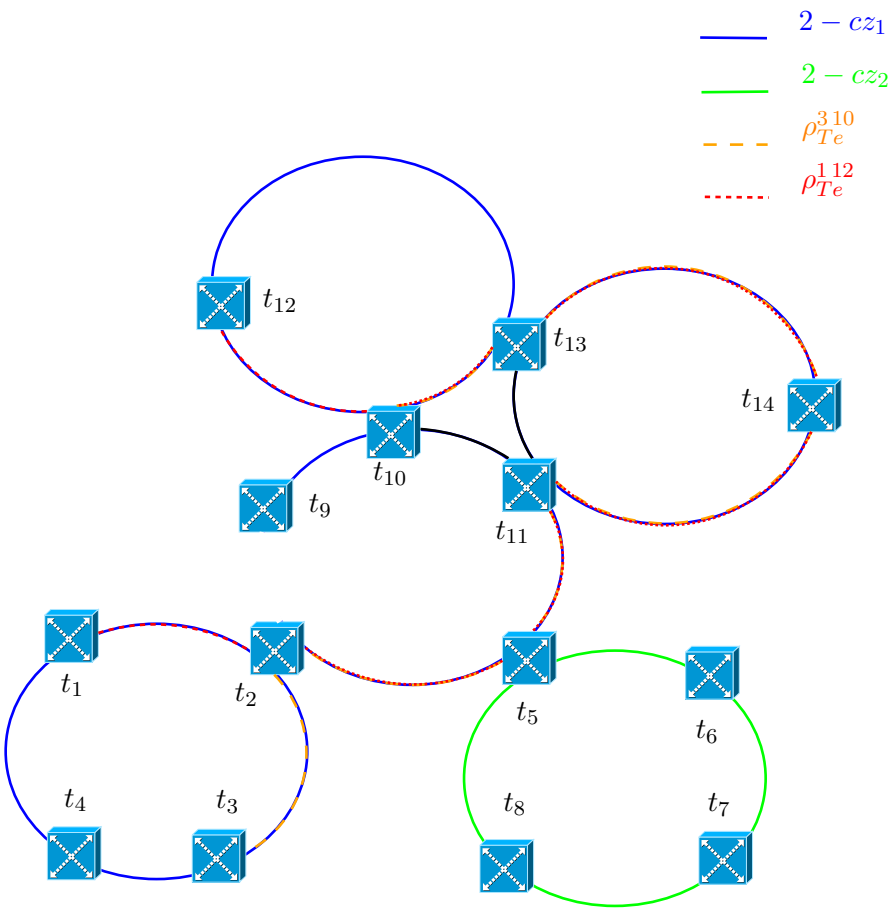

Figure 4.8 – Transport routes in the 2-connected graph when link $e = (t_2, t_9)$ fails.

The pseudo-code of this step is showed below. Its inputs are the 2-Connected graph $2Cz$ and the traffic demand matrix M while its output is the routing for each transport edge e that fails, ρ_{Te}^{ij}.

Pseudo-code 4.3 Simple Failure Routing.

Inputs: $2Cz$, M
Outputs: ρ_{Te}^{ij}

1: **for** (Each link in $2Cz$) **do**
2: $C(e) = \infty$; // Set the cost of the edge of the 2-connected graph to ∞ to simulate the failure so the SPA would not run over them.
3: **for** $(u, v \in, E_D)$ **do**
4: **if** $M(u, v) \neq 0$ AND $\{e \cap \rho_T^{uv}\} \neq \emptyset$ **then**
5: $src = tns(u)$; // If $M(u, v)$ is not zero it is because there is a traffic demand from u to v, so we find the transport node that originates the data flow.
6: $dst = tns(v)$; // If $M(u, v)$ is not zero it is because there is a traffic demand from u to v, so we find the transport node that terminates the data flow.
7: $[sp(i), spcost(i), \hat{G}_T, Inv] = SPA[u, v, G_T, D, C, I]$; // We call SPA. Its inputs are the pair of nodes for which we would like to find the shortest path, u and v, the Transport Network G_T, the distance matrix, D, the const installation matrix, C and the installation budget I. Its outputs are the shortest path between u and v, the array sp the cost associated to that path, $spcost$, the expanded Transport Network $\hat{G}_T$ and the remaining budget Inv.
8: $\rho_{Te}^{uv} = sp(i)$;// Save the new route of the flow for the case that link e fails.
9: **end if**
10: **end for**
11: **end for**

4.2.2.5 Step 5: Data Network Routing

When we conclude with the previous step, we obtain the path that the data flows will follow in the Transport Network not only for the nominal scenaraio but also for all single-link failure scenarios. As we explained in section 3.1 the outputs of the algorithm are the following:

1. $\hat{\mathbf{G}}_{\mathbf{T}} = (\mathbf{V}_{\mathbf{T}}, \hat{\mathbf{E}}_{\mathbf{T}})$: The expanded Transport Network.
2. $\mathbf{B} = \{\mathbf{b_{ij}}\}_{\mathbf{(i,j)} \in \mathbf{E_D}}$: Capacities assigned to the data links.
3. $\mathbf{\Phi} = \{\rho^{\mathbf{ij}}_{\mathbf{D_t}}\}_{\mathbf{i,j} \in \mathbf{V_D}, \mathbf{t} \in \{\emptyset \cup \hat{\mathbf{E_T}}\}}$: Data routes for each failure scenario as well as the nominal scenario.
4. $\mathbf{\Psi} = \{\rho^{\mathbf{ij}}_{\mathbf{T}}\}_{\mathbf{(i,j)} \in \mathbf{E_D}, \mathbf{b_{ij}} \neq \mathbf{0}}$: Transport routes for each data link in which a capacity is assigned.

We also need to calculate the routing cost of the solution:

$$\mathbf{Cost}(\hat{\mathbf{G_T}}, \mathbf{B}, \mathbf{\Phi}, \mathbf{\Psi}) = \sum_{\mathbf{e}=(\mathbf{v_i}, \mathbf{v_j}) \in \mathbf{E_D}} \mathbf{r}(\rho^{\mathbf{ij}}_{\mathbf{T}}) \mathbf{T}(\mathbf{b_e})$$

So we already obtained points 1 and 4. This step will calculate the data routes for all scenarios and the assigned capacities for the data links (points 2 and 3). Once we have all the information we will be able to calculate the routing cost.

In order to find the relation between the Transport Network and Data Network we map transport links to data links. Thus, the algorithm runs the following procedure:

- In case that both transport nodes of the link has a correspondant data node it maps that transport link to the data link that connects them. In the case of Figure 4.9 it maps the links as follows:
  - $(t_1, t_2) \rightarrow (v_1, v_2)$
  - $(t_1, t_3) \rightarrow (v_1, v_3)$
- In case that any of the transport nodes of a link does not have a correspondant data node we do the following:
  - For each traffic flow that is transported by that node we search for its closest neigbours nodes on that route that have a correspondant data node. All the links that the algorithm transited until finding these nodes are mapped to the same data link.

  Taking Figure 4.9 as reference, if we suppose that there is a traffic flow from t_3 to t_2, links (t_3, t_4) and (t_4, t_2) will be mapped to data link (v_3, v_2).

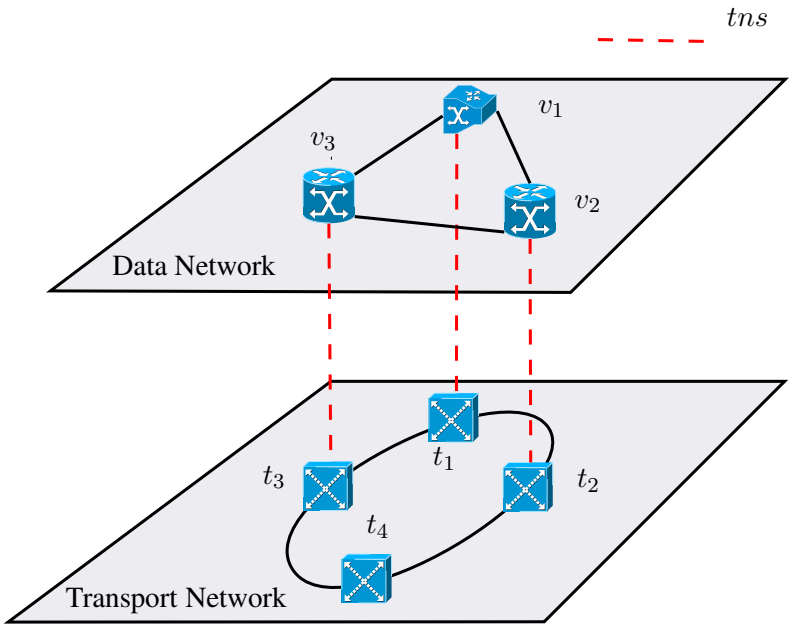

Figure 4.9 – Network of the example.

After doing this, the algorithm calculates the assigned capacities for each data link. This is performed calculating the summatory of the traffic that is being transported by the transport links that are mapped into them. As in each routing scenario the traffic that is being transported by the transport links varies, the algorithm must choose the scenario in which the transport link transports the largest traffic.

The following example illustrates this.

Example: Consider the Data and Transport Networks of Figure 4.10. For the sake of simplicity we consider that there are no transport link faiulres.

Suppose we have the following traffic demand matrix:

- $m_{1\,2} = 900Mbps$
- $m_{2\,3} = 750Mbps$

Suppose that given the technology and link-length information the shortest path for each flow are the following:

- $\rho_T^{1\,2} = \{(t_1, t_5); (t_5, t_2)\}$
- $\rho_T^{2\,3} = \{(t_2, t_6); (t_6, t_3)\}$

These routes are showed in dotted-line in Figure 4.10.

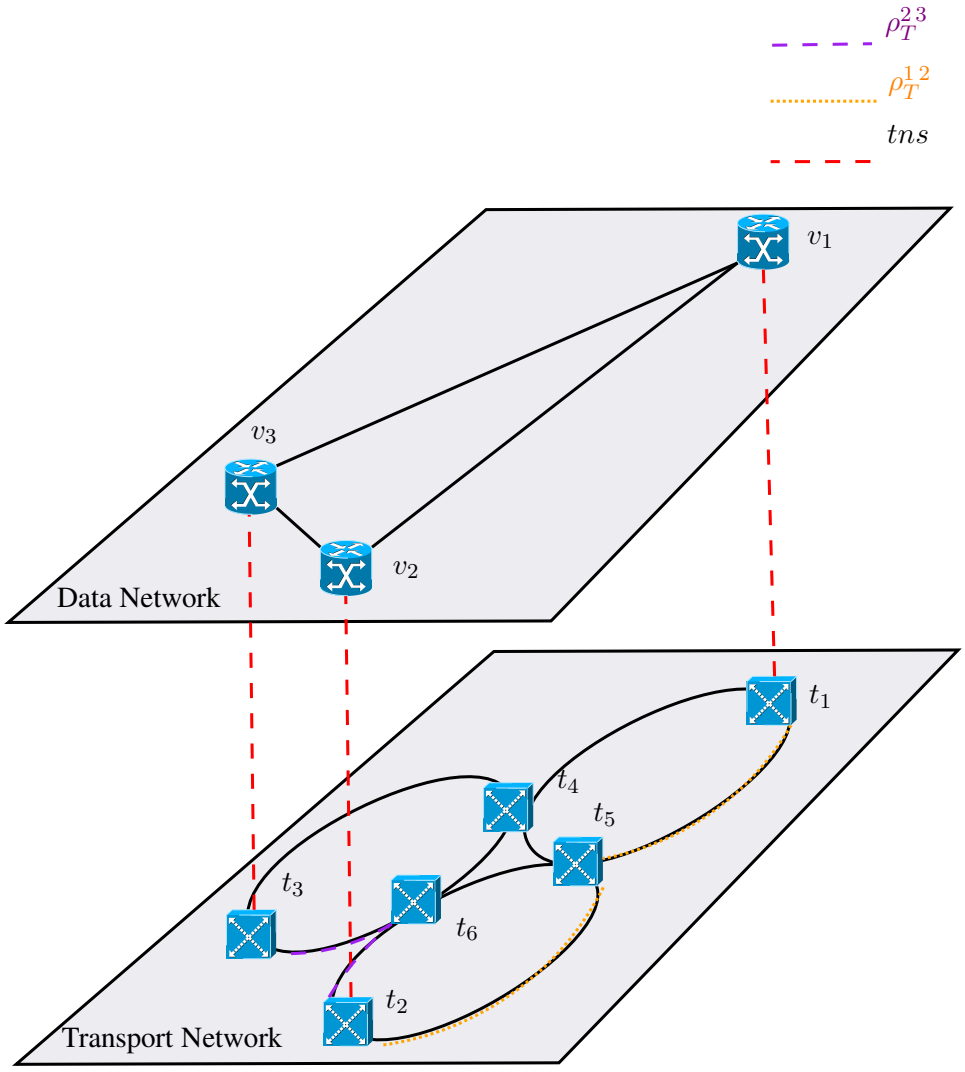

Figure 4.10 – Data and Transport Network of the example.

In this case we have the following mapping:

- $\{(t_1, t_5), (t_5, t_2)\} \rightarrow (v_1, v_2)$
- $\{(t_2, t_6), (t_6, t_3)\} \rightarrow (v_2, v_3)$

Suppose we have the following available capacities to assign in the Data Network:

- $1\,Gbps$
- $10\,Gbps$

In this case as both traffic demands can fit in $1\,Gbps$ both data links are configured as follows:

- $b_{12} = 1\,Gbps$
- $b_{23} = 1\,Gbps$

Suppose that, for any reason, link (v_2, v_3) could not be assigned. In that case the traffic in the Data Network must be transported by data route $\{(v_2, v_1), (v_1, v_3)\}$. In that case link (v_2, v_1) would need to transport both traffic demmads. As $m_{1\,2} + m_{2\,3} > 1\,Gbps$ data links are configured as follows:

- $b_{12} = 10\,Gbps$
- $b_{13} = 1\,Gbps$

Once the algorithm calculated all the outputs it proceeds to find the routing costs as defined in equation 3.1.

The pseudo-code of this step is showed below. The inputs are the 2-Connected Network, $2Cz$ and the traffic demand matrix M. On the other hand the outputs are the capacities assigned to the data links, B, and the data routes for each failure scenario as well as the nominal scenario Φ.

Pseudo-code 4.4 Data Network Routing.

Inputs: $2Cz, M$
Outputs: B, Φ

Find the Data Network links to be used:
for $(t = (t_i, t_j) \in 2Cz)$ **do**
  for $(u, v \in, E_D)$ **do**
    if $M(u, v) \neq 0$ **then**
      Find the closest neighbours of t_i and t_j, n_i and n_j in the flow originated in $tns(u)$ and terminated in $tns(v)$ such that n_i and n_j have v_a and v_b as correspondant nodes in the Data Network respectively;
      Map the path $\{n(i), \ldots, n(j)\}$ to $(v_a, v_b) \in E_D$;
      Set $cap_{ab} = 0$; // Set the initial capacity of the data link to 0.
      if $(v_a, v_b) \notin S$ **then**
        Add (v_a, v_b) to a set of data links S;
      else
        EXIT; // If two neighbours can only be mapped to the same couple of data nodes the algorithm exits as there is not feasible solution. This is explained deeply in Section 4.2.3.
      end if
      Construct $\rho_{D_t}^{uv}$;
      Add $\rho_{D_t}^{uv}$ to Φ;
    end if
  end for
end for
Calculate the capacities for the data links:
for $(e \in S)$ **do**
  for (Each routing scenario) **do**
    if $(\rho_T^{uv} \cap e) \neq \emptyset$ **then**
      Add the capacity of (u, v) to $capTemp(e)$; // $capTemp$ is a vector used to store the capacities temporarily.
    end if
    $cap(e) = \max(cap(e), capTemp(e))$;
    $capTemp(e) = 0$;
  end for
  Set $b(e)$ as the minimum available capacity that transports $cap(e)$;
end for

4.2.2.6 Shortest Path Algorithm

This function implements an heuristic based on Dijkstra's shortest path algorithm [19] that finds the shortest path between two nodes of the Transport Network. Its inputs are the following:

- Source and destination nodes.
- Transport Network G_T.
- Distance matrix D.
- Installation costs matrix C.
- Installation budget I.

The outputs of the SPA are the following:

- The shortest path between source and destination nodes and its associated cost.
- The modified Transport Network $\hat{G}_T$.
- The updated installation budget Inv.

Note that the Transport Network can be either modified or not. This depends on the Transport Network, the length and costs of the potential links to be added and the installation budget. In case that the SPA finds a shortest path adding some new links whose installation cost is lower than the budget then the Transport Network is modified.

The core of the SPA is based on the same idea that uses the well known Djkstra's algorithm [19]. The big difference is that, for each transport node, the SPA searches not only the real neighbours but also all potential ones. A transport node t is a potential neighbour of a transport node v if the installation cost of the link that connects them is smaller than the available installation budget. So, the SPA checks that condition for each node when calculating the shortest path. In case that a link is installed the installation budget is updated to reflect the installation of that link.

The pseudo-code of the SPA is showed below. Its inputs are the pair of nodes for which we would like to find the shortest path, src and dst, the Transport Network G_T, the distance matrix, D, the const installation matrix, C and the installation budget I. Its outputs are the shortest path between src and dst, the array sp, the cost associated to that path, $spcost$, the expanded Transport Network $\hat{G}_T$ and the remaining budget Inv.

Pseudo-code 4.5 Shortest Path Algorithm.

Inputs: src, dst, G_T, D, C, I.
Outputs: sp, $spcost$, $\hat{G}_T$, Inv.

$dist(u) = \infty \quad \forall u \in V_T \setminus \{src\}$; // Set the distance between the $source$ and the rest of the nodes to ∞.
$dist(src) = 0$; // Set the distance from src to src to 0.
$visited(u) = 0 \quad \forall u \in V_T$; // Array of visited nodes.
while $(sum(visited) \neq |V_T|)$ **do**
  $u = \text{argmin}\{dist(i) : i \in V_T,\ visited(i) = 0\}$;
  $visited(u) = 1$; // Mark the node as visited.
  for $(i = 1 : |V_T|)$ **do**
    if $(dist(u) + D(u,i) < dist(i))$ **then**
      if $((u,i) \in \hat{E}_T)$ **then**
        $dist(i) = dist(u) + D(u,i)$; // Update the distance.
        $prev(i) = u$; // Set u as the previous node.
      else if $(C(u,i) \leq I)$ **then**
        $dist(i) = dist(u) + D(u,i)$; // Update the distance.
        $prev(i) = u$; // Set u as the previous node.
        $I = I - C(u,i)$; // Update the budget.
        $\hat{E}_T = \hat{E}_T \cup \{(u,i)\}$; // Add the new edge to G_T.
      end if
    end if
  end for
end while
$sp = [dst]$; // The shortest path array.
while $(sp(1) \neq src)$ **do**
  $sp = [prev(sp(1)),\ sp]$; // Build the shortest path array.
end while
$spCost = dist(dst)$;// Calculate the distance from src to dst.

The following example shows how the SPA works.

Example: Consider the Transport Network of Figure 4.11.

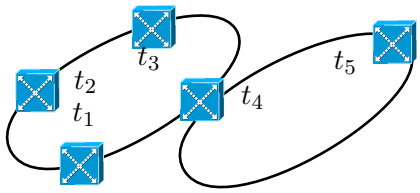

Figure 4.11 – Transport Network of the example.

The inputs for the algotithm are the following:

Distance matrix D:

$$\mathbf{D} = \begin{bmatrix} 0 & 10 & 20 & 10 & 30 \\ 10 & 0 & 10 & 20 & 60 \\ 20 & 10 & 0 & 10 & 15 \\ 10 & 20 & 10 & 0 & 25 \\ 30 & 60 & 15 & 25 & 0 \end{bmatrix}$$

Installation costs matrix C:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 100 & 0 & 300 \\ 0 & 0 & 0 & 500 & 400 \\ 100 & 0 & 0 & 0 & 150 \\ 0 & 500 & 0 & 0 & 0 \\ 300 & 400 & 150 & 0 & 0 \end{bmatrix}$$

Installation budget $I = 350$.

Suppose we must find the shortest path between t_2 and t_5. The SPA will proceed as follows:

Lines 1 to 2 of Pseudo-code 4.5:
Set the distance from t_2 to the rest of the nodes to ∞ and 0 to itself.

Lines 3 to 20 of Pseudo-code 4.5:
Set the distance to each candidate and mark the node t_2 as visited. In case that the link must be installed it can only be updated if its installation cost is lower than the installation budget, $C(t_2, t_i) < I$.

As t_1 is one of the nodes with minimum distance, set $u = t_1$.

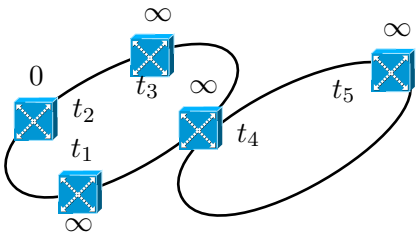

Figure 4.12 – Transport Network of the example - lines 1 to 2 of Pseudo-code 4.5:.

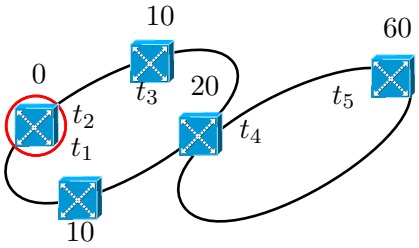

Figure 4.13 – Transport Network of the example.

Update the distances to the rest of *unvisited nodes* t_i as $\min(d(t_2, t_i), d(t_2, t_1) + d(t_1, t_i))$. In case that the link must be installed and the new distance is smaller it can only be updated if its installation cost is lower than the installation budget, $C(t_1, t_i) < I$.

In this case the only updated distance is the one to t_5 as $d(t_2, t_5) = 60 > d(t_2, t_1) + d(t_1, t_5) = 40$. This is possible as the installation cost of that link is lower than the installation budget $C(t_1, t_5) = 300 < I = 350$.

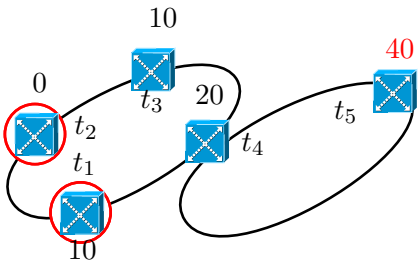

Figure 4.14 – Transport Network of the example.

Repeat lines 4 to 20 calculating the distances from the new visited node, t_1. After running the lines new visited node is t_4. In this case all the distances remain unchanged.

Repeat lines 4 to 20 calculating the distances from the new visited node, t_4. After running the lines new visited node is t_3.

In this case the only updated distance is the one to t_5 as $d(t_2, t_1) + d(t_1, t_5) = 40 > d(t_2, t_3) + d(t_3, t_5) = 25$. This is possible as the installation cost of that link is lower than the installation budget $C(t_3, t_5) = 150 < I = 350$.

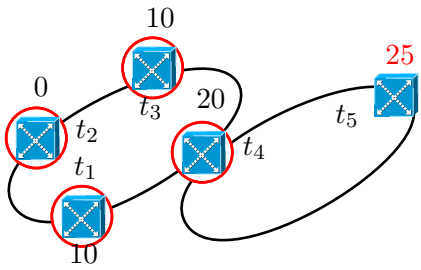

Figure 4.15 – Transport Network of the example.

Lines 21 to 25 of Pseudo-code 4.5:

As there are not any unvisited nodes the algotithm calculates the shortest path and its cost. In this case the shortest path is $sp = [t_2, t_3, t_5]$ with cost 25.

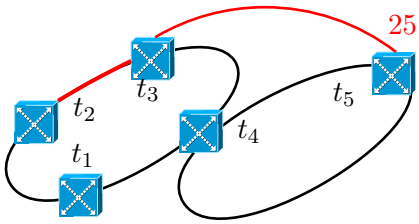

Figure 4.16 – Transport Network of the example with the shortest path found by the SPA.

4.2.3 Unfeasible Scenarios

As we commented previously, there are some scenarios in which our algorithm does not find a feasible solution. These scenarios do not represent a case of interest and of course were not included in the set of cases we studied throughout this work.

Issues could eventually arise when we map the transport routes to data routes in the Step 5 of the algorithm. This happen when two or more different transport routes can only be mapped to the same data route. Our problem is designed in a way that a data route can only be mapped to one and only one transport route (definitions 2.4.7 and 2.4.8). Thus, we can not find an alternative route to transport the data flow in case of a single link failure in the Transport Network. That condition was established in previous works that were done at our research institute [17, 18, 32, 38]. As one of the goals of the present project is to compare our results with the ones obtained in previous projects we kept this as part of the definition of the problem.

However, that condition is not always met in real telecommunication networks. In fact, one of the most imoprtant characteristics in the design of different layers within a telecommunication network is to guarantee certain independence between these layers. Into that context, it is perfectly allowed that a modification on the Transport Network such as a link failure is not noticed in the Data Network. Our algorithm can be easily adapted to support this scenario as well.

A simple example when this occurs is showed in Figure 4.17.

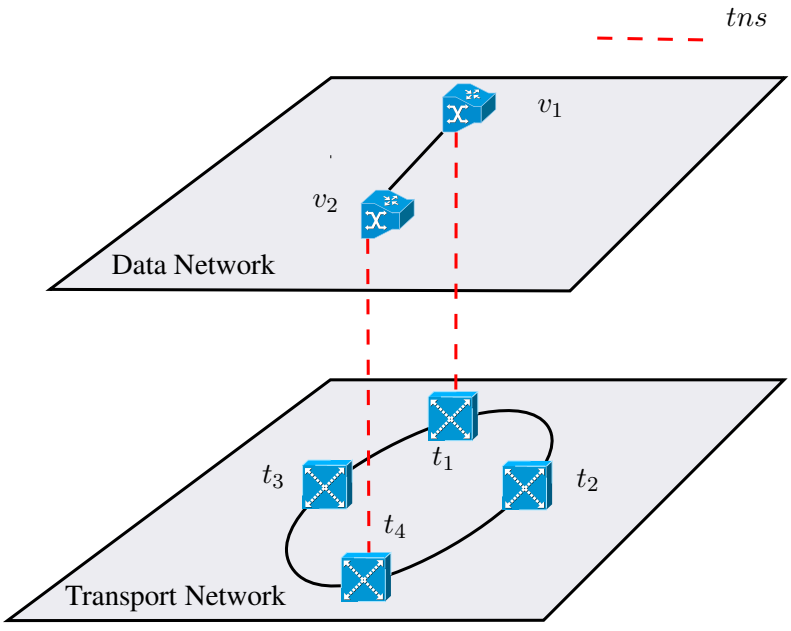

Figure 4.17 – Unfeasible scenario.

In the case of Figure 4.17 we have no choice but to map routes $\{(t_1, t_2); (t_2, t_4)\}$ and $\{(t_1, t_3); (t_3, t_4)\}$ to the same data route, $\{(v_1, v_2)\}$.

Part IV

RESULTS

Chapter 5

Test Cases Description

In this chapter we explain the test cases designed for which we run the algorithm. We divide the test cases into two different sets. The first set of test cases intends to validate the developed algorithm and compare its performance against other techniques to solve this kind of problems.

On the other hand, the second set of test cases are proposed to evaluate all functionalities of the algorithm. This is, the possibility of installing new transport links with some installation budget constrains.

Both test cases are generated using the information provided by ANTEL from their Data and Transport Network. This is an added value of this project as it is extremely interesting to run our algorithm using data from a real network such as ANTEL's. This clearly shows how this kind of techniques can be applied to help solving problems from real networks.

5.1 First Set of Test Cases

The test cases were defined during a project carried out by researchers from the University of the Republic and ANTEL. They describe different situations and realities of ANTEL, considering several parameters that are under their control [38].

We choose this set of test cases because a couple of projects already worked on them [32, 38]. Thus, by adjusting some inputs, it is possible to compare the performance of our algorithm with other ways of solving this kind of problems such as using meta-heuristics [38] or binay integer programming models [32]. This can be easily achieved by setting the installation budget to zero, so there would not be any possibility of installing additional links in the Transport Network.

5.1.1 Problem Data

This section describes the inputs to the problem. As we said previously, all the data was provided by ANTEL:

- The Transport Network, including the nodes, edges and their lengths.
- The available capacities $\hat{B}$ and their cost per kilometer, $T : \hat{B} \to \mathbb{R}_0^+$.

- The Data Network.

The topology of the Transport Network used to solve the problem is showed in Figure 5.1.

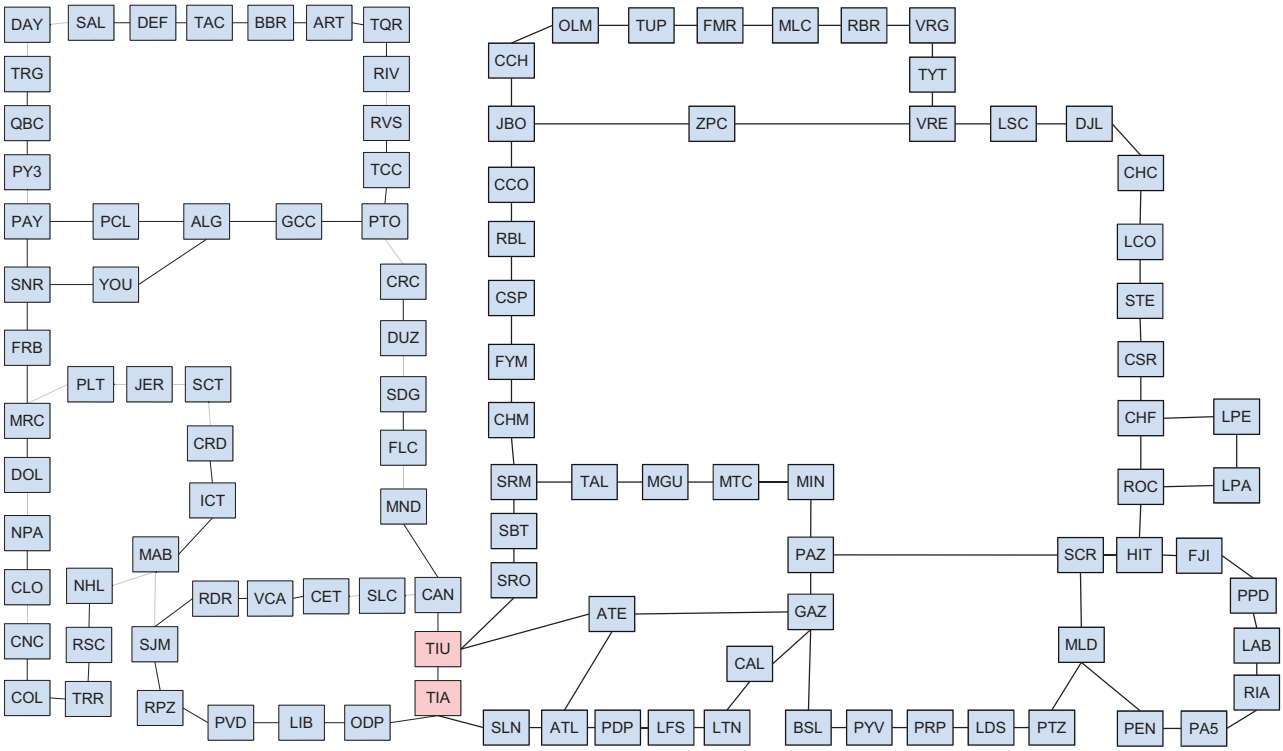

Figure 5.1 – ANTEL's Transport Network.

As this network is extremely big to make any useful calculation [32] it is divided in two regions; the East Region and the West Region. By dividing the network in this way we still maintain the ring topology while generating two smaller networks. Both networks can be observed in Figure 5.2 and 5.3 respectively. It can be seen how nodes TIU and TIA appear in both networks. The data nodes corresponding to each region were preserved along with the induced data links.

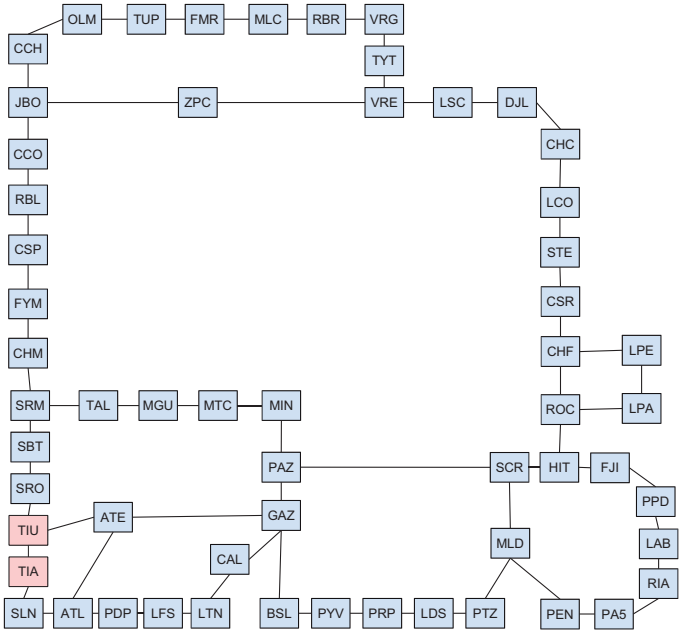

Figure 5.2 – East Region of ANTEL's Transport Network.

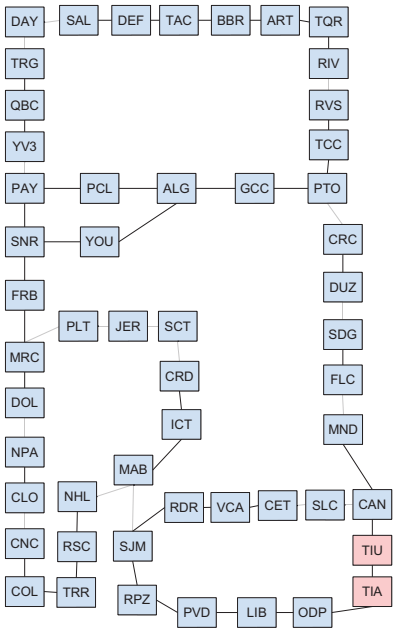

Figure 5.3 – West Region of ANTEL's Transport Network.

These regions are named `east_copy` and `west_copy` respectively.

We only consider traffic demands between nodes of each region.

Among the different test cases we modified some variables. The first variable that we modified is the bandwidth. Table 5.1 shows the set of available speeds, $\hat{B}$, and their costs. This information was provided by ANTEL.

$\hat{\mathbf{B}}$	**Speed** (Mbps)	**VCAT/DWDM** Mapping	**Useful Capacity** (Mbps)	**E1 Equivalent** (2 Mbps) (Number of E1)	**Cost** (US$/km)
b_0	0	-	0	0	0.0
b_1	10	5 VC12	9.5	5	3.46
b_2	20	10 VC12	19	10	6.92
b_3	40	20 VC12	38	20	13.84
b_4	50	1 VC3	42	21	14.54
b_5	100	2 VC3	84	42	29.10
b_6	140	1 VC4	132	63	43.60
b_7	280	2 VC4	264	128	88.60
b_8	10,000	1 λ	10,000	5,263	104.00

Table 5.1 – Transport Network capacities and costs.

When we run the algorithm different bandwidths were created. They span from 1000 Mbps to 10000 Mbps with a step of 1000 Mbps. The costs were assigned proportionally to the bandwidth. This is shown in Table 5.2.

$\hat{\mathbf{B}}$	**Speed** (Mbps)	**Cost** (US$/km)
b_0	0	0
b_1	1000	10
b_2	2000	20
b_3	3000	30
b_4	4000	40
b_5	5000	50
b_6	6000	60
b_7	7000	70
b_8	8000	80
b_9	9000	90
b_{10}	10000	100

Table 5.2 – Modified capacities and costs.

We introduce this variation because when testing the integer programming model [32] this modification was also done to compare that model with some metaheuristics [17, 18]. Thus, in order to compare the performance of the algorithm with the integer programming model we

must have the same set of test scenarios. The upper capacity, 10000 Mbps, was kept having as reference the capacity and cost information provided by ANTEL. The tests that introduce this change in the set of bandwidths are identified by the word `cap` in their name.

Last but not least, we also modify the traffic demand. For this set of tests the traffic weight of the network was increased. Traffic demands were generated randomly and uniformly distributed from 0 to 30000 Mbps. The number of nodes with traffic demand is half the number of data edges. The set of tests that includes this modification are identified by the word `charge` in their names, giving the notion that in these networks almost all edges will be designed and charged, if possible, close to its maximum capacity.

5.2 Performance Test Cases

In this set of test cases we run the algorithm using the complete network, showed in Figure 5.1. We run these test cases to test the algorithm in a big Transport Network with many data nodes exchanging data with each other while testing its potential to find new transport links that can transport the data flows more efficiently reducing the transport routing costs.

In order to let the algorithm evaluate the installation of new transport links we set the distance of these new links as the one of a straight line that connects its endpoints. As we count on the geographical information for each transport node this can be done easily. This is shown in the following example.

Example: Consider the sub-network of the Transport Network showed in Figure 5.4.

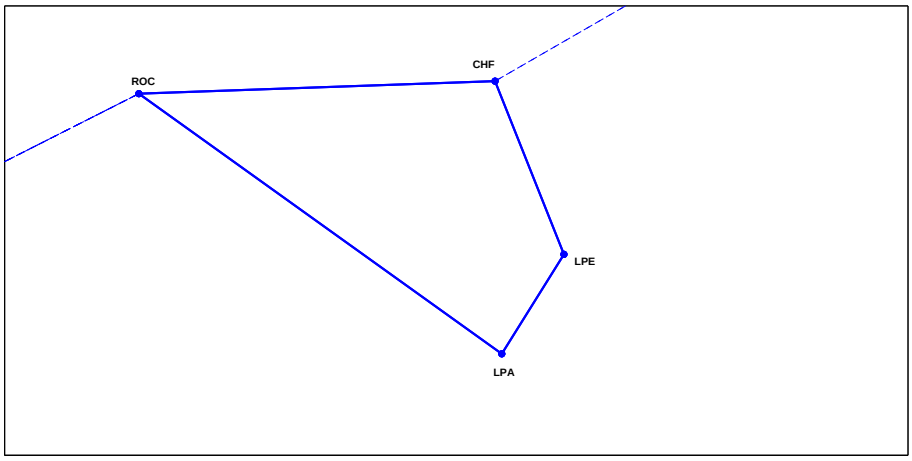

Figure 5.4 – Sub-network of the Transport Network of the example.

Transport links are showed in blue-solid lines. However, some transport links can be installed between these transport nodes. These potential links are shown in Figure 5.5 in red-dotted lines.

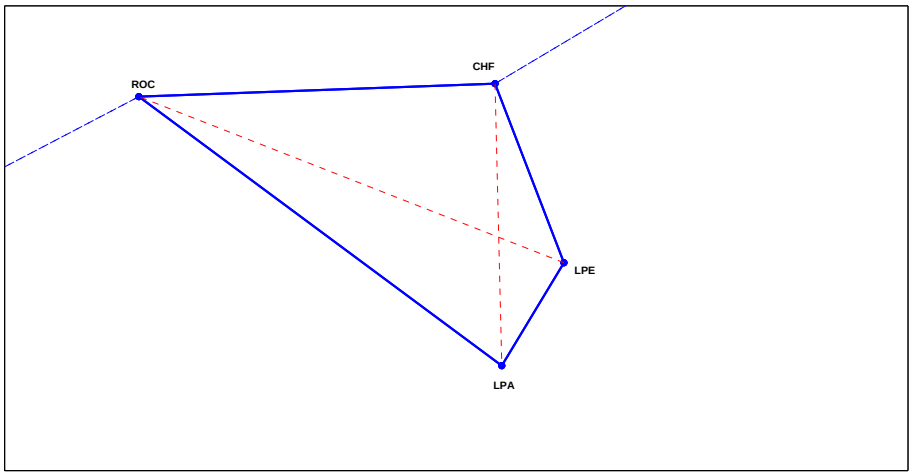

Figure 5.5 – Sub-network of the Transport Network of the example with potential links.

For the sake of simplicity the installation cost will for each potential transport link is set as the same value of the distance between its endpoints. The installation budget is set as a distance value. For example, an installation budget of $I = 1000$ means that the algorithm can install additional transport links if and only if the length of the sum of all installed links is below $1000\,km$.

In this set of test cases we have all variables fixed but the installation budget. Thus, we will run some examples varying that parameter and study the performance of the algorithm.

Chapter 6

Results of the Test Cases

In this chapter we explain the results we obtained when we run the test cases.

The results from the developed algorithm, the integer programming model and metaheuristics were obtained after running the test cases in two servers from the Computer Science Institute of the University of the Republic whose main features are the following:

- Intel Core i7-975, 16GiB (3.33GHz, 8MiB Cache, 6.40GT/s).
- Intel Core 2 Quad Q9550, 4GiB (2.83GHz, 12MiB Cache, 1333MHz FSB).

Now we show and analyze the results obtained for both set of test cases.

6.1 Results of the First Set of Test Cases

Table 6.1 shows most relevant data for the test cases.

Test Case	Number of Data Links	Number of Data Nodes	Number of Traffic Demands	Number of Capacities
`east_copy`	15	18	11	2
`east_copy_cap`	15	18	11	10
`east_copy_charge_cap`	15	18	7	10
`west_copy`	47	18	26	2
`west_copy_cap`	47	18	26	10
`west_copy_charge_cap`	47	18	26	10

Table 6.1 – Relevant data of test cases.

Now we analyze the outputs for the test scenario `east_copy` as a demonstration of how the algorithm proceeds to find the solution.

Figure 6.1 shows the plot of the Transport Network.

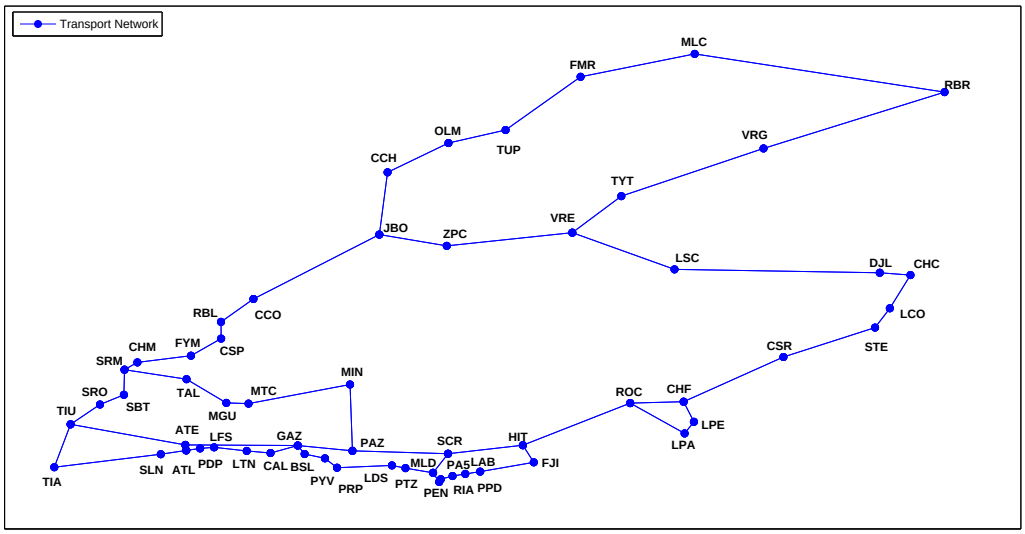

Figure 6.1 – Transport Network for the test case `east_copy`.

Figure 6.2 shows the nominal routing scenario after mapping the data flows into the Transport Network.

These routes transports the data flows in the Transport Network when all the links are active. Table 6.2 summarizes that information.

Route Number	First Endpoint	Second Endpoint
1	GAZ	ATL
2	ATL	TIU
3	MLD	GAZ
4	MIN	PAZ
5	SRM	MIN
6	MLC	TYT
7	$PA5$	MLD
8	PAZ	SCR
9	ROC	SCR
10	ROC	TYT
11	SRM	TIU

Table 6.2 – Routes in the Transport Network in the nominal routing scenario.

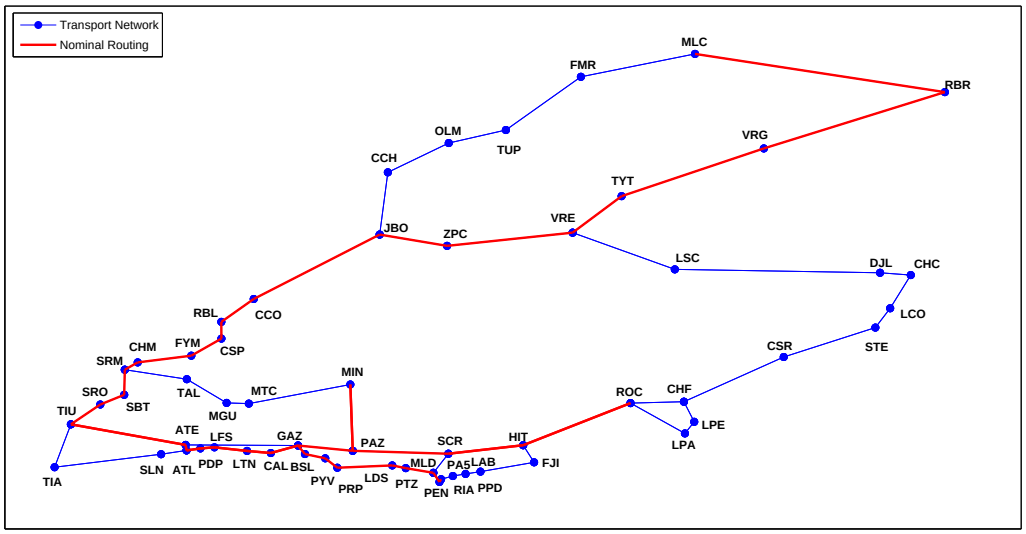

Figure 6.2 – Nominal routing for the test case `east_copy`.

As the routes are connected to each other, the algorithm only finds one connected subgraph. The algorithm identifies the following grade-1 nodes:

- MLC
- MIN
- ROC
- PEN

After the grade-1 nodes are identified, the algorithm finds the 2-connected network. In order to achieve that, it connects each grade-1 node to the transport node in the connected subgraph whose distance is the shortest (without considering the one that is already connected to it). Figure 6.3 shows the 2-connected network identified by the algorithm.

Once the 2-connected network is identified the algorithm calculates the routing for each single-link failure scenario, the Data Network routing and calculates the routing cost.

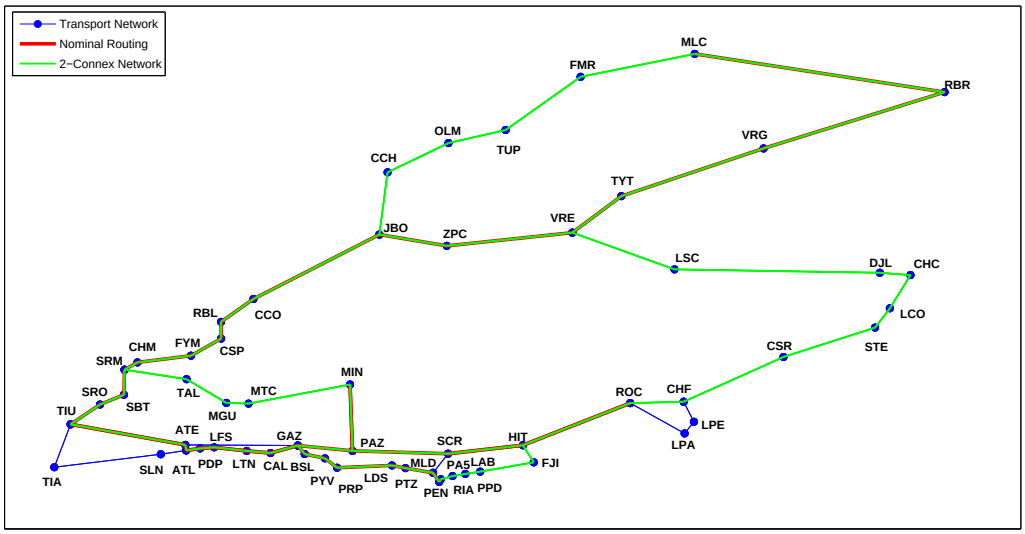

Figure 6.3 – 2-connected network identified by the algorithm for the test case `east_copy`.

Table 6.3 presents the results of the metaheuristics and integer programming model. The column VNS corresponds to the best cost obtained by solving the test case using VNS (Variable Neighborhood Search) [17], and the column T_{VNS} represents the elapsed time. In the same way, column TS corresponds to the best cost obtained applying Tabu Search [18], and the column T_{TS} represents the elapsed time. Similarly, column IP corresponds to the cost obtained using the integer programming model, while T_{IP} represents the elapsed time. This information was obtained from [32].

Test Case	**VNS**	**T_{VNS} (s)**	**TS**	**T_{TS} (s)**	**IP**	**T_{IP} (s)**
`east_copy`	112206	5	112207	90	89175	1.3
`east_copy_cap`	107891	7	107891	77	15114	28
`east_copy_charge_cap`	111247	240	95185	140	28702	0.6
`west_copy`	977649	45	977650	78	605957	70471
`west_copy_cap`	816818	33	9400048	112	188020	1699
`west_copy_charge_cap`	774090	34	940048	40	277387	16528

Table 6.3 – Results of the metaheuristics and the integer programming model.

Table 6.4 shows the results of the algorithm using the same set of test cases. It also shows the results of Table 6.3.

Test Case	MOBCRN		VNS		TS		IP	
	Cost	**T** (s)	Cost	**T** (s)	Cost	**T** (s)	Cost	**T** (s)
east_copy	93692	2.5	112206	5	112207	90	89175	1.3
east_copy_cap	15880	4.8	107891	7	107891	77	15114	28
east_copy_charge_cap	29193	2.3	111247	240	95185	140	28702	0.6
west_copy	730328	10.7	977649	45	977650	78	605957	70471
west_copy_cap	292310	9.4	816818	33	9400048	112	188020	1699
west_copy_charge_cap	364328	9.7	774090	34	9400048	40	277387	16528

Table 6.4 – Results of the algorithm versus the metaheuristics and the integer programming model.

The results from Table 6.4 shows that the routing cost obtained when using the algorithm are slightly higher than the ones obtained when using the integer programming model. However they are much lower than the metaheuristics.

The brightest side of the MOBCRN is regarding time performance. Four out of six tests showed a better performance than the fastest way of solving the problem, including these in which the IP cost is much than the MOBCRN (`west_copy` test cases).

We can conclude that despite the IP algorithm finds better solutions for large networks, the elaped time in which the MOBCRN finds the solutions is extremely lower than the IP. For example, in `west_copy` test case, the time it tooked for the IP to find solutions were around **19 hours and 35 minutes**, while the MOBCRN found the solution in just **10.7 seconds**.

Table 6.5 summarizes and quantifies these observations.

Test Case	Cost improvement over VNS	Cost improvement over TS	IP cost improvement over MOBCRN
east_copy	16.5 %	16.5 %	5.1 %
east_copy_cap	85.3 %	85.3 %	5.1 %
east_copy_charge_cap	73.8 %	69.3 %	1,7 %
west_copy	25.3 %	25.3 %	20.5 %
west_copy_cap	64.2 %	96.9 %	55.5 %
west_copy_charge_cap	52.9 %	52.9 %	31.3 %

Table 6.5 – Comparison between the cost of the MOBCRN and the VNS, TS and IP respectively.

6.2 Results of the Second Set of Test Cases

As we explained in section 5.2 in this set of test cases we fix all the parameters but the installation budget, I. As we set I as the total length of transport links that can be installed, we study the distribution of its lengths to determine the different values of I.

Figures 6.4 and 6.5 shows the distribution and cumulative distribution of the lengths of the potential transport links.

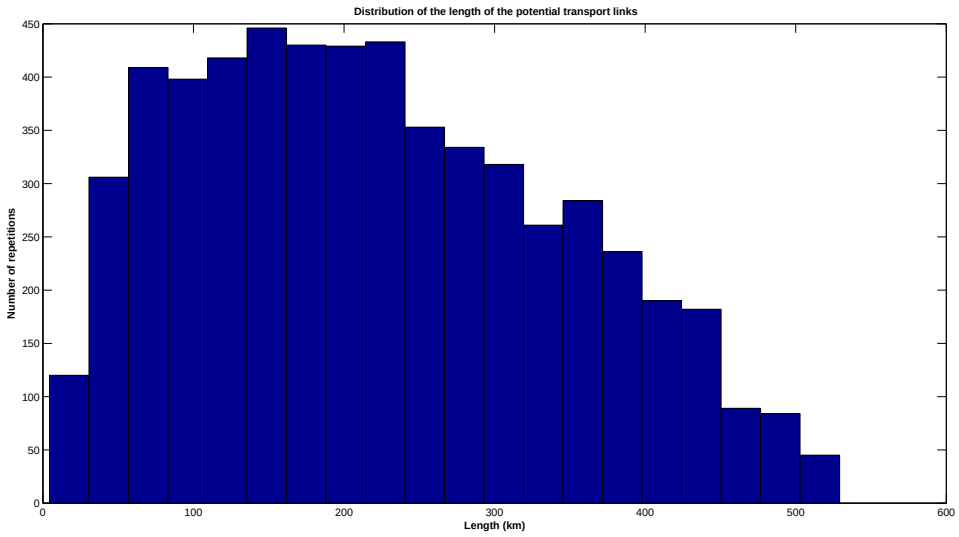

Figure 6.4 – Distribution of the lengths of the potential transport links.

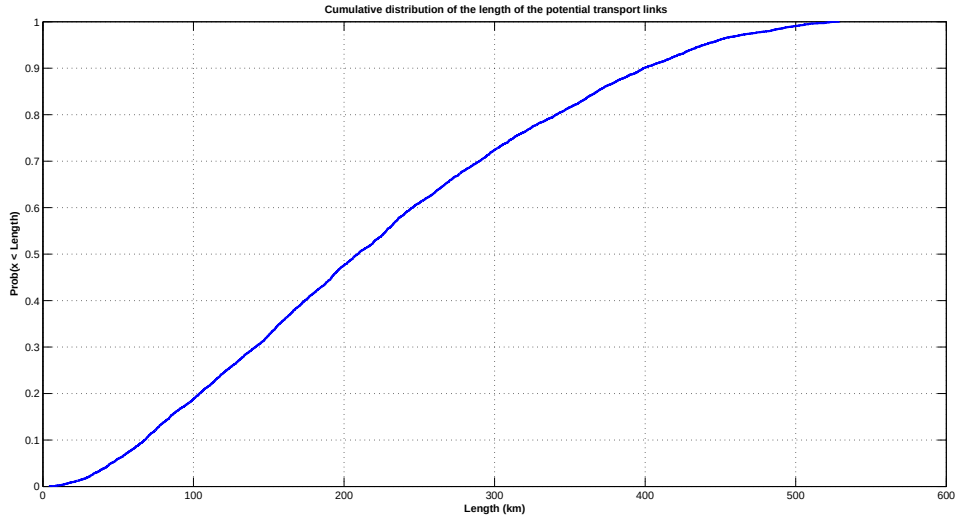

Figure 6.5 – Cumulative distribution of the lengths of the potential transport links.

With that information we construct a plot in which we can see how many transport links can be installed according to I. This can be observed in Figure 6.6.

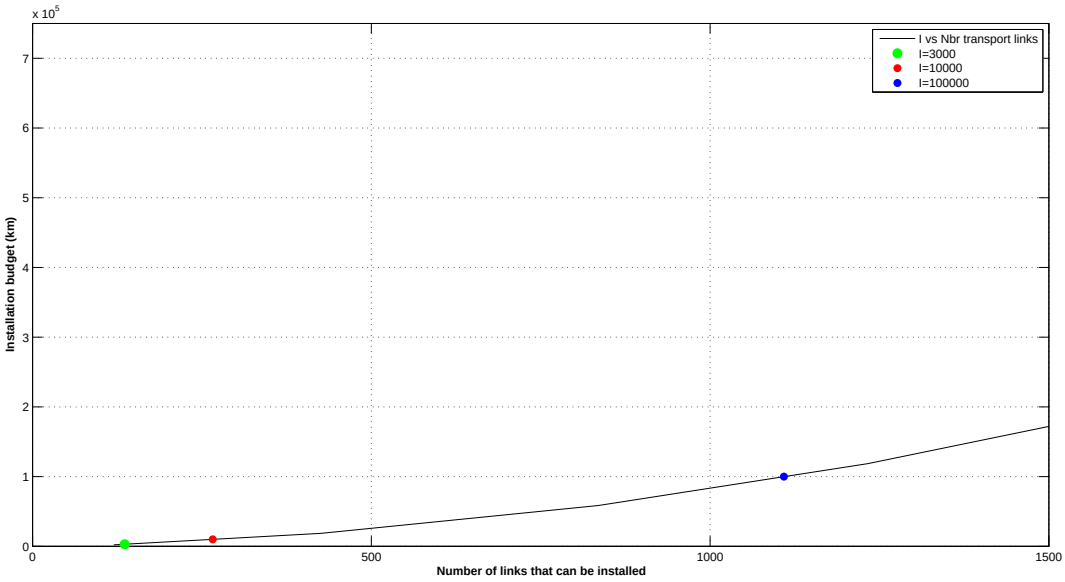

Figure 6.6 – Number of transport links that can be installed versus I.

Based on Figure 6.6 we define three different numbers for the installation budget. These numbers are listed in table 6.6 and are showed in Figure 6.6 as coloured dots.

	Length (km)	**Number of potential links that can be installed**
I_1	3000	136
I_2	10000	266
I_3	100000	1109

Table 6.6 – Number of links that can be installed versus I.

In addition to these three values of I we add the case in which $I = 0$, this is when there is not any possibility to install new transport links.

Table 6.7 shows test cases we defined according to each I value.

Test Case Number	Data Links	Data Nodes	Transport Links	Transport Nodes	Demands	Number of Capacities	I (km)
1	55	34	122	109	36	2	0
2	55	34	122	109	36	2	3000
3	55	34	122	109	36	2	10000
4	55	34	122	109	36	2	100000

Table 6.7 – Relevant data of the test cases.

6.2.1 Test case number one

Figure 6.7 shows the Transport Network used to run the algorithm. This is the complete version of ANTEL's Transport Network.

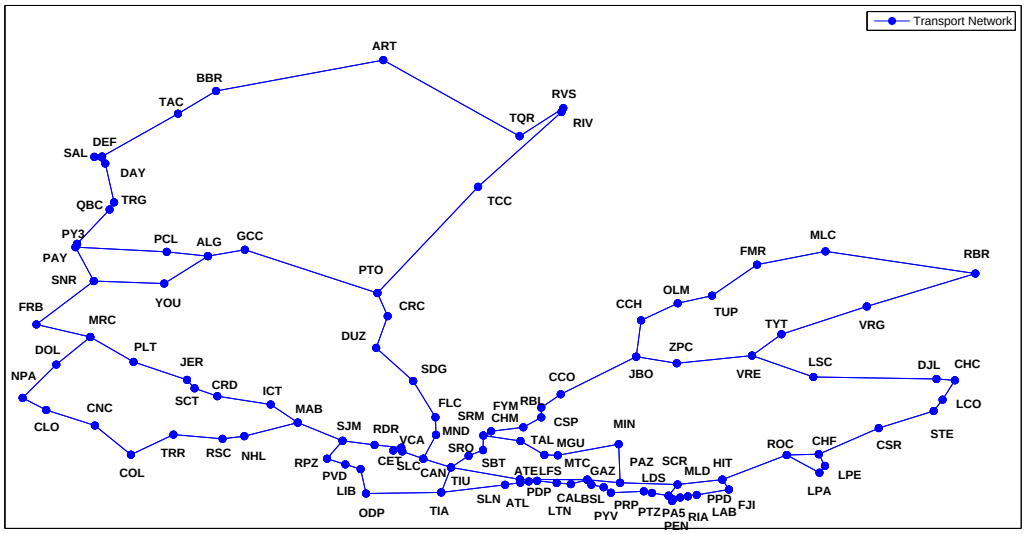

Figure 6.7 – Transport Network used for the second set of test cases.

Figures 6.8 and 6.9 shows the nominal routing and the 2-connected network for the test case number one after running the algorithm.

In this test case the routing cost is 175010.

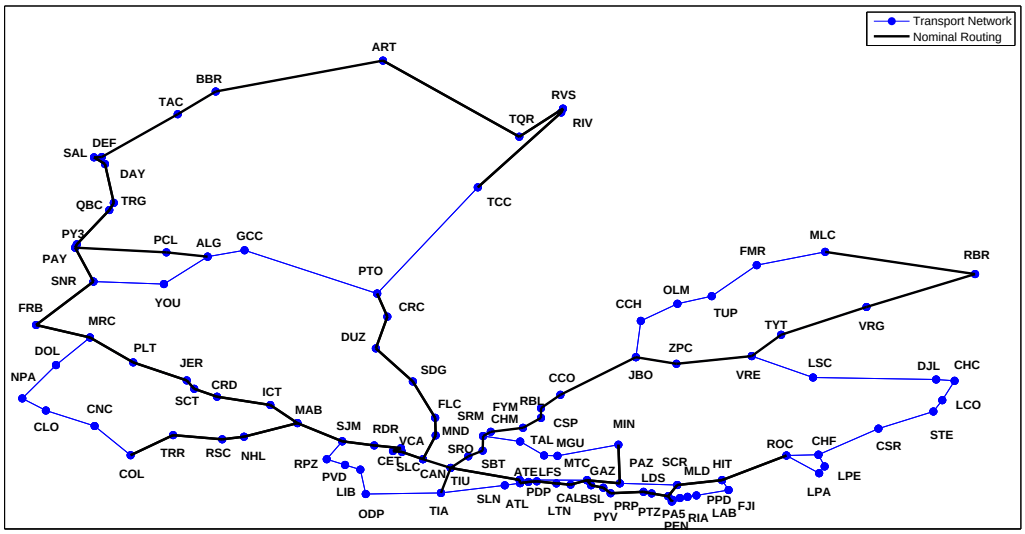

Figure 6.8 – Nominal routing on top of the Transport Network for test case number one.

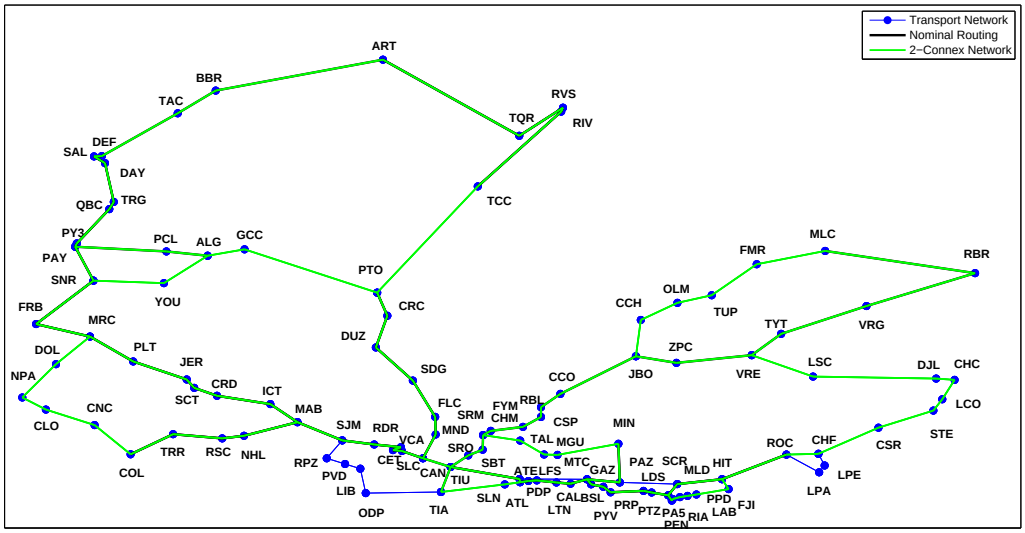

Figure 6.9 – 2-connected network calculated by the algorithm for test case number one.

6.2.2 Test case number two

Figure 6.10 shows the Transport Network for the second test case with the installed transport links. In this case 12 new transport links were installed after running the algorithm.

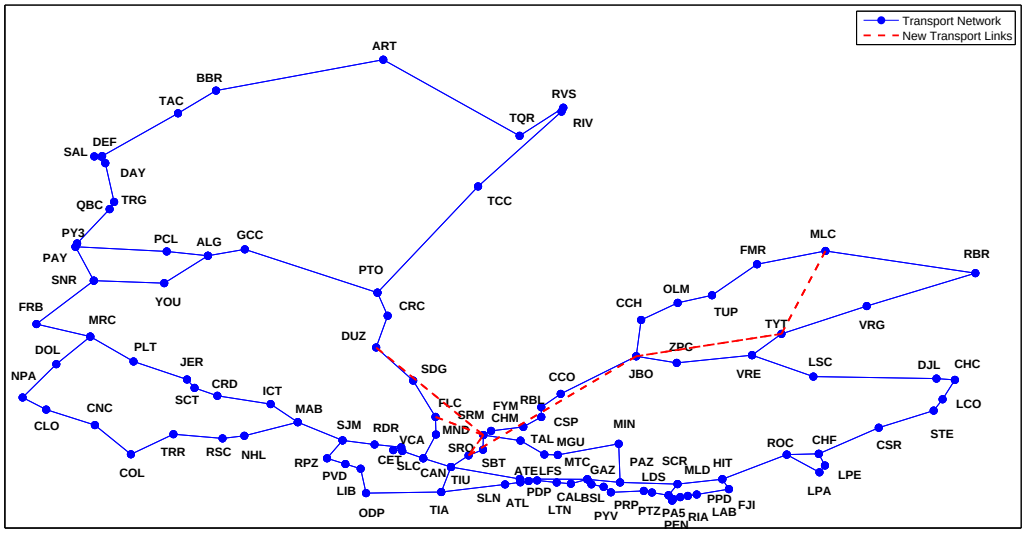

Figure 6.10 – Transport Network used for the second set of test cases.

Figure 6.11 shows the nominal routing for this test case. It can be seen how the nominal routing is performed on top of the installed transport links.

Finally, Figure 6.12 shows the 2-connected network used by the algorithm to find the solution.

In this test case the routing cost is 174146.

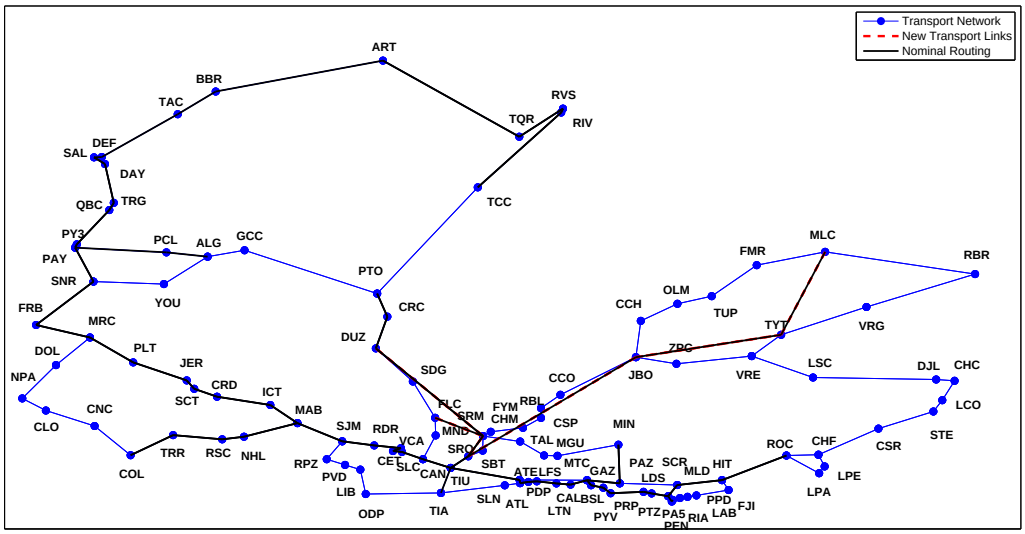

Figure 6.11 – Nominal routing on top of the Transport Network for test case number two.

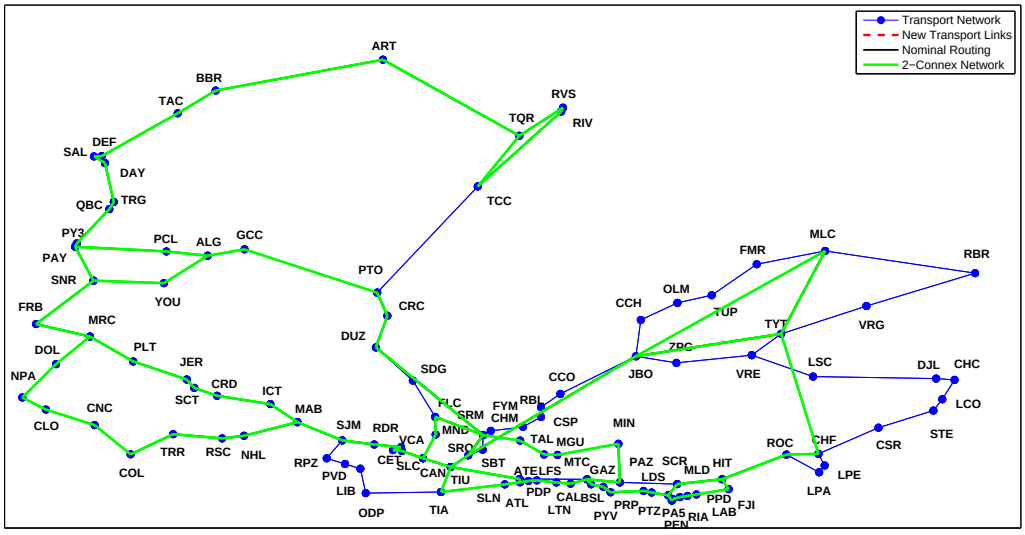

Figure 6.12 – 2-connected network calculated by the algorithm for test case number two.

6.2.3 Test case number three

Figure 6.13 shows the Transport Network for the second test case with the installed transport links. In this case 26 new transport links were installed after running the algorithm.

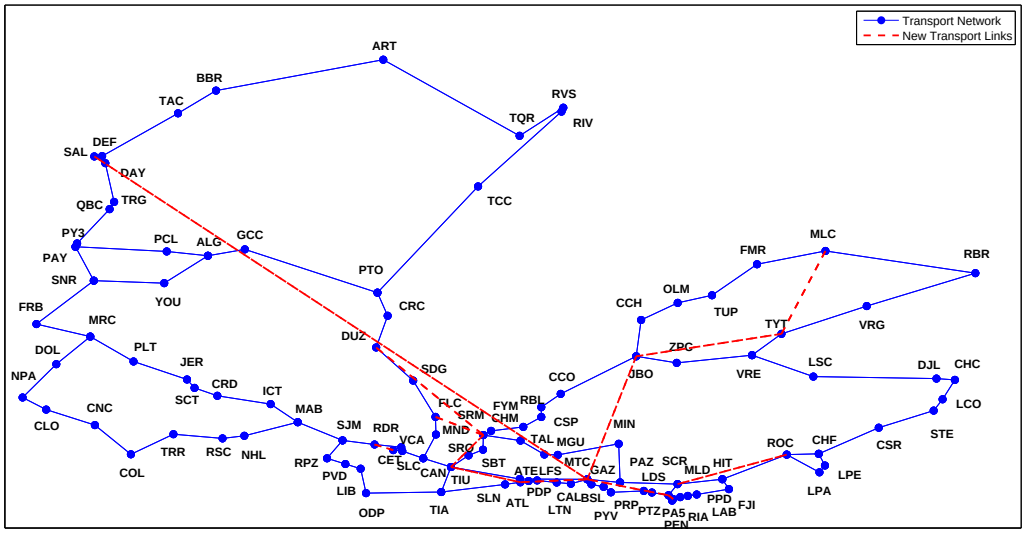

Figure 6.13 – Transport Network used for the third set of test cases.

Figure 6.14 shows the nominal routing for this test case. As in the previous test it can be seen how the nominal routing is performed using the installed transport links.

Finally, Figure 6.15 shows the 2-connected network used by the algorithm to find the solution. It is interesting to observe how the 2-connected network uses the installed transport links discarding many transport links from the original Transport Network.

In this test case the routing cost is 174066.

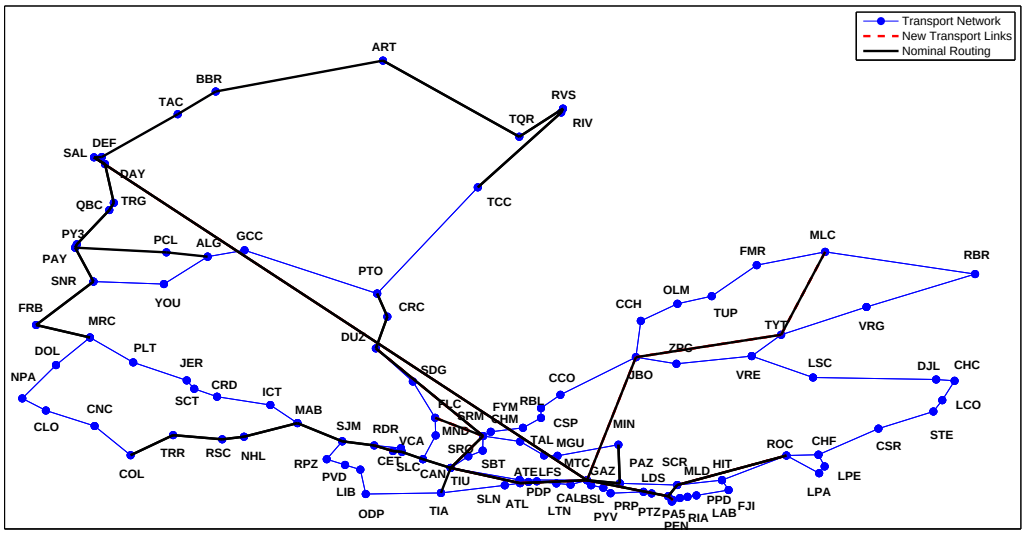

Figure 6.14 – Nominal routing on top of the Transport Network for test case number three.

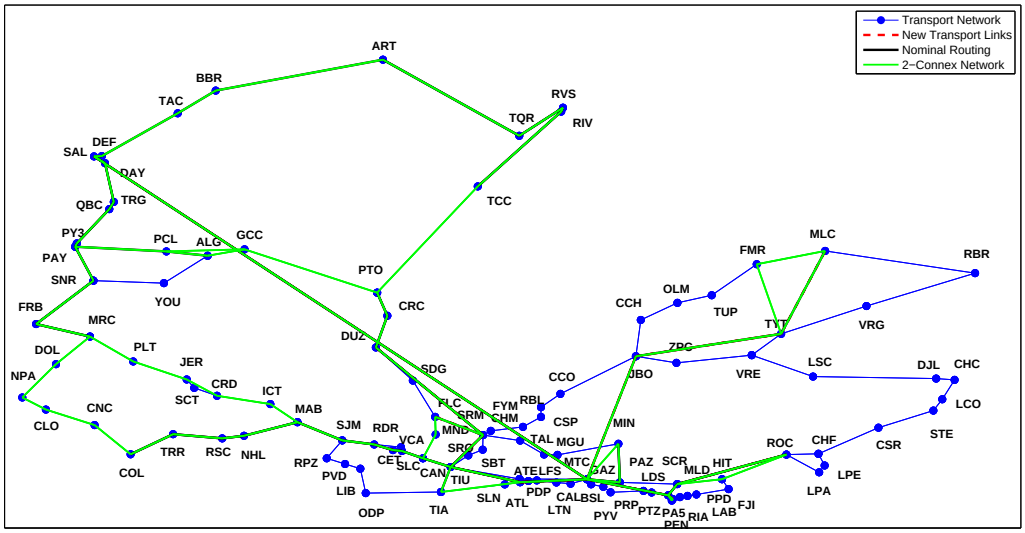

Figure 6.15 – 2-connected network calculated by the algorithm for test case number three.

6.2.4 Test case number four

Figure 6.16 shows the Transport Network for the second test case with the installed transport links. In this case 42 new transport links were installed after running the algorithm.

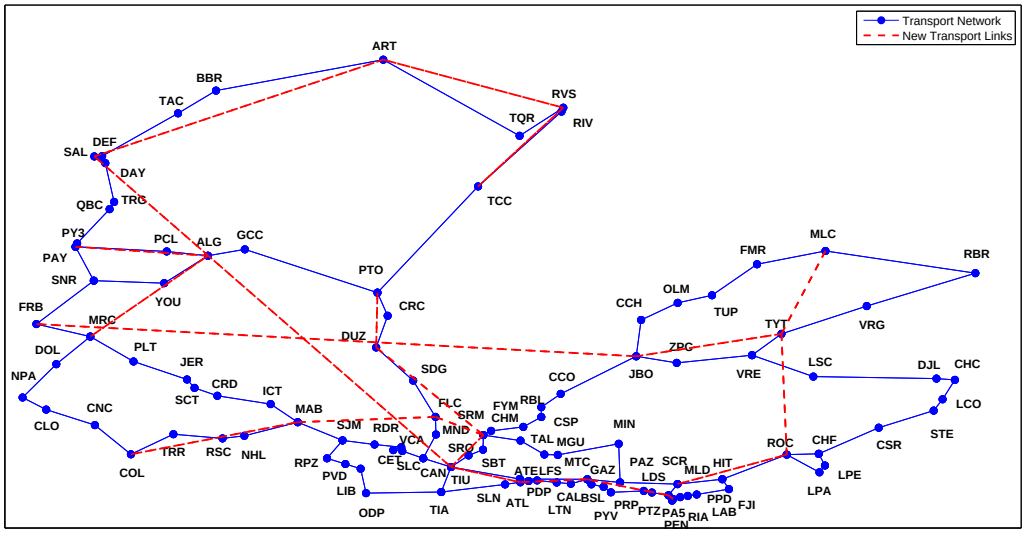

Figure 6.16 – Transport Network used for the fourth set of test cases.

Figure 6.17 shows the nominal routing for this test case. It is interesting to observe that the nominal routing is performed using only installed links. This means that I was big enough to install direct links from and to all traffic endpoints. Thus, the routing cost find in this scenario is the minimum possible.

Finally, Figure 6.18 shows the 2-connected network used by the algorithm to find the solution. This is the 2-connected network of minimum cost.

In this test case the routing cost is 159244.

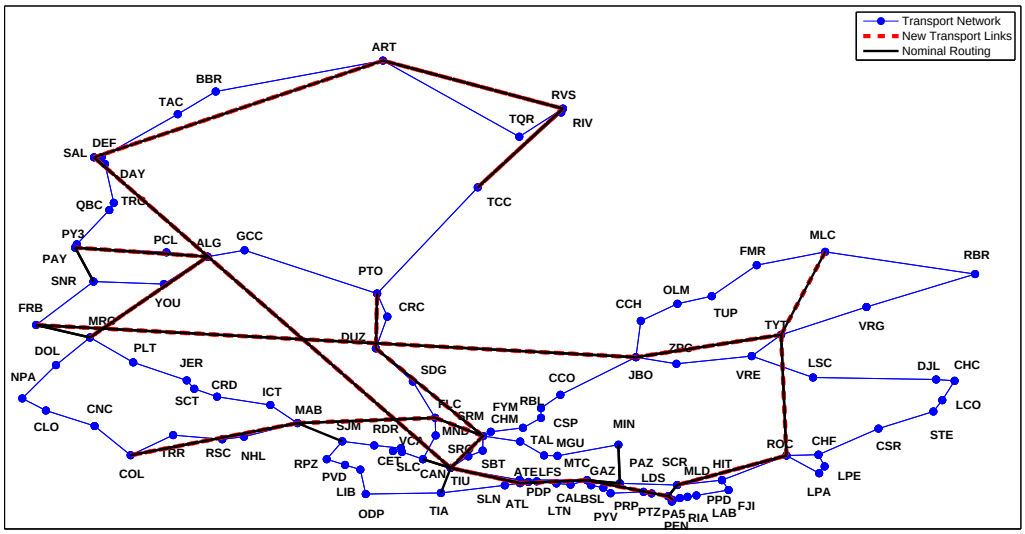

Figure 6.17 – Nominal routing on top of the Transport Network for test case number four.

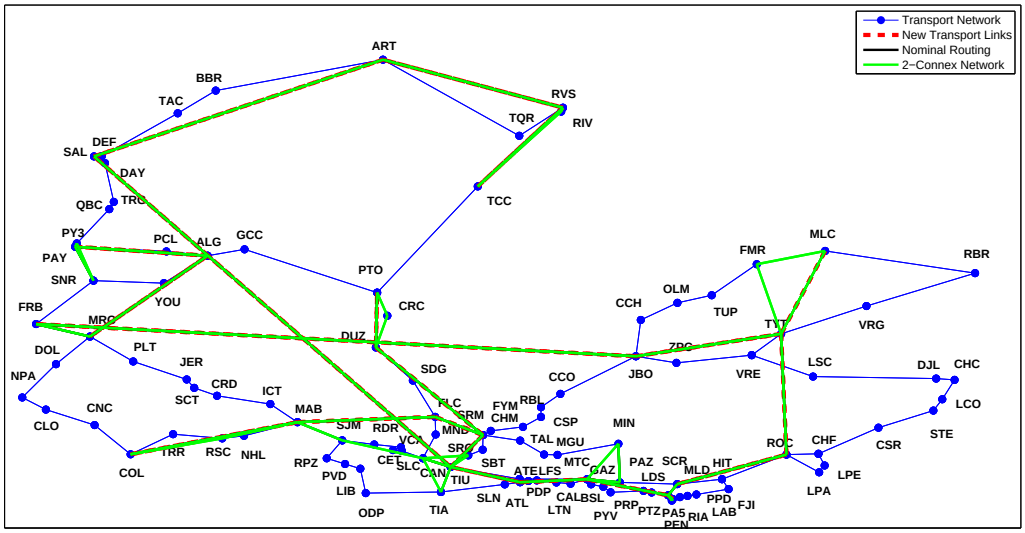

Figure 6.18 – 2-connected network calculated by the algorithm for test case number four.

6.2.5 Conclusions

Table 6.8 summarizes the results obtained when running the test cases.

Test Case Number	**Routing Cost**	**Time (s)**	**Installation** (budget)	**Installed Transport Links**
1	175010	104	0	0
2	174146	127	3000	12
3	174066	134	10000	26
4	159244	139	100000	42

Table 6.8 – Results obtained after running the algorithm for the test cases.

We observe that when higher is I more links are installed, which makes the routing cost being lower. When I increase the nominal routing scenario tends to be more likely to the installed transport links. In the case of test number four both are the same, concluding that the algorithm had enough installation budget to install new links for each pair of traffic endpoints.

Chapter 7

Conclusions

The present project addressed a complex and actual problem for many telecom operators that is how to design and grow the capacity of their Transport Network in order to minimize the routing cost of the data traffic generated in the Data Network.

First, we introduced the MOBCRN problem and presented the state of the art. After that, we showed that the MOBCRN problem is **NP-Hard** and designed a polynomial algorithm that finds solutions for the MOBCRN.

Following that point, we tested the algorithm against some other ways of solving similar problems using the information from a real telecom operator, ANTEL. At this point we proved that the algorithm performs as expected, giving similar values as the best ways of solving the problem.

Finally, we tested the algorithm with the complete ANTEL's network in a very complex scenario for different levels of installation budget, explaining the different outputs of the algorithm for each case.

Because of these points, we can conclude that the presented algorithm performes within the range of previous works, adding the possibility to evaluate the installation of new transport links. This not only allows a telecom operator to design its multi-overlay network, but also plan a growth of its existing network. This was proved using real information showing good results, which is a clear example of the potential of this kind of tools.

Chapter 8

Open Problems and Future Work

Despite the algorithm performs in the range of previous works in the same area, there are many improvements that can be done in order to have faster and better solutions for the problem.

On the one hand, the way the algorithm installs new links is far from being optimal. The present algorithm installs links starting from the first traffic demand to be analyzed. This can frequently produce situations in which the optimal set of installed links are not chosen. A possible solution for this problem is to consider all possible combinations of installed links according to the installation budget and keep the one that minimizes the routing cost. This was discarded beacause that approach consumes a lot of computational resources. A more clever way to solve it could be to formulate an integer programming model to assign the optimal set of installed links.

On the other hand, a similar situation is faced when we find the shortest paths for the nominal routing. As the algorithm runs a shortest-path algorithm for each flow separately, the optimal solutions is not likely to be found.

As this algorithm only tries to minimize the routing cost for a given traffic scenario, it can be intereseting to formulate a problem to study which is the combination of installed links that produces the minimun routing cost for a set of traffic scenarios that a telecom operator is likely to experience during its day-by-day operation.

Moreover, it would also be intereseting to study wich is the set of links that minimizes the product between the installation budget and routing cost. This will help the operator to know not only which transport links must be installed to optimize its network, but also how much budget they must consider.

A reformulation of the problem can also be performed to find solutions under unfeasible scenarios as stated in Section 4.2.3. Covering these scenarios would help to find solutions to a wider range of network configuations.

Bibliography

[1] 802.3 Ethernet Working Group. Technical report, Institute of Electrical and Electronics Engineers.

[2] EIBONE, Efficient Integrated Backbone project website.

[3] ITU-T Recommendation G.803: Architecture of transport networks based on the synchronous digital hierarchy (SDH). Technical report, International Telecommunication Union, 2004.

[4] ITU-T Recommendation X.200: Basic Reference Model: The basic model. Technical report, International Telecommunication Union, 2004.

[5] Fundamentals of DWDM Technology. Technical report, Cisco Networks, 2006.

[6] ITU-T Recommendation G.784: Management aspects of synchronous digital hierarchy (SDH) transport network elements. Technical report, International Telecommunication Union, 2006.

[7] ITU-T Recommendation G.783: Characteristics of synchronous digital hierarchy (SDH) equipment functional blocks. Technical report, International Telecommunication Union, 2008.

[8] ITU-T Recommendation G.707: Network node interface for the synchronous digital hierarchy (SDH). Technical report, International Telecommunication Union, 2009.

[9] Dimitris Alevras, Martin Grötschel, and Roland Wessäly. A network dimensioning tool. In *Preprint SC 96-49, Konrad-Zuse-Zentrum fur Informationstechnik*, pages 49–96. Konrad-Zuse-Zentrum fur Informationstechnik, 1996.

[10] Yossi Azar and Oded Regev. Combinatorial algorithms for the unsplittable flow problem. *Algorithmica*, 2006.

[11] A. Balakrishnan, T. L. Magnanti, and P. Mirchandani. A dual-based algorithm for multi-level network design. *Management Science*, 40(5):567–81, 1994.

[12] Anantaram Balakrishnan, Thomas L. Magnanti, and Prakash Mirchandani. Modeling and heuristic worst-case performance analysis of the two-level network design problem. *Management Science*, 40(7):846–867, 1994.

[13] Anantaram Balakrishnan, Thomas L. Magnanti, and Prakash Mirchandani. Designing hierarchical survivable networks. *Operations Research*, 46(1):116–136, January 1998.

[14] Anantaram Balakrishnan, Thomas L. Magnanti, and Prakash Mirchandani. Connectivity-splitting models for survivable network design. *Networks*, 43(1):10–27, January 2004.

[15] R. Bhandari. Optimal physical diversity algorithms and survivable networks. *Proc. 2nd IEEE Symp. Computers and Communications (ISCC)*, pages 433–441, 1997.

[16] Daniel Bienstock, Sunil Chopra, Oktay Günlük, and Chih-Yang Tsai. Minimum cost capacity installation for multicommodity network flows. *Mathematical Programming*, 81:177–199, 1998.

[17] Andrés Corez. Multi-overlay network planning by applying a variable neighbourhood search approach. Master's thesis, Universidad de la República Oriental del Uruguay, Montevideo, Uruguay, 2010.

[18] Francois Despaux. Optimización de una red de datos IP/MPLS sobre SDM/DWDM usando tabú search. Caso de estudio: Red de datos de un operador de telefonía nacional. Master's thesis, Universidad de la República Oriental del Uruguay, Montevideo, Uruguay, 2011.

[19] E. W. Dijkstra. In *Numerische Mathematik*, pages 269–271.

[20] Thomas A. Feo and Mauricio G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[21] Bernard Fortz and Michael Poss. An improved benders decomposition applied to a multi-layer network design problem. *Oper. Res. Lett.*, 37(5):359–364, 2009.

[22] Alexander Gersht and Alexander Shulman. A new algorithm for the solution of the minimum cost multicommodity flow problem. *26th IEEE Conference In Decision and Control*, 26:748–758, 1987.

[23] M. Grötschel, C.L. Monma, and M. Stoer. Design of survivable networks, 1993.

[24] D. Katz, Kompella K., and D. Yeung. RFC 3630: Traffic Engineering (TE) Extensions to OSPF Version 2. Technical report, IETF, 2003.

[25] Hervé Kerivin and A. Ridha Mahjoub. Design of survivable networks: A survey. In *In Networks*, pages 1–21, 2005.

[26] Hervé Kerivin, Dritan Nace, and Thi-Tuyet-Loan Pham. Design of capacitated survivable networks with a single facility. *IEEE/ACM Trans. Netw.*, 13(2):248–261, 2005.

[27] A.M.C.A. Koster, S. Orlowski, C. Raack, G. Baier, and T. Engel. Single-layer cuts for multi-layer network design problems. *Telecommunications Modeling, Policy, and Technology*, pages 1–23, 2008.

[28] T. L. Magnanti, P. Mirchandani, and R. Vachani. Modeling and Solving the Two-Facility Capacitated Network Loading Problem. *Operations Research*, 43:142–157, 1995.

[29] T. Magnanti, P. Mirchandani, and R. Vachani. Modeling and solving the capacitated network loading problem. *Operations Research Center Working Paper*, 01:239–291, 1991.

[30] Haruko Okamura and P.D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75 – 81, 1981.

[31] P. Pan, Swallow G., and A. Atlas. RFC 4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels. Technical report, IETF, 2005.

[32] Cecilia Parodi. Integer Optimization Applied to the Design of Robust Minimum Cost Multi-Layer Networks. Master's thesis, Universidad de la República Oriental del Uruguay, Montevideo, Uruguay, 2011.

[33] A. Bley, U. Menne, R. Klähne, C. Raack, and R. Wessäly. Multi-layer network design – A model-based optimization approach. In *Proceedings of the 5th Polish-German Teletraffic Symposium 2008*, pages 107–116, Berlin, Germany, 2008.

[34] Christian Raack, Arie M.C.A. Koster, Sebastian Orlowski, and Roland Wessäly. On cut-based inequalities for capacitated network design polyhedra. *Networks*, 57(2):141–156, 2011.

[35] S. Orlowski, A. Koster, C. Raack, and R. Wessäly. Two-layer network design by branch-and-cut featuring mip-based heuristics. In *In Proceedings of the Third International Network Optimization Conference*. INOC, 2007.

[36] S. Ramamurthy and Biswanath Mukherjee. Survivable wdm mesh networks, part 1 - protection. In *INFOCOM*, pages 744–751, 1999.

[37] S. Ramamurthy and Biswanath Mukherjee. Survivable wdm mesh networks, part 2 - restoration. In *INFOCOM*, pages 2023–2030, 1999.

[38] Claudio Risso. Optimización de Costos en Redes Multicapa Robustas. Master's thesis, Universidad de la República Oriental del Uruguay, Montevideo, Uruguay, 2010.

[39] E. Rosen, A. Viswanathan, and R. Callon. RFC 3031: Multiprotocol Label Switching Architecture. Technical report, IETF, 2001.

[40] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee. Fault management in ip-over-wdm networks: Wdm protection versus ip restoration. *IEEE J.Sel. A. Commun.*, 20(1):21–33, September 2006.

[41] K. Sunggy, G. Sahin, and S. Subramaniam. Cost efficient lsp protection in ip/mpls over wdm overlay networks. In *In Communications, ICC 03. IEEE International Conference on*, pages 1278–1282 vol.2, 2003.

[42] M. Skutella. Approximating the single source unsplittable min-cost flow problem. In *In Foundations of Computer Science. Proceedings. 41st Annual Symposium on*, pages 136–145, 2000.

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

[44] Robert Endre Tarjan. Deep-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160.

[45] Pushkar Tripathi. A deterministic algorithm for the vertex connectivity survivable network design problem. *Comput. Res. Repos. 2010, Article No. 1004.1208 (2010). arXiv:1004.1208*.

List of Figures