TESIS

a ser presentada el día 14 de Octubre de 2010 en la

Universidad de la República, UdelaR

para obtener el título de

MAGISTER EN INGENIERÍA MATEMÁTICA

para

Andrés COREZ

Instituto de Investigación : LPE - IMERL
Componentes universitarios :
UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

Título de la tesis :

Multi-Overlay Network Planning
by applying a Variable Neighbourhood Search Approach

a realizarse el 14 de Octubre de 2010 por el comité de examinadores

Dr. Franco	ROBLEDO	Director de Tesis
Msc. Ing. Graciela	FERREIRA	
Msc. Ing. Antonio	MAUTTONE	
Dr. Aldo	PORTELA	
Dr. Pablo	RODRIGUEZ-BOCCA	

Acknowledgements

I would like to acknowledge every people that in one way or another has had an influence in the accomplishment of this work. From those that have brought about ideas, suggestions, experiences or knowledge; those who have shown methods, techniques or work procedures; or those who have informed or collaborated in the resolution of the problem. To those who have counseled, animated or given me strength to carry out this enterprise.

Firstly, I am endlessly grateful to my Academic Director Dr. Franco Robledo for his permanent motivation and valuable advice, supporting my work incessantly and in every possible way. He has trusted me and guided me to the completion of this labour.

I am much obliged with the telecommunication enterprise of the Oriental Republic of Uruguay, ANTEL, and the Statistics and Probability Laboratory, LPE, in sight that this work has been undertaken as part of the specific activity known as "Optimal Design of Robust Multi-overlay Networks" *in the framework of the agreement between ANTEL and the Engineering University of the Republic University,* UdelaR.

I must mention and thank the Engineering Mathematics SCAPA, besides from its supervision and direction of its students, for the scholarship I perceived during part of my Master studies. This allowed me to complete several courses and advance in this work in a much more focused and unconstrained manner.

To IMERL and IIE institutes of the Engineering University for favouring as far as possible their teachers to continue their post-graduate studies.

I would also like to thank to the MSc. Claudio Risso for his unique knowledge of the problem and his skilful development of several phases of the project. To the Eng. François Despaux for his constant and pleasant coding help. He was there every time I asked and has been quite a dedicated companion. To Dr. Eduardo Canale for his instruction in networks and algorithms and his contributions in mathematical aspects. To Eng. Cecilia Parodi for her friendship, cooperation and suggestions.

To my girlfriend, family, and friends for encouraging, comforting and holding me all the time, even with a diminished attention paid to the most important people of my life. I am infinitely thankful.

Finally, I would like to thank the MSc. Eng. Graciela Ferreira, MSc. Eng. Antonio Mauttone, Dr. Aldo Portela and Dr. Eng. Pablo Rodriguez Bocca for the honour of assessing this work.

General index

Index **1**

I INTRODUCTION **7**

1. Introduction **9**
1.1. Motivation . . . 9
1.2. Objective . . . 10
1.3. Documentation organization . . . 11

2. Problem description **13**
2.1. Introduction to networks . . . 13
2.1.1. OSI Model . . . 13
2.1.2. Transport Network . . . 13
2.1.3. Data Network . . . 14
2.1.4. MPLS . . . 14
2.2. Overlay Networks . . . 15
2.2.1. Overlay Scheme . . . 15
2.2.2. Examples . . . 15
2.3. The problem . . . 16

II THEORETICAL FRAMEWORK **17**

3. Formalisation **19**
3.1. Introduction . . . 19
3.2. Data Network . . . 19
3.2.1. Data nodes . . . 19
3.2.2. Data links . . . 20
3.3. Transport Network . . . 21
3.3.1. Transport nodes . . . 21
3.3.2. Transport edges . . . 22
3.4. Underlay and Overlay relation . . . 23
3.5. Attributes of the problem . . . 24

3.5.1. Traffic . . . 24
3.5.2. Cost . . . 24
3.5.3. Traffic routing . . . 26
3.5.4. Robustness . . . 27

4. Mathematical programming formalisation **29**
4.1. Complete model . . . 29
4.1.1. Definitions . . . 29
4.1.2. Mathematical model . . . 31
4.2. Solved Access-Edge connection model . . . 34
4.2.1. Reduced-MORN mathematical model . . . 37
4.3. Complexity . . . 37
4.3.1. Introduction . . . 37
4.3.2. **P** and **NP** . . . 38
4.3.3. **NP**-completeness . . . 38
4.4. Algorithm complexity . . . 39
4.4.1. 2-edge-connected topology conditions . . . 39
4.4.2. Complexity-related problems . . . 41
4.4.3. STESNP relation . . . 41
4.4.4. MW2ECSN relation . . . 42
4.4.5. SPG relation . . . 43
4.4.6. ANDP relation . . . 44

5. Non-linear binary integer programming model **47**
5.1. Introduction . . . 47
5.2. Problem variables . . . 47
5.3. Complete binary integer model . . . 48
5.4. Constraints . . . 51

III METAHEURISTICS **55**

6. VNS description **57**
6.1. Introduction . . . 57
6.2. Metaheuristics . . . 57
6.3. VNS generalities . . . 57
6.3.1. Combinatorial optimisation problems . . . 58
6.3.2. Neighbourhoods . . . 59
6.3.3. Local search procedure . . . 59
6.4. Fundamental schemes . . . 61
6.4.1. Descendent VNS . . . 61
6.4.2. Reduced VNS . . . 61
6.4.3. Basic VNS . . . 62
6.4.4. General VNS . . . 63

6.4.5. Scheme selected . . . 63
6.5. Virtues of VNS as metaheuristic . . . 64

7. MORN Heuristic **65**
7.1. GRASP construction . . . 65
7.1.1. Initialisation . . . 66
7.1.2. Access-Edge Assignation . . . 70
7.1.3. $G_E sol$ routing over G_T . . . 70
7.1.4. Is $G_E sol$ feasible? . . . 70
7.1.5. G_E reduction . . . 71

8. VNS customisation of the problem **73**
8.1. Introduction . . . 73
8.2. LS: Edge Elimination . . . 73
8.3. LS: Path Reduction . . . 75
8.4. LS: Path Reallocation . . . 76
8.5. LS: Link Decomposition . . . 77
8.6. LS: Insertion/Elimination . . . 79
8.7. LS: Capacity Reduction . . . 81

IV RESULTS **83**

9. Test cases description **85**
9.1. Scenarios Set 1 . . . 85
9.1.1. Demand . . . 85
9.1.2. Requirements . . . 86
9.1.3. Contents . . . 86
9.1.4. Architecture . . . 86
9.2. Data for the problem . . . 87
9.3. Scenarios Set 2 . . . 89

10. Performance tests and analysis of results **93**
10.1. Parameters of testing . . . 93
10.2. Results Set 1 . . . 94
10.3. Results Set 2 . . . 94

11. Conclusions **97**
11.1. Conclusions of the implemented solution . . . 97
11.2. Extensions and future work . . . 98
11.3. Personal experience . . . 98

A. Test case examples **101**
A.1. East Network . . . 101
A.2. West Network . . . 107

Bibliography **117**

List of Figures **118**

Summary

This thesis presents an approach for the topological design and sizing of an IP/MPLS multi-overlay network with the purpose of minimising the economical resources involved on ANTEL (*Administración Nacional de Telecomunicaciones*) infrastructure. The overlay network is an MPLS Data Network, which physically exchange traffic over an existent Transport infrastructure. The solution to find must be of optimal cost, robust to simple failures in the Transport Network, and deal with differential traffic. The mathematical model for this kind of problems is obtained by using weighted graphs which leads to a combinatorial optimisation formulation. As the problem undertaken is NP-Hard concerning computational complexity, a metaheuristic methodology is employed, which reaches approximate though often optimal solutions in a reasonable time. The metaheuristic selected is VNS (Variable Neihbourhood Search), which has shown positive qualities as simplicity, efficiency and effectiveness among others. Then, the implemented local searches and achieved results are described.

Part I

INTRODUCTION

Chapter 1

Introduction

1.1. Motivation

Multi-overlay Networks are a widespread reality nowadays. Internet over the PSTN, P2P over internet, ATM over SDH, are just a few examples of them. There are several reasons for their existence such as organisational, economical, technological, historical, or regulative. Sometimes a new technology has been developed over an already installed network, generally different technological equipment makes use of the same underlay network, in occasions it is economically convenient to rent underlay services of another enterprise, or a company decides it is preferable to separate the administration of each layer under different divisions.

On the grounds that the market of telecommunications has presently become ever competitive, the planning of a feasible network with an increasing degree of optimality in the solutions obtained is vital. The optimal design of the topologies of the networks would lead to a substantial reduction of the economical resources of an enterprise.

Consequently, the study of Multi-overlay Networks and the creation of mathematical models to deal with them is of growing interest. It is also a difficult task with a large field of application.

This thesis is part of a research project carried out jointly by the telecommunication enterprise of the Oriental Republic of Uruguay, ANTEL [1], and the Statistics and Probability Laboratory, LPE [2], a specific activity known as *"Optimización de Costos Bajo Diseño Robusto en Redes Multi-Overlay"*.

The project mingles two perspectives that in general are difficult to combine. Firstly, the challenge of the economical, result-aimed, deadlined and applied viewpoint involved when an enterprise requests researching services of an investigation institute. On the other hand, several academical aspects have arisen in the way that considerable investigation has been carried out

1. *Administración Nacional de Telecomunicaciones*.
2. Engineering University of the Republic University.

during the development of the project as well as it has inspired many topics for Master's thesis as this one.

1.2. Objective

The problem to solve consists in the design of a Network divided in two layers. The overlay network is a MPLS Data Network which physically exchange traffic over a Transport Network infrastructure. Such design should account for economical resources. In particular, the cost is related to the distance of paths [3] in the transport structure and the capacity of tunnels in the data layer.

It also ought to meet requirements regarding traffic routed and its pertaining parameters of quality. Different kind of traffic has different requirements.

In addition, the network designed must be robust to simple transport edge failures, i.e. failures involving only one edge.

Finally, the solution found not only should be feasible, but also as optimal as possible. Besides, the problem to solve and the algorithms implemented should reflect the particular reality of ANTEL. However, this thesis has taken an holistic point of view with a sufficiently abstract formulation that allows variations in all parameters so that any kind of network could be adopted as input for optimisation.

In brief, the purpose of this work is to find :
- In which stations should be installed MPLS equipment.
- Which links should be created in the Data Network in order to allow for convenient tunnels of data traffic.
- In the last case, which should be the best tunnels between nodes and which should be their capacity.
- For each data link established, which should be the best path in the Transport Network to be mapped.
- Which technology should be used for each transport edge included.
- How to handle different kind of traffic with different quality parameters.
- How to route traffic in each case when only one transport edge fails in the configuration found.

The information for this target is :
- Set of Network Stations (these are typically Telephone Stations).
- Stations where is feasible to install MPLS switches.
- Links between switches that are viable to be established.
- Capacities available for each data link.
- Transport technologies available for each bandwidth and its cost.

3. A path is a sequence of edges.

– Transport Network topology : Networks location, distances between them, optical fibre installed.
– Origin, destination and amount of traffic that enters and exits the network.
– The statistical behaviour of clients traffic.

Summarising, in this thesis we will concentrate on the modelling and resolution of the problem of the design of a survivable IP/MPLS Data Network over ANTEL transport infrastructure. A strategic planning problem for ANTEL and for its data business projection in medium term.

When formally modelling the problem, we noted that its actual specificities were substantially different from other similar models present in relevant state-of-the-art literature researched. From the bibliographical analysis of the existing literature, to the best of our knowledge, no mathematical optimisation model was found which contemplated the same constraints than the model here studied.

It is noteworthy that during 2008 we intensively worked with Dr. Maurice Queyranne from the University of British Columbia, who visited the the project team for about a week. He was of fundamental assistance in the refinement of the non-linear integer programming model associated to the problem. Such model was taken afterwards by Eng. Cecilia Parodi in the context of her Master Thesis.

Other models of multi-overlay optimisation and robust network design appear in the literature for example in [1, 17, 19, 30].

1.3. Documentation organization

This thesis is organised in chapters as follows :

Chapter 1, Introduction, it presents a general introduction to the thematic related to the thesis.

Chapter 2, Problem description, it roughly explains the main features of networks and the multi-overlay scheme for the non-familiarised reader, returning to the problem to solve.

Chapter 3, Formalisation, it describes the high level model used to characterise the networks, interconnections, costs and other variables.

Chapter 4, Mathematical programming formalisation, it refers to the methods used to approach the problem solving from a computational point of view. It begins with a formal definition of entities presented in the previous chapter. Then, it presents an initial combinatorial optimisation formulation with the description of the objective function and its constraints. Next, simplifications are introduced between the data nodes connections leading to another model. Finally, the problem complexity is analysed.

Chapter 5, Non-linear binary integer programming model, it defines binary variables for the problem. It introduces the objective function and constraints. Each constraint is thoughtfully explained.

Chapter 6, VNS description, it mentions the principal aspects of the selected metaheuristic used to solve the problem, fundamental schemes and VNS virtues.

Chapter 7, MORN Heuristic, it gives an overview of the entire metaheuristic employed to reach an optimal solution, building an initial feasible solution in the first place in order to perform local searches next. In particular, it concentrates on the GRASP[4] procedure to obtain the initial feasible solution.

Chapter 8, VNS customisation of the problem, it continues last chapter indicating the way VNS is customised to the problem. Each local search implemented is carefully detailed.

Chapter 9, Test cases description, it shows the functional tests developed to evaluate the system. The base for testing was created from real data provided by the national telephone operator, ANTEL.

Chapter 10, Performance tests and analysis of results, it studies the results achieved contrasting with the real current network operation.

Chapter 11, Conclusions, it analyses the achievements accomplished and mentions the possible course of action to extend the investigations.

4. Greedy Randomised Adaptive Search Procedure.

Chapter 2

Problem description

2.1. Introduction to networks

This chapter begins introducing several general concepts pertinent to the thesis.

2.1.1. OSI Model

In a telecommunication network it is possible to consider certain levels of structures with similar functions in order to make abstractions in design and make different brands compatible.

The OSI [1] Reference Model, divides the network architecture into seven layers as shown in Figure 2.1. The user from the left transmit data to the receiver at the right. Each layer provides services to upper layers, and make use of layers below. In this way, a horizontal protocol between instances of the same level is established.

As an illustration of these concepts, let us take the HTTP protocol which belongs to the application layer. This is the most upper layer and uses the services of all the layers below it to communicate with a server and request a web page of interest. However, it is possible to think of this as a horizontal dialogue between the PC and the server.

Layers can be divided in two groups :

1. Transport services (layers 1, 2, 3 y 4).
2. User support services (layers 5, 6 y 7).

2.1.2. Transport Network

Transport Network deals with submission and multi-channelling of different kind of information in distinct formats, both analogic and digital. Traditionally, its structure and characteristics were dependent on the type of information carried. For example, cable television has its

1. Open System Interconnection.

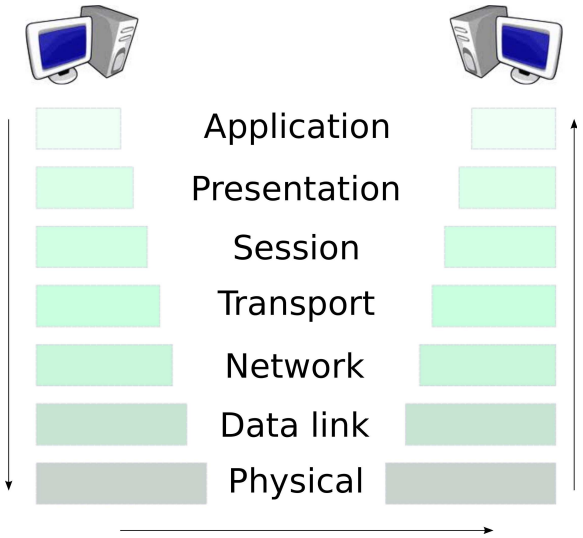


Figure 2.1 – OSI Model Layers

own transport network, as well as mobile or circuit switched telephony has their own.

As long as digitalization appeared, networks started to converge in the sense that they were capable of transporting any kind of information, independent of its origin. For instance, E1/T1 and ISDN, based on the circuit switched telephone network, ATM and SDH, based in optic fibre.

2.1.3. Data Network

Data Network is related to the submission of data. It only takes into account the nodes that exchange traffic disregarding the particular physical path that the packages transferred follow. Because of this, the links between the nodes that have certain demand (*flow*) to send are virtual.

2.1.4. MPLS

MPLS [2] is a telecommunication network mechanism of high performance which combines layer 2 (Data link) and 3 (IP) of the OSI model in order to improve and simplify the interchange of packages in the network. In this way, the knowledge of bandwidth, latency, utilization of links provides to the operator the flexibility to route and divert traffic in case of link failures, congestion, etc.

From the quality of service point of view, MPLS allows to handle different kinds of data streaming based on priorities and contracted services. In such manner, it is possible to offer

2. MultiProtocol Level Switching.

different service plans to the clients, for example, giving priority to higher bandwidth with minimum latency and loss for multimedia services.

2.2. Overlay Networks

2.2.1. Overlay Scheme

An Overlay Network is a network built above another one. The overlay network nodes (above) are connected between virtual links. Each link is associated to a path in the underlay network (below), and that path may consist of several physical links. In the case of the problem related to this thesis, the overlay network corresponds to the Data Network and the underlay network corresponds to the Transport Network.

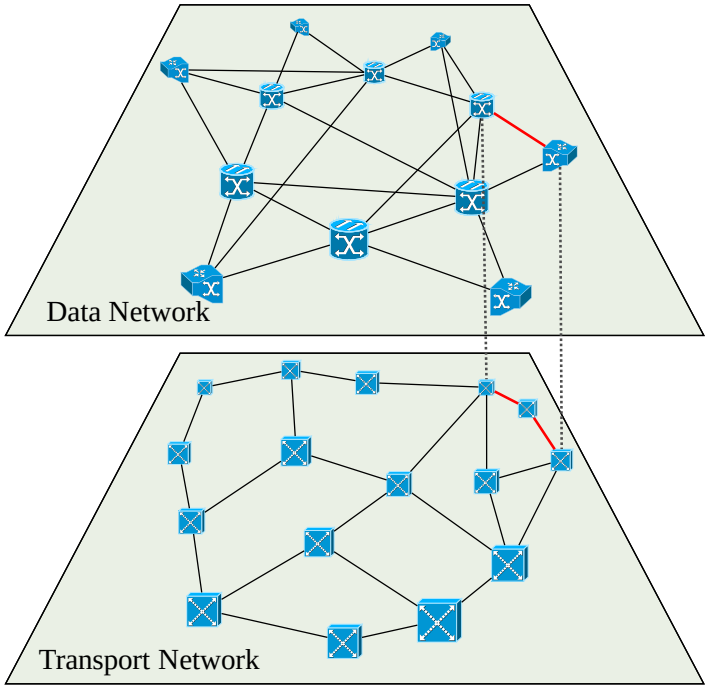


Figure 2.2 – Red Overlay

2.2.2. Examples

Some examples of overlay networks are the following :

Internet It uses the Telephone Network as underlay.

P2P It runs over internet and the nodes are the distinct instances of the application. A point to point tunnel structure is generated to send the information.

ATM The connection between interior nodes use SDH links.

MPLS It is similar to ATM combining other transport technologies such as DWDM.

2.3. The problem

Now that several useful concepts have been described, the main issues concerning the problem itself will be pointed.

The problem is the design of an IP/MPLS Network. This network should have optimal cost and must be robust to simple failures in the Transport Network. Besides, several types of technologies in the Transport Network must be taken into account as well as different kinds of traffic in the Data Network.

Despite the solution implemented is general enough as to deal with any IP/MPLS Network as input and returns a satisfactory optimal design, this thesis has focused on one test case. In particular, the IP/MPLS Network to work on will be the Uruguayan network of ANTEL. Because of this, the Transport Network is already established since ANTEL has its infrastructure spread all over the country. The Data Network with the potential edges to utilise is also given and the traffic between nodes have been relieved.

The design will discard the Uruguayan Metropolitan Region and will concentrate on the remaining part of the country. The reason for this is that the MPLS deploy is being developed in Montevideo by the time of this work and also because the most important cost related to the Transport Network appears when large distances are considered.

Part II

THEORETICAL FRAMEWORK

Chapter 3

Formalisation

3.1. Introduction

In this chapter the main objects and characteristics relevant to the problem are defined, explaining their properties and relations. It is also described the models adopted to represent reality and the simplifications taken and the reasons for being considered.

As previously stated, there are two networks, the Data and the Transport Network. We will model both networks with graphs called $G_D = (V_D, E_D)$ and $G_T = (V_T, E_T)$ respectively.

3.2. Data Network

The Data Network $G_D = (V_D, E_D)$, where V_D stands for the set of nodes and E_D the set of potential virtual links, is a non-directed graph. This is a virtual network. Its edges do not physically exist. It is an abstraction to reperesent and model several aspects of the network easily, hiding the particular real path that the traffic follows, which is a service provided by the network underlay.

This network main characteristics are :
- **Mesh topology :** It comprises of a high quantity of links. It is dense in its structure. No assumptions will be made about planarity.
- **Dynamic traffic :** Either the routes or the volume of traffic are able to vary in time.

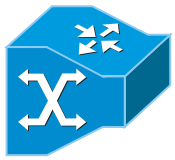
3.2.1. Data nodes

A data node represent commuters IP DSL (DSLAM xDSL) and commuters MPLS. The former are equipment that receive and deliver traffic from and towards final clients. The latter are more powerful equipment. They are able to route traffic towards its destiny, keeping information of the topology of the network and its statics. They also detect link failures or congested routes and are able to reroute traffic appropriately.

Because of this, according to the functionality of the data nodes in the network we consider two kind of data nodes called *access nodes* and *edge nodes* and noted V_A and V_E respectively. Each node correspond to one and only one of these classes, so they constitute a partition of V_D.

Typically, access nodes act as last mile commuters that gather traffic from several clients. Edge nodes are usually connected to several access nodes and other edge nodes.

We will represent access and edge nodes as shown in Fig-3.1.



(a) Access node : DSLAM

(b) Edge node : MPLS

Figure 3.1 – Access and edge data node representation

Another classification of data nodes is between *fixed* and *optional* or *Steiner* nodes, noted V_F and V_S, which also form a partition of V_D. The fixed nodes are those that are obliged to be present in the solution of the problem, for instance, those planned to send or receive traffic. Steiner nodes may be included in the solution or not. Generally, some of them are used as intermediary when there is a benefit for the network topology, i.e. when the network cost is lower.

It is noteworthy that the access nodes are a subset of the fixed nodes as shown in Fig-3.2. Otherwise, in the case an access node was optional and not included in the solution, then clients would not receive a satisfactory service as they will be disconnected to the network.

3.2.2. Data links

Each arch $e_{ij} = (v_i, v_j) \in E_D$ represents a potential link to be established in order to carry traffic either between nodes v_i and v_j, or as an intermediate link in a path between other two nodes of the Data Network. We say they are potential in the sense that their presence in the design is considered but they may be discarded.

One attribute of a link is its capacity. There is a set of capacities available $\hat{B} = \{\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_{\bar{B}}\}$. The set of capacities is determined by the technology of the transport underlay and it is data for the problem. Every link will be dimensioned with a suitable capacity chosen from the set $\hat{B}$ and it is assumed that if a link does not become part of the solution, its capacity is

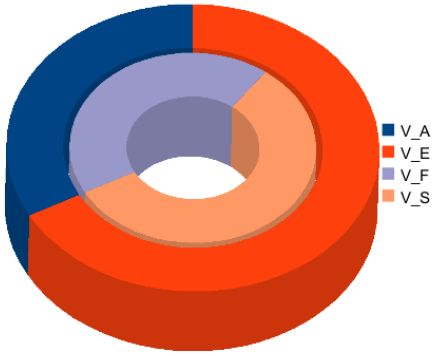


Figure 3.2 – Data node chart

zero (so we make $\hat{b}_0 = 0$). In this way, part of a solution of the problem will be the set $B = \{b_{ij} \in \hat{B}, \forall e_{ij} \in E_D\}$.

It is remarked that knowing the set B, it is known the links included in the solution, so that no other information is needed. On the other hand, it is possible to recognise whether a node v_i integrates the solution when $b_{ij} \neq 0$ for some v_j.

Another point is that in sight that links are non-directional, b_{ij} is the same as b_{ji}.

3.3. Transport Network

Similarly to the Data Network, the Transport Network is noted $G_T = (V_T, E_T)$, where V_T is the set of terminal nodes and E_T is the set of physical links. Once again it will be assumed to be non-directed.

This network main characteristics are :
- **Ring topology :** It consists of several rings that share one or more adjacent links. Consequently, the network is 2-node-connected and also planar.
- **Static traffic :** The traffic is completely static.

3.3.1. Transport nodes

The nodes of the Transport Network represent the *Network Stations*, which are the buildings where the physical telecommunication infrastructure is installed.

Despite the fact that a Network Station compounds several different equipment, the following valid generalisation will be made. It will be considered that a transport node mingles all the distinct technologies (SDH, DWDM, etc.) and merge the diverse characteristics such as

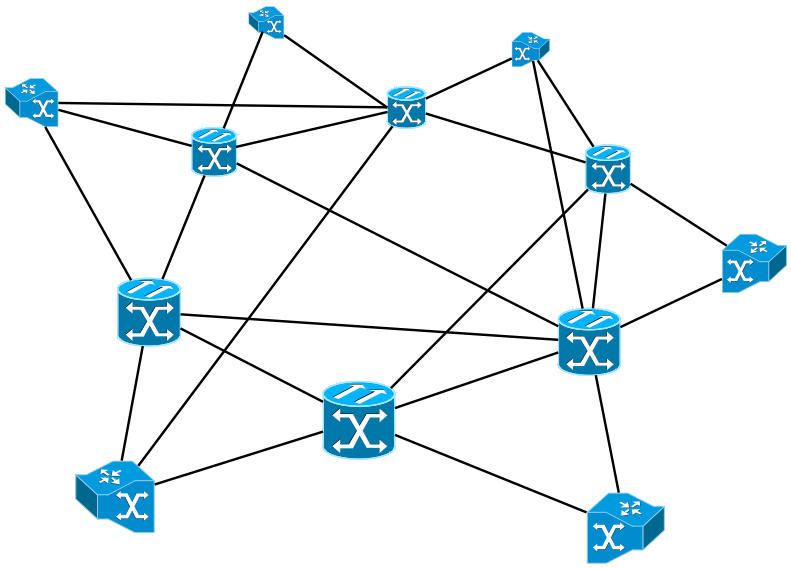


Figure 3.3 – Example : Data Network

ports, bandwidths, in an illimited capacity terminal, which is able to make use of the desired appropriate technology.

We will represent transport nodes as shown in Fig-3.4.

Figure 3.4 – Transport node representation

3.3.2. Transport edges

In this case, the transport edges play the part of the canalisation that connects the Network Stations.

It has been made the assumption to idealise transport edges as having infinite capacity (or bandwidth) in the sense that such a quantity of equipment should be installed as to contemplate any required demand.

Another aspect is that when the event of a simple failure occurs, the data network connections that utilise that canalisation are affected.

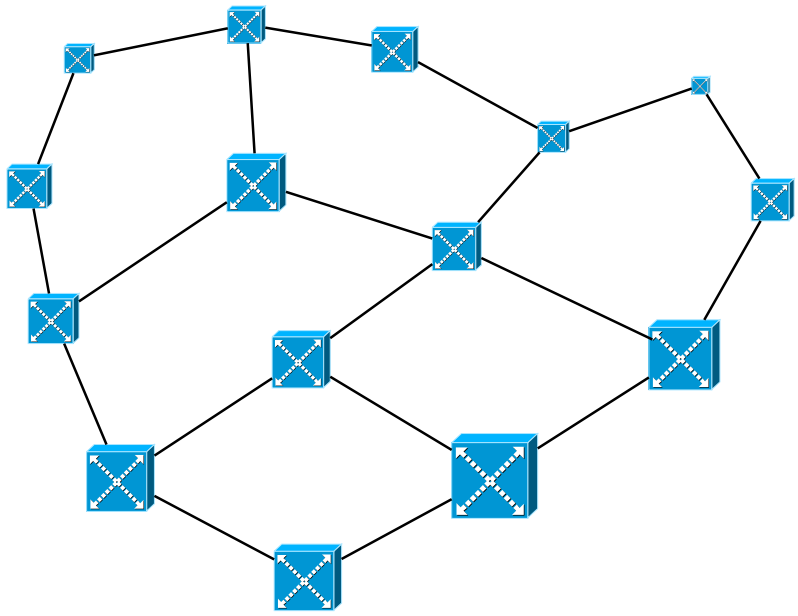


Figure 3.5 – Example : Traffic Network

3.4. Underlay and Overlay relation

In each Network Station there is a lot of equipment. The transport equipment in it is collapsed as one logical transport node. The data nodes planned for the network are installed at the Network Stations. All equipment related to delivering traffic to the clients is also condensed as one ideal access data node. Similarly, the routing MPLS equipment is considered as an edge data node.

Summarising, in a Network Station there is always a transport node. The amount of data nodes is either zero, one or two, and in the last case, both data nodes are of different type (one is access and the other edge). However, it is impossible to conceive a transport or data node outside a network station. We define then the *network station function* as

$$ns : (V_D \cup V_T) \times (V_D \cup V_T) \to \{0, 1\},$$

which complies that $ns(u, v) = 1$ iff both u and v belong to the same station.

Given a data node, it is determined the corresponding transport node belonging to the same station. The *transport network station function*, $tns : V_D \to V_T$ such that $tns(v) = t$ when

$ns(v,t) = 1$, returns the transport node t located in the same network station of v.

On the other hand, we will force each data edge to have a unique transport path associated. Hence, given two data nodes $v_i, v_j \in V_D$, we call ρ_T^{ij} the set of *transport paths* that the flow between $tns(v_i), tns(v_j) \in V_T$ will follow in the Transport Network in a solution of the problem. Notice that the condition that maps a data edge to a unique transport path could be mathematically expressed as $|\rho_T^{ij}| = 1$.

3.5. Attributes of the problem

3.5.1. Traffic

On any communication, a pair of nodes may, or may not exchange traffic. Such traffic will be modelled as being constituted of two components denoted committed traffic and excess traffic :

- **Committed traffic :** It is the traffic that should be exchanged all the time and even in the face of simple transport link failure. It can be thought as a traffic with which there exists a compromise with an enterprise, person, or purpose, so that it must be met 100 % of the time. It also is possible to be regarded with multimedia applications traffic such as *Voice over IP* or *Video on Demand*.
- **Excess traffic :** It is the eventual traffic which is transferred just part of the time. As the fraction of time that traffic is 100 % available is called *quality factor*, it is said that traffic committed has a quality factor of 100 % and excess traffic has a quality factor under 100 %. This percentage does not have to be the same in flawless or flawed scenarios. Commonly, excess traffic is internet traffic and is treated as *best effort*.

The traffic between a pair of nodes $v_i, v_j \in V_D$ is noted $\vec{m}_{ij} = (\dot{m}_{ij}, \ddot{m}_{ij})$. Clearly the couple correspond to committed, $\dot{m}_{ij}$, and excess, $\ddot{m}_{ij}$, traffic.

A fancy artefact we introduce to store the information of traffic is called the traffic matrix. It contains the traffic vectors for all data nodes. Mathematically, the matrix is $\bar{M} = ((\vec{m}_{ij}))_{1 \leq i,j \leq |V_D|}$, and is data of the problem. Its entries are known because they represent the demand of the terminals, the contracted or up/downloaded by the clients.

As the only nodes with requirements of traffic exchange are the fixed ones, is evident that if $\vec{m}_{ij}$ is non zero, then $v_i, v_j \in V_F$.

Finally, we remark that the traffic considered here is nominal, or “sold”, while the real effective traffic depends on several factors and can even be treated as aleatory.

3.5.2. Cost

The aim of the project that enclose this thesis is contributing to reduce the cost of ANTEL business related to the network structure. The Transport Network plays an important role for

such purpose as we analyse in this section.

When it comes to the cost of the Transport Network we must calculate the cost of each transport path. Thus, two possible technological alternatives must be revised :

– *TDM*[1] *technology :* The cost depends on both distance between stations and bandwidth selected –or the linear assumption used in calculations–. It is clear that the farther apart the nodes are, the excavation to lay the fibers is more expensive. The reason why cost is proportional to bandwidth is that TDH reserves containers for traffic of different sizes. This means that the more a flow loads a link, the more resources it consumes.

  Summarizing, the calculated cost of transmission of a flow regarding traditional technologies (PDH/SDH)[2] has the expression :

$$cost(\rho_T^{ij}, b_{ij}) = k \times r(\rho_T^{ij}) \times b_{ij}$$

  where k is a constant parameter, $r(\rho_T^{ij})$ is the distance of the path ρ_T^{ij} in the Transport Network that and b_{ij} is the capacity assigned to the edge (v_i, v_j) in the Data Network.

– *DWDM*[3] *technology :* The cost depends only on distance. This happens due to the fact that in DWDM, a particular wavelength is chosen according to the bandwidth required for a link. Several connections can be established over the same physical link using a different value of wavelength. So that, the only relevant magnitude is distance :

$$cost(\rho_T^{ij}, b_{ij}) = k \times r(\rho_T^{ij}).$$

In sight of the aforementioned, both technological alternatives have advantages depending on the capacity to install. The graphic in Fig-3.6 combines this information, showing that according to the capacity needed it is convenient to make use of one particular technology or the other. If the capacity required is small, then SDH is convenient, however, if a high capacity link is needed, then DWDM is the choice.

No other constrains are imposed with respect to technologies in order to keep it simple. Any technology is valid for any distance. The cost is proportional to distance. A detail to note is the assumption that each transport edge of a path has the same assigned capacity.

The most inexpensive option to pick depends on the bandwidth needed. Consequently, with the intention to overcome the difficulty of handling different technologies, it is adopted the formalism of a function T which returns the minimal cost per kilometre of the suitable technology according to the bandwidth required.

So that, the cost is calculated :

$$cost(\rho_T^{ij}, b_{ij}) = r(\rho_T^{ij}) \times T(b_{ij})$$

1. Time-Division Multiplexing.
2. Plesiochronous/Synchronous Digital Hierarchy.
3. Dense Wavelength Division Multiplexing.

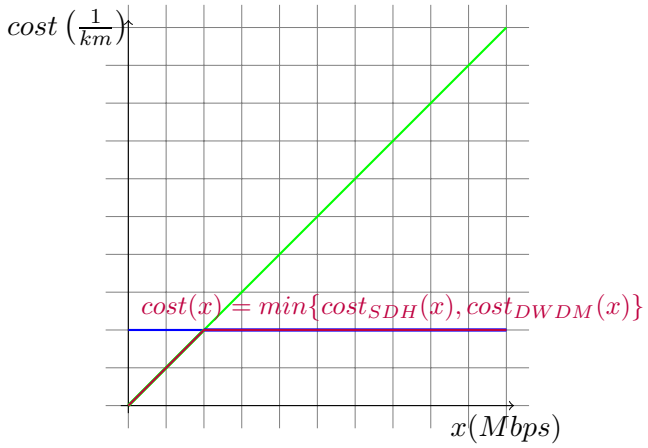


Figure 3.6 – Cost function

The procedure followed to simplify the model is based on focusing on the design of the topology, leaving the technology hidden in the cost.

3.5.3. Traffic routing

Let a traffic matrix $\bar{M}$. The design of the network must explicitly answer how the traffic goes from a node v_i to a node v_j, with demand $\vec{m}_{ij} \neq \vec{0}$. This is to say which are the paths of data links that the traffic will always follow for those nodes. This fixed routes from v_i to v_j in order to send traffic are called *tunnels*.

We embrace the totality of possible tunnels in G_D in a set noted P_D. Then, a *routing scenery*, ρ_D, is a set of paths in G_D, or a subset of P_D.

The routes from a data node v_i to a node v_j in a routing scenery are all the routes in ρ_D which begin in v_i and end in v_j.

Example.

Let a Data Network as shown in Fig-3.7, with the following sets of nodes, edges and routes :
$V_D = \{v_1, v_2, v_3, v_4, v_5, v_6\}$.
$E_D = \{(v_1, v_2), (v_1, v_3), (v_1, v_6), (v_2, v_3), (v_2, v_5), (v_2, v_6), (v_3, v_4), (v_3, v_5), (v_4, v_5), (v_5, v_6)\}$.
$\rho_D = \{\{(v_1, v_2)\}, \{(v_1, v_3)\}, \{(v_1, v_2), (v_2, v_3)\}, \{(v_2, v_3)\}, \{(v_2, v_6)\},$
$\{(v_2, v_3), (v_3, v_5), (v_5, v_6)\}, \{(v_3, v_4), (v_4, v_5)\}, \{(v_4, v_5)\}, \{(v_5, v_6), (v_6, v_1)\}\}$.

In this example, the possible routes from the data node v_1 to v_3 were encountered to be $\{(v_1, v_3)\}$ and $\{(v_1, v_2), (v_2, v_3)\}$. ♠

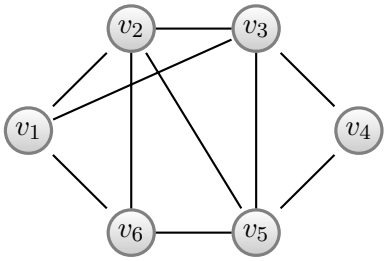


Figure 3.7 – Routing example

3.5.4. Robustness

One major aspect that turns the problem presented in this work more complicated is the characteristic of robustness to simple transport failures that it is required in the design of the network.

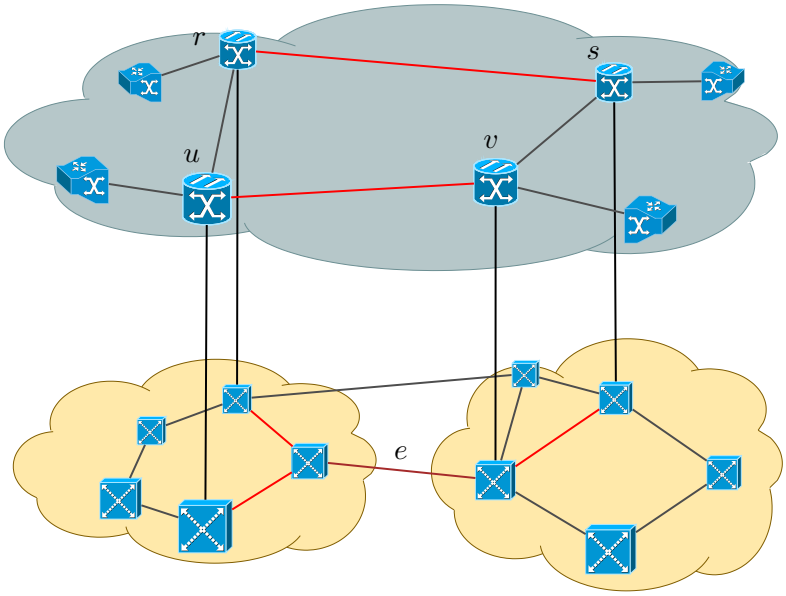


Figure 3.8 – Simple transport failure

A transport failure usually occurs when optical fibre is unintentionally cut by heavy machinery performing road maintenance works. This translates to a lost edge in the Transport Network, which generally will be reflected in the Data Network as several data links failing. We will make use of Fig-3.8 to illustrate a transport edge failure. If the transport edge $e \in E_T$ fails, the data links $u - v$ and $r - s$ that were mapped to a transport path which included e will also fail. As a result, all the tunnels which exchanged traffic through those edges will be down.

Even in every simple failure event, all the demands of traffic must be met. This means that new data tunnels must be established with a proper capacity, routed through feasible transport paths in the remaining Transport Network.

We expect by now the reader has already attained a major comprehension of the complete problem. In next chapter, the notions of entities, relations and restrictions previously introduced in a descriptive manner will be formalised with definitions and equations arriving to a combinatorial optimisation formulation.

Chapter 4

Mathematical programming formalisation

4.1. Complete model

In this chapter, a mathematical point of view is adopted and the concepts already presented are formalised in definitions. Despite of how heavy this might seem, the reader at this moment is expected –hopefully– to have grasped such an insight on the problem as to relieve him of an exceedingly understanding effort.

4.1.1. Definitions

Definition 4.1.1 *$G_D = (V_D, E_D)$ and $G_T = (V_T, E_T)$ are two non-directed graphs modelling the Data Network and the Transport Network respectively.*

Definition 4.1.2 *Let G_D the Data Network, there exist two classifications for data nodes :*

- *access nodes V_A and edge nodes V_E, that form a partition of the set of data nodes, i.e. $V_D = V_A \cup V_E$ and $V_A \cap V_E = \emptyset$.*
- *fixed nodes V_F and Steiner nodes V_S, that form a partition of the set of data nodes, i.e. $V_D = V_F \cup V_S$ and $V_F \cap V_S = \emptyset$.*

Property 4.1.3 $V_A \subset V_F$.

Definition 4.1.4 *Let $V_D = \{v_1, v_2, \ldots, v_h\}$, there exists a traffic matrix $\bar{M} = ((\vec{m}_{ij}))_{1 \leq i,j \leq h}$ where the vectors $\vec{m}_{ij} = (\dot{m}_{ij}, \ddot{m}_{ij}) \in \{\mathbb{R}_0^+ \times \mathbb{R}_0^+, \forall v_i, v_j \in V_D\}$. $\dot{m}_{ij}$ is the committed traffic between the nodes v_i and v_j while $\ddot{m}_{ij}$ is the excess traffic between them.*

Property 4.1.5 *It is imposed to be consequent with reality that $\bar{M}$ is symmetric, i.e. $\dot{m}_{ij} = \dot{m}_{ji}$ and $\ddot{m}_{ij} = \ddot{m}_{ji}$. Besides $m_{ij} \neq \vec{0}$ only when $v_i, v_j \in V_F$.*

Definition 4.1.6 *Let $E_D = \{(v_i, v_j) : 1 \leq i, j \leq h\}$ the set of data links, then $\hat{B} = \{\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_{\bar{B}}\}$ where $0 = \hat{b}_0 < \hat{b}_1 < \cdots < \hat{b}_{\bar{B}}$, is the set of possible capacities for the links of the data graph.*

The problem of choosing the capacities of the links is formulated as finding the function $b : E_D \to \hat{B}$, where we note $b(v_i, v_j) = b_{ij} \in \hat{B}$, i.e. the set $B = \{b_{ij} \in \hat{B} : e_{ij} \in E_D\}$.

It has been remarked that in each network station there is always a transport node and sometimes a data node. This inspire the following :

Definition 4.1.7 *Let* $ns : (V_D \cup V_T) \times (V_D \cup V_T) \to \{0, 1\}$ *the* network station function*, which* $ns(u, v) = 1 \iff$ *both* u *and* v *belong to the same station.*

It is clear that $ns(u, v) = 0 \,\forall\, u, v \in V_T$, and $\forall v \in V_D$, $\exists t \in V_T \,/\, ns(v, t) = 1$.

Definition 4.1.8 *Let* $tns : V_D \to V_T$ *the* transport network station function *such that* $tns(v) = t$ *when* $ns(v, t) = 1$.

Definition 4.1.9 *Let* $G_D = (V_D, E_D)$*,* P_D *is the set of all possible paths in the graph* G_D*. We name* routing scenario *to every subset* $\rho_D \subset P_D$*, and* $g_D : V_D \times V_D \to 2^{P_D}$ *is the* routes function *which determines all possible paths between two pair of data nodes.*

Analogously,

Definition 4.1.10 *Let* $G_T = (V_T, E_T)$*,* P_T *is the set of all possible paths in the graph* G_T*. We name* flow configuration *to every subset* $\rho_T \subset P_T$*, and* $g_T : V_T \times V_T \to 2^{P_T}$ *is the* flows function *which determines all possible paths between two pair of transport nodes.* [1]

Definition 4.1.11 *Let* $G_D = (V_D, E_D)$*, a routing scenario* $\rho_D \in P_D$ *and two data nodes* $v_i, v_j \in V_D$*, the* routes from v_i to v_j *are the elements of the set* $\rho_D^{ij} = g_D(v_i, v_j) \cap \rho_D$*.*

Definition 4.1.12 *Let* $G_T = (V_T, E_T)$*,* $G_D = (V_D, E_D)$*, a flow configuration* $\rho_T \subset P_T$ *and a data link* $e_d \in E_D$ *such that* $e_d = (v_i, v_j)$*,* $v_i, v_j \in V_D$*, the* flow implementation for e_d *are the elements of the set* $\rho_T^{ij} = g_T(tns(v_i), tns(v_j)) \cap \rho_T$*.*

Definition 4.1.13 *Let the networks* $G_D = (V_D, E_D)$ *and* $G_T = (V_T, E_T)$*, we call* routing scenarios function *to* $\Phi : (E_T \cup \emptyset) \to 2^{P_D}$ *such that each* $e_t \in E_T$*, and the empty set, is assigned to a routing scenario* $\rho_D \subset P_D$*. In particular,* $\Phi(\emptyset)$ *is denoted* nominal routing scenario*.*

Definition 4.1.14 *Let the networks* $G_D = (V_D, E_D)$ *and* $G_T = (V_T, E_T)$*, we call* flows configuration function *to* $\Psi : E_D \to 2^{P_T}$ *such that each* $e_d \in E_D$ *is assigned to a flow configuration* $\rho_T \subset P_T$*.*

Definition 4.1.15 *Let* $T : \hat{B} \to \mathbb{R}$ *the* technology cost function *such that it assigns to each available capacity its cost per length unit. It is assumed that* $T(\hat{b}_0) = T(0) = 0$

1. The *power set* of A, 2^A, is the set of all subsets of A.

Definition 4.1.16 *Let $r : E_T \to \mathbb{R}$ the* distance function *such that it assigns to each transport link its length. It is possible to extend the domain of the function to any flow implementation ρ_T^{ij}, considering the sum of distances of its links, i.e. $r : E_T^* \to \mathbb{R}$ where $r(\rho_T^{ij}) = \sum_{e \in \rho_T^{ij}} r(e)$. It is assumed that $r(\emptyset) = 0$.*

Definition 4.1.17 *Let $cost : P_T \times \hat{B} \to \mathbb{R}$ the* cost function *such that $cost(\rho_T^{ij}, b_{ij}) = r(\rho_T^{ij}) \times T(b_{ij})$ returns the cost incurred in the Transport Network for a flow from data node v_i to v_j, which follows the path ρ_T^{ij}, and has capacity b_{ij}.*

Definition 4.1.18 *Let $z_Q : \mathbb{R} \to \mathbb{R}$ the* excess traffic capacity function *such that it assigns to each bandwidth sold, $\ddot{m}_{ij}$, the minimum required capacity for the links which carry that traffic.*

4.1.2. Mathematical model

Once the formal definition of abstract objects that represent reality has been made, their relations and constraints are presented.

Firstly, we identify the data of the problem :

- Data Network :
  - $\boldsymbol{G_D = (V_D, E_D)}$: Data Network graph, nodes and links,
  - $\boldsymbol{V_A}$: set of access nodes,
  - $\boldsymbol{V_E}$: set of edge nodes,
  - $\boldsymbol{\bar{M}}$: traffic matrix, and
  - $\boldsymbol{z_Q}$: excess traffic capacity function.
- Transport Network :
  - $\boldsymbol{G_T = (V_T, E_T)}$: Transport Network graph, nodes and links,
  - $\boldsymbol{\hat{B}}$: capacities available,
  - $\boldsymbol{T}$: technology cost function,
  - $\boldsymbol{r}$: distance function, and
  - $\boldsymbol{ns}$: network station function.
  - $\boldsymbol{tns}$: transport network station function.

Secondly, we remind that the variables of the problem are :

- $\mathbf{B}$: assigned capacities for the data links,
- $\boldsymbol{\rho_D}$: data routes, and
- $\boldsymbol{\rho_T}$: transport paths.

At this point we have the necessary elements to expose the complete optimisation model, that is henceforth referred as MORN (Multi-Overlay Robust Network) and has the following mathematical form :

$$
\begin{cases}
\min\limits_{B,\Phi,\Psi} \displaystyle\sum_{\substack{\rho_T=\Psi(e)\\ e=(v_i,v_j)\\ e\in E_D}} r(\rho_T^{ij})T(b_{ij}) & \\
|\rho_T^{ij}| = 1 & \forall e\in E_E, e=(v_i,v_j), b_{ij}\neq 0, \rho_T=\Psi(e). \\
|\rho_T^{ij}| = 1 & \forall e\in E_D, e=(v_i,v_j), v_i\in V_A, v_j\in V_E, b_{ij}\neq 0, ns(v_i,v_j)=0, \rho_T=\Psi(e). \\
\displaystyle\sum_{\substack{b_{ij}\neq 0\\ \forall v_j\in V_E}} 1 = 1 & \forall v_i\in V_A. \\
\displaystyle\sum_{\substack{b_{ij}\neq 0\\ \forall v_j\in V_A}} 1 = 0 & \forall v_i\in V_A. \\
b_{pq} \geq \displaystyle\sum_{\forall v_j\in V_D} \dot{m}_{pj} + z_Q \left(\sum_{\forall v_j\in V_D} \ddot{m}_{pj} \right) & \forall v_p\in V_A, v_q\in V_E, b_{pq}\neq 0. \\
\vec{m'}_{ij}^{\,\emptyset} = \vec{m}_{ij} + \displaystyle\sum_{\substack{b_{ei}\neq 0\\ \forall v_e\in V_A}} \vec{m}_{ej} + \sum_{\substack{b_{fj}\neq 0\\ \forall v_f\in V_A}} \vec{m}_{if} + \sum_{\substack{b_{ei}\neq 0\\ \forall v_e\in V_A}} \sum_{\substack{b_{fj}\neq 0\\ \forall v_f\in V_A}} \vec{m}_{ef} & \forall v_i, v_j\in V_E. \\
\vec{m'}_{ij}^{\,t} = \vec{m}_{ij} + \displaystyle\sum_{\substack{\Psi(ei)\cap\{t\}=\emptyset\\ b_{ei}\neq 0\\ \forall v_e\in V_A}} \vec{m}_{ej} + \sum_{\substack{\Psi(fj)\cap\{t\}=\emptyset\\ b_{fj}\neq 0\\ \forall v_f\in V_A}} \vec{m}_{if} + \sum_{\substack{\Psi(ei)\cap\{t\}=\emptyset\\ b_{ei}\neq 0\\ \forall v_e\in V_A}} \sum_{\substack{\Psi(fj)\cap\{t\}=\emptyset\\ b_{fj}\neq 0\\ \forall v_f\in V_A}} \vec{m}_{ef} & \forall v_i, v_j\in V_E, \forall t\in E_T. \\
|\rho_{D_t}\cap t| = 0 & \forall t\in E_T, \rho_{D_t}=\Phi(t). \\
|\rho_{D_t}^{ij}| = 1 & \forall v_i, v_j\in V_E, \forall t\in\{\emptyset\cup E_T\}, \rho_{D_t}=\Phi(t), \vec{m'}_{ij}^{\,t}\neq\vec{0}. \\
b_{pq} \geq \displaystyle\sum_{\forall v_i, v_j\in V_E} \left(\dot{m'}_{ij}^{\,t} | \rho_{D_t}^{ij}\cap e| \right) + z_Q \left(\sum_{\forall v_i, v_j\in V_E} \left(\ddot{m'}_{ij}^{\,t} | \rho_{D_t}^{ij}\cap e| \right) \right) & \forall e\in E_E, e=(v_p,v_q), \forall t\in\{\emptyset\cup E_T\}, \rho_{D_t}=\Phi(t), \vec{m'}_{ij}^{\,t}=(\dot{m'}_{ij}^{\,t}, \ddot{m'}_{ij}^{\,t}).
\end{cases}
$$

Below, each constraint is exhaustively described.

– The first term is the objective function to optimise. It consists of a sum of costs that must be minimised. For that purpose, a tern of functions (B, Φ, Ψ) must be obtained.

  In each term of the sum it is added the transport cost of each data link $e \in E_D$ included in the solution. As previously stated, the cost is expressed as the product of the cost of the path that the flow follows $r(\rho_T^{ij})$ and the cost of the corresponding capacity selected for the path $T(b_{ij})$ per kilometre.

  The rest of the equations are constraints.

– The second equation is the first constraint. It points out that each link between edge

nodes, $e \in E_E$, that is part of the solution ($b_{ij} \neq 0$), must have a unique associated flow in G_T.

- The second constraint indicates that every data link between an access node $v_i \in V_A$ and an edge node $v_j \in V_E$, that is part of the solution ($b_{ij} \neq 0$), must be implemented with two flows, if the nodes are in different stations ($ns(v_i, v_j) = 0$).
- The third constraint establishes that each access node $v_i \in V_A$, is connected to only one edge node in the solution. In the sum only the edge effective neighbours, those $v_j \in V_E$ with $b_{ij} \neq 0$, are counted and the amount must be one.
- The fourth constraint expresses that no access node $v_i \in V_A$, is connected to another access node in the solution. In the sum only the access effective neighbours, those $v_j \in V_A$ with $b_{ij} \neq 0$, are counted and the amount must be zero.
- The fifth constraint means that the connection from an access node ($v_p \in V_A$), to its corresponding edge node ($v_q \in V_E, b_{pq} \neq 0$), must have capacity enough to satisfy the sum of demands, either committed or excess requirements to any other network node.
- The sixth constraint defines the entries of the traffic matrix $\bar{M'}_0$. The effective demand between nodes $v_i, v_j \in V_E$ comprises of its own ($\vec{m}_{ij}$), plus the demands of the access nodes connected to v_i ($v_e \in V_A$ such that $b_{ei} \neq 0$) or v_j ($v_f \in V_A$ such that $b_{fj} \neq 0$). The access nodes transfer traffic among them or towards these edge nodes, through the tunnel established between the latter.
- The seventh constraint is equivalent to the previous one, although considering all the failure scenarios ($t \in E_T$). Here, those access nodes which associated transport path intersect the link down are not taken into account.
- The eighth constraint implies that in any event of a simple traffic link failure ($t \in E_T$), the proposed routing ($\rho_{D_t} = \Phi(t)$), must not evidently employ that link in the solution ($|\rho_{D_t} \cap t| = 0$).
- The ninth constraint claims that at every possible scenario, either nominal or with a link down ($t \in \{\emptyset \cup E_T\}$, corresponding routing output ($\rho_{D_t} = \Phi(t)$), must contain a unique tunnel between any two edge nodes $v_i, v_j \in V_E$ with effective demand of traffic between them for that particular scenario ($\vec{m'}^t_{ij} \neq \vec{0}$).
- The last constraint is similar yet a little more complicated than the fifth one and involves edge nodes. In view that a link between edge nodes carries not only traffic between them, but also traffic exchanged between other nodes for the scenario $t \in \{\emptyset \cup E_T\}$, some details are added.

  The notion is that for every tunnel for this scenario ($\rho^{ij}_{D_t}$) which goes through the link e (i.e. $|\rho^{ij}_{D_t} \cap e| = 1$) [2], the corresponding contributions of demanded traffic are summated.

2. Remember that simple tunnels do not repeat links.

4.2. Solved Access-Edge connection model

This section explains the hypothesis made, the model created, and the solution proposed to connect the access nodes (V_A) to edge nodes (V_E).

An access node is connected to only one data node. If that other node was an access node, it would be impossible for the both nodes to send traffic to other nodes apart from them, because no other connections are allowed for them and they would be isolated from the rest. So that, an access node should always be connected to an edge node.

Even though an access node is connected to only one edge node, there is no constraint about the quantity of connections through different transport paths.

If the access and edge nodes are in the same Network Station, then the connection is made locally. However, the connections access-edge that make use of the Transport Network are of "non-critical" places in sight that important places has both access and edge nodes.

An essential point to study is whether the access-edge connections would need to be protected to transport failures or not. If the first one was the model to apply, at least two independent transport paths, i.e. edge-disjoint paths, would be needed from an access node to its corresponding edge node.

Generally, as the topology we work on is composed of joined rings, in order to get to a near edge node through two independent transport paths one has to go round both arcs of the ring. This has several disadvantages. Firstly, the cost is superior compared with the option in which both flows follow the same shortest path. This is even worse taking into account that the alternative path can be as large as 100 times the shortest path. Secondly, the capacity of the ring that contains both nodes gets totally consumed.

Station	$\hat{\mathbf{b}} \in \hat{\mathbf{B}}$
A	322
B	30
C	8
D	20
E	184

Table 4.1 – Example : Disbalanced stations and capacities.

On the other hand, let us imagine a model with access-edge connections unprotected where an associated edge node is chosen according to the nearest neighbour criteria. Then, as Table-4.1 and Fig-4.1 illustrate in an example of 5 nodes A, B, C, D and E, the access nodes B and C get associated with edge node A, because it is closer in distance than E, however node

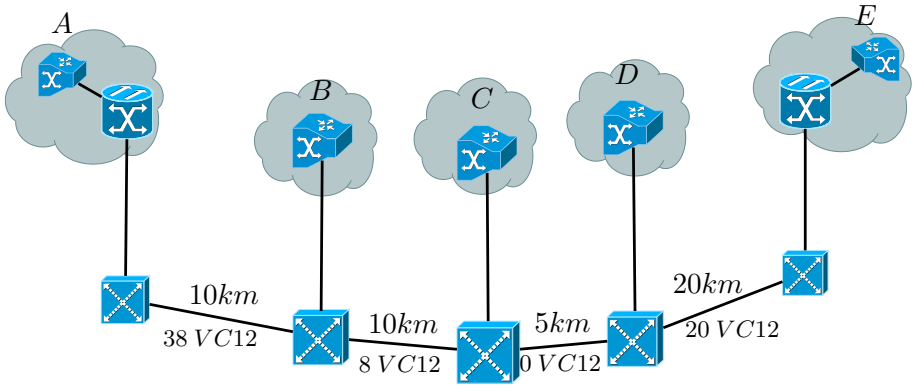


Figure 4.1 – Example : Disbalanced Transport Network

D gets related to E. Notice that as A and E nodes are important and have much more traffic to send, there is an edge node in the same Network Station. We can see that the transport link corresponding to $A-B$ concentrate the traffic from B and C, while transport link $C-D$ has no charge. We can conclude that following this procedure, the charge in the Transport Network would get disbalanced and an error of precision in the calculation of the cost of the network would appear, as Def-4.1.17 assumes balanced utilisation of the capacity, meaning that each transport edge of a path carries a similar amount of flow. Besides, it is important that the path between two important stations would have the same capacity. Consequently, the capacity for the path $A-E$ would consist of 4 parts of $38\ VC12$, summing a total of $152\ VC12$, but the real charge of the path would be $38+8+20=66\ VC12$, which is less than half charge (43 %). It gets worse as more quantity of access are in the middle.

After the ideas handled above, we now summarise the assumptions introduced for solving the access-edge connections and its benefits. The first step consist of finding a way to assign the demand of access nodes to edge nodes in order to get rid of the former. This means that afterwords we will work with a reduced perturbed network, where the access nodes have been eliminated and its associated demands transferred to edges nodes.

In the interest of facing the disbalancing of capacities, we will conceive the splitting of the access node at a station in two access nodes with half the real demand. Each of them connected to both closer edge nodes. In Table-4.2 and Fig-4.2 we continue the previous example. Nodes B, C, and D send half their demand to A and E, leaving the same traffic utilization in every transport edge, and a balanced inter-edge situation. The capacity of the edges then should be designed with four parts of $29\ VC12$, which does not fall apart in a mayor extent from the design with the connection to only one nearest neighbour [3]. So with this perturbation, the charge of the Transport Network does not vary too much, and it is also expected that the solution for G_E does not change either, or if it does, with a minimum impact.

3. It changes about 24 % in this example.

Station	$\hat{\mathbf{b}} \in \hat{\mathbf{B}}$	Station	$\hat{\mathbf{b}} \in \hat{\mathbf{B}}$
B_1	15	B_2	15
C_1	4	C_2	4
D_1	8	D_2	8

Table 4.2 – Example : Balanced stations and capacities.

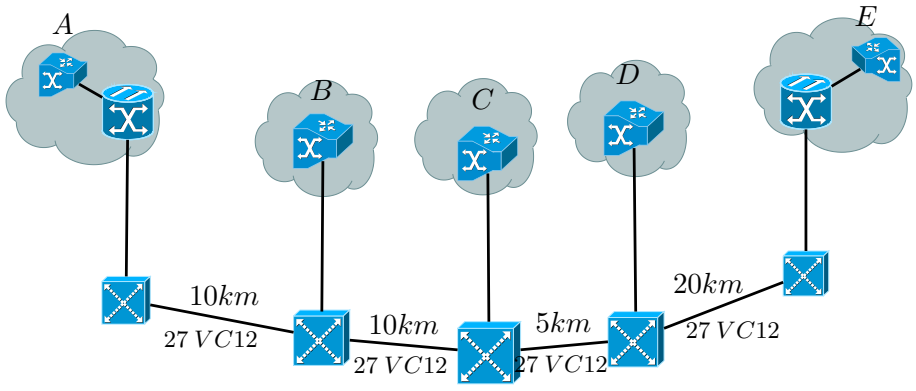


Figure 4.2 – Example : Balanced Transport Network

Not only the utilization of a two port LAG on every access node connected to two different edge nodes shows high disponibility in services but it also allows better and cheaper configurations than the option of considering two independent transport paths. As transport failures are only one manifestation of the multiple ways the network can fail, we change the formulation of the problem relaxing the constraints that regards access nodes robustness. In case of a simple transport failure, the solution of the problem should be able to route all the traffic in every link of G_E not affected by the failure, with the exception of those access nodes that have actually been affected. This is to say that we cannot extend the tolerance to failures in access-edge connections.

Taking into account the above analysis, the model we will work on has the Access-Edge connection solved. In other words, we are now able to omit the access nodes, obtaining a simplification of the problem. The resulting network has only edge nodes with traffic associated to them which corresponds to the access nodes they condense. Moreover, the traffic matrix is now unique for every scenario thanks to the fact that a transport edge failure does not modify the demand of data nodes any more. This reduces the difficulty in the construction of algorithms, without losing generality of the model, precision of results, or representation of the reality.

From now on, the preceding assumptions will be taken and we will name the problem as reduced-MORN.

4.2.1. Reduced-MORN mathematical model

This simplified model deletes several equations leading to a formulation with the form :

$$
\begin{cases}
\min\limits_{B,\Phi,\Psi} \displaystyle\sum_{\substack{\rho_T=\Psi(e)\\ e=(v_i,v_j)\\ e\in E_D}} r(\rho_T^{ij})T(b_{ij}) & \\
|\rho_T^{ij}| = 1 & \forall e\in E_E, e=(v_i,v_j),\ b_{ij}\neq 0, \rho_T=\Psi(e). \\
|\rho_{D_t} \cap t| = 0 & \forall t\in E_T, \rho_{D_t}=\Phi(t). \\
|\rho_{D_t}^{ij}| = 1 & \forall v_i,v_j\in V_E,\ \forall t\in\{\emptyset\cup E_T\},\ \rho_{D_t}=\Phi(t), \vec{m}_{ij}\neq\vec{0}. \\
b_{pq} \geq \displaystyle\sum_{\forall v_i,v_j\in V_E} \left(\dot{m}_{ij}|\rho_{D_t}^{ij}\cap e|\right) + z_Q\left(\sum_{\forall v_i,v_j\in V_E}\left(\ddot{m}_{ij}|\rho_{D_t}^{ij}\cap e|\right)\right) & \forall e\in E_E, e=(v_p,v_q),\ \forall t\in\{\emptyset\cup E_T\},\ \rho_{D_t}=\Phi(t),\ \vec{m}_{ij}=(\dot{m}_{ij},\ddot{m}_{ij}).
\end{cases}
$$

We can see that we achieve a much simpler mathematical model. It consists of finding a minimal cost network with :

- One path in G_T for routing every edge in G_E.
- Routing schemes for every simple failure scenario without using the affected transport edge.
- A tunnel in G_E between any two data nodes $v_i, v_j \in V_E$ that exchange traffic in nominal or failure scenarios of G_T.
- A selection of capacities for links in E_E where the requirements of demand for the chosen tunnel configuration is granted.

4.3. Complexity

The aim of this section is to provide the reader the fundamentals for algorithm complexity comprehension. It is shown that for several computational problems it is possible to find an efficient algorithm to solve it, while there exist other kind of problems so difficult that require too much time to find the optimal solution. As MORN is one of this problems, it is finally proved that there is no known polynomial time bounded algorithm that can solve every instance of MORN to optimality. In order to be precise, we will demonstrate that MORN is a **NP-Hard** problem.

4.3.1. Introduction

The algorithms complexity is measured in terms of *time* and *space* complexity. Both are relative to the given number of inputs, usually expressed as the problem "size". The time complexity concerns the number of elementary arithmetic operations required to execute an algorithm. The space complexity regards the amount of elementary objects that the program demands to

store during its execution.

The difficulty of a problem is related to its structure and the size of the instance to be considered. In graph theory, this size is generally the number of vertices of the graph.

Complexity theory is applied to decision problems which have an answer either *YES* or *NO*. A decision problem π consists of a set of instances D_π and a subset of instances $Y_\pi \subset D_\pi$ for which the answer is *YES*.

Combinatorial optimization problems like our are those with a feasible solution space finite though with large cardinality [24]. Due to the fact that it is possible to correlate these problems with decision problems, the theory is suitably applicable [9].

4.3.2. P and NP

Definition 4.3.1 *A problem $\pi \in$ **P** if there exists an algorithm of polynomial complexity to solve it (the number of steps is less than a polynomial function in n, the input size).*[4] *In this case we say that the problem is tractable.*

Definition 4.3.2 *A problem $\pi \in$ **NP** if it is possible to verify in polynomial time that an instance of* YES *is right.*

Although problems in **P** allow an efficient algorithm, nothing is said about how complicated it is to find a solution for a problem in **NP**. Clearly, $\mathbf{P} \subset \mathbf{NP}$, as if it is viable to solve it in polynomial time, a solution must be checked in polynomial time. However, while most researchers believe the opposite, the open conjecture to present time is whether $\mathbf{P} = \mathbf{NP}$.[5] The theory explained in the following section is even based in the assumption that the inequality holds.

4.3.3. NP-completeness

A key idea in **NP** theory is polynomial transformation :

Definition 4.3.3 *A polynomial reduction from decision problem π' to π, is a function $f : D_{\pi'} \to D_\pi$ that can be computed in polynomial time and for every $d \in D_\pi$, $d \in Y_{\pi'} \iff f(d) \in Y_\pi$. We note $\pi' \preccurlyeq \pi$ if there exist a polynomial transformation from π' to π.*

Observe the significance of polynomial reduction in the following lemma :

Lemma 4.3.4 *If $\pi' \preccurlyeq \pi$, then $\pi \in$ **P** implies $\pi' \in$ **P** (and equivalently $\pi' \notin$ **P** implies $\pi \notin$ **P**).*

4. Regarding computer models, a problem is in **P** if there exists a DTM (Deterministic Turing Machine) of polynomial complexity that solves it.

5. And a valuable one as it is one of the Millennium Prize Problems.

The implications of the lemma is that one has the elements to determine the difficulty of any given problem, just by finding another equivalent complexity-known problem and establishing a polynomial reduction with the former.

Resuming the argument, if **P** $\neq$ **NP**, then there exist problems that are intractable, this means that they are either undecidable (there is no algorithm able to solve it), or require exponential time to solve.

The **NP-Complete** problems are regarded as the hardest problems in **NP** because every other problem in **NP** can be reduced by a polynomial time bounded transformation into them.

Mathematically, a problem π is **NP-Complete** if :

1. $\pi \in$ **NP**
2. For every $\pi' \in$ **NP**, $\pi' \preccurlyeq \pi$.

Occasionally, it cannot be shown that a problem belongs to **NP**, but verifies condition 2. So the problem is said to be **NP-Hard**, meaning that is at least as difficult as all the problems in **NP**.

Owing to the fact that transitive property holds in polynomial reduction, if there is a problem $\pi \in$ **NP-c** $\cap$ **P**, then **P** $=$ **NP**. Otherwise, if there is a problem $\pi \in$ **NP** $\setminus$ **P**, then **P** $\neq$ **NP**. It is not known any problem with any of those properties.

4.4. Algorithm complexity

We have already obtained the main tools to prove that MORN is a **NP-Hard** problem.

We will introduce some notation before the demonstrations. Let a graph $G = (V, E)$, then let us consider the following definitions [5] :

Definition 4.4.1 *G is* k-edge-connected *if $|V| > k$ and $G \setminus F$ is connected for every set $F \subset E$ of fewer than k edges. The greatest integer k such that G is k-edge-connected is the* edge-connectivity *$\lambda(G)$ of G. In particular, we have $\lambda(G) = 0$ if G is disconnected.*

Definition 4.4.2 *Let $X \subset V \cup E$. We say that X* separates *G if $G \setminus X$ is disconnected. In particular, a vertex which separates two other vertices of the same connected component is a* cutvertex*, and an edge separating its ends is a* bridge*.*

Definition 4.4.3 *If $\{V_1, V_2\}$ is a partition of V, the set $E(V_1, V_2)$ of all the edges of G crossing this partition is called a* cut*. A minimal non-empty cut in G is a* bond*.*

4.4.1. 2-edge-connected topology conditions

Theorem 4.4.4 *Let networks $G_T = (V_T, E_T)$ and $G_D = (V_D, E_D)$ in the conditions of the reduced-MORN problem. Then a 2-edge-connected topology in both networks is a necessary condition for the existence of feasible solutions.*

Proof.
(A) $G_T = (V_T, E_T)$ must be 2-edge-connected.

Firstly, suppose that G_T is not connected. Then there are at least two disjoint connected components H_1 and H_2 included in G_T.
Let $\hat{i}, \hat{j} \in V_D$ such that $tns(\hat{i}) \in H_1$, $tns(\hat{j}) \in H_2$. If $m_{\hat{i}\hat{j}} \neq 0$ there is no way to route traffic from $\hat{i}$ to $\hat{j}$. If $m_{\hat{i}\hat{j}} = 0$ $\forall \hat{i}, \hat{j} \in V_D$ such that $tns(\hat{i}) \in H_1$ and $tns(\hat{j}) \in H_2$, then the problem would be dissociable in equivalent subproblems.

Now, suppose that G_T is a connected graph but not 2-edge-connected. Then, there is a link $\hat{e} \in E_T$ that is a bridge in G_T.
Let H_1 and H_2 the disjoint connected components resulting from taking out the link $\hat{e}$ from G_T, then $G_T = H_1 \uplus H_2$.
Let $\hat{i}, \hat{j} \in V_D / m_{\hat{i}\hat{j}} \neq 0$ and such that $tns(\hat{i}) \in H_1$, $tns(\hat{j}) \in H_2$. (Otherwise, the problem would be dissociable in equivalent subproblems.)

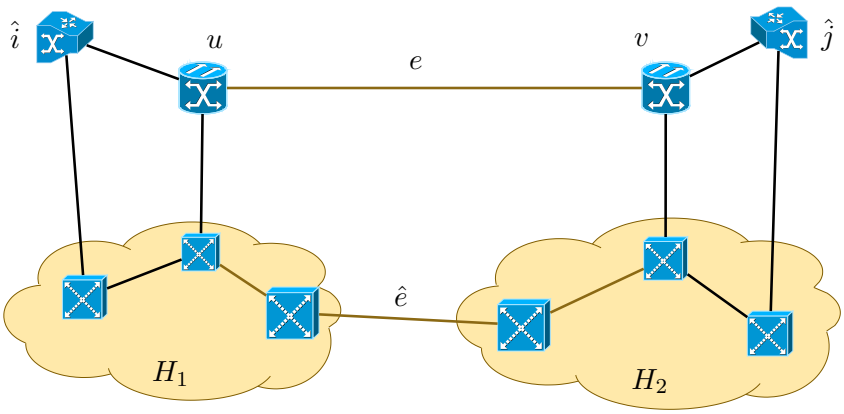


Figure 4.3 – Illustration of elements in proof of Theorem 4.4.4

Let $\rho_{D_{\hat{e}}}^{\hat{i}\hat{j}}$ the tunnel from $\hat{i}$ to $\hat{j}$ in G_D at the failure scenery $\hat{e}$. It is clear that $\exists e = (u, v) \in \rho_{D_{\hat{e}}}^{\hat{i}\hat{j}}$ in which $\hat{e} \in \rho_T^{tns(u)tns(v)}$.
If this was not the case, considering :

$$\hat{p} = \bigcup_{\forall (p,q) \in \rho_{D_{\hat{e}}}^{\hat{i}\hat{j}}} \{(x, y) : (x, y) \in \rho_T^{tns(p)tns(q)}\}$$

would be a path that connects $tns(\hat{i})$ with $tns(\hat{j})$ in $G_T \setminus \{\hat{e}\}$.
This contradicts the fact that $\hat{e}$ was a bridge edge in G_T. Then G_T is 2-edge-connected.

The insight needed for the proof is that every data link $(\hat{i}, \hat{j}) \in E_D$ such that $tns(\hat{i}) \in$

H_1, $tns(\hat{j}) \in H_2$ cannot be routed when the transport bridge edge $\hat{e}$ fails, causing all together the disconnection of G_D.

(B) $G_D = (V_D, E_D)$ must be 2-edge-connected.
Remember that here we assumed $V_D = V_E$ (edge nodes integrated as their own the flow of its dependent access nodes).
Suppose again that G_D is connected (otherwise the problem would be dissociable in equivalent subproblems), but not 2-edge-connected. Then, there is a link $\hat{e}_D \in E_D$ that is a bridge in G_D.
Let Y_1 and Y_2 the disjoint connected components resulting from taking out the link $\hat{e}_d$ from G_D, then $G_D = Y_1 \uplus Y_2$.
Let $\hat{i}, \hat{j} \in V_D / m_{\hat{i}\hat{j}} \neq 0$ and such that $\hat{i} \in Y_1$, $\hat{j} \in Y_2$.
Let $\rho_{D_0}^{\hat{i}\hat{j}}$ the tunnel from $\hat{i}$ to $\hat{j}$ in G_D at the nominal scenery.
Let us note $\hat{e}_D = (u, v)$. Let $\rho_T^{tns(u)tns(v)}$ the routing of $\hat{e}_D$ in G_T, and choose any $\hat{e}_T \in \rho_T^{tns(u)tns(v)}$.
Then, in the failure scenery $\hat{e}_T$, the graph G_D is separated in the components Y_1 and Y_2. So the tunnel $\rho_{D_{\hat{e}_T}}^{\hat{i}\hat{j}}$ between $\hat{i}$ and $\hat{j}$ would not possibly exist at the failure scenery $\hat{e}_T$. Consequently, G_D cannot have bridge edges and is 2-edge-connected.

□

4.4.2. Complexity-related problems

Next we will give some insight of how difficult reduced-MORN is. Firstly, we will show that particular instances of the reduced-MORN problem are **NP-Complete**. We will prove so by reducing those particular cases to the Steiner Two-Edge-Survivable Network Problem, which is a well-known **NP-Complete** problem [2]. Consequently, we claim that the complexity of the reduced-MORN problem is at least **NP-Hard**.

Furthermore, we will consider two kinds of relaxation of the problem releasing some constraints, so making it "easier". We will prove that even with such relaxations, those instances of the problem are equivalent to well-known **NP-Complete** problems. In one case, we will show the equivalence to the Steiner Problem in Graphs, in the other case, the equivalence will be to with the Access Network Design Problem, both **NP-Complete** problems [4, 9, 16, 28].

4.4.3. STESNP relation

Definition 4.4.5 STESNP : Steiner Two-Edge-Survivable Network Problem.
Let $G = (V, E)$ a non-directed simple graph. Given $T \subset V$ a distinguished subset of V. Let $C = \{c_{ij}\}_{(i,j)\in E}$ a matrix of positive real costs associated the edges of G. The STESNP(G,C,T) consists of finding a 2-edge-connected subgraph $H \subset G$ of minimum cost that covers T [2, 3].

Proposition 4.4.6 *STESNP with triangle inequality of costs between edges is a **NP-Complete** problem [2].*

Theorem 4.4.7 *There exist **NP-Complete** instances of the reduced-MORN problem.*

Proof. Let us consider the instance of reduced-MORN with the following parameters :

1. $\bar{M}_0 = \bar{M}_e \ \forall e \in E_T$ (the traffic matrices in nominal and failure sceneries are the same).
2. $\exists$ a set $\hat{V}_D \subset V_D$ such that : $\begin{cases} \dot{m}_{ij} = \ddot{m}_{ij} = 1 & \forall i,j \in \hat{V}_D \\ \dot{m}_{ij} = \ddot{m}_{ij} = 0 & \text{otherwise.} \end{cases}$
3. $\hat{B} = \{b_0, b_1\}$ with $b_0 = 0$ and $b_1 \gg \sum_{\dot{m}_{ij} \neq 0} 2\dot{m}_{ij}$.
4. G_T and G_D are complete graphs such that G_D "copies" the transport network G_T.
5. $T(b_{ij}) = \begin{cases} 1 & \text{if } b_{ij} = b_1 \\ 0 & \text{otherwise.} \end{cases}$
6. The set $C = \{r(e)\}_{e \in E_T}$ satisfies the triangle inequality among the edges of G_T. These are the only costs that intervene in the objective function.

In these conditions, the reduced-MORN problem is equivalent to the problem of finding a 2-edge-connected subgraph $H_T \subset G_T$ of minimal cost that covers the set of nodes $\hat{V}_T$, with $\hat{V}_T = \{tns(v) : v \in \hat{V}_D\}$.

The solution of this instance of reduced-MORN is $S = (H_T, H_D, \tilde{B})$ where :
– H_D is the "copy" of H_T in G_D, and
– $\tilde{B} = \{b_{ij} = b_1 : (i,j) \in H_D\}$.
Due to condition 3 and 5, there are only one capacity available for data links and its cost equals one when used. Because of that, when sizing H_D by using $\tilde{B}$, the cost of the solution is not altered by the technology variable. This means that the cost in the objective function is only affected by the particular transport edges that underlay the chosen data links.

Let $e_T \in H_T$. Then, if e_T fails, only one data link $e_D \in H_D$ is affected, as for condition 4 we are regarding the case in which the Data Network H_D is the copy of H_T. Let $\hat{H}_D = H_D \setminus \{e_D\}$. As condition 3 states that $b_1 \gg \sum_{\dot{m}_{ij} \neq 0} 2\dot{m}_{ij}$, we have that $\forall \hat{i}, \hat{j} \in \hat{V}_D$ such that $\dot{m}_{\hat{i}\hat{j}} \neq 0$, the existence of a tunnel of non-null capacity is guaranteed between $\hat{i}$ and $\hat{j}$ in $\hat{H}_D$. The fact that this capacity is enough to take all the traffic required from $\hat{i}$ to $\hat{j}$ is also assured by condition 3.

This proves the resistance of S to simple failures of the transport network.

Finally, given that the problem STESNP with triangle inequality of costs between edges is a **NP-Complete** problem, this instance of reduced-MORN is **NP-Complete**.

□

4.4.4. MW2ECSN relation

Definition 4.4.8 MW2ECSN : Minimum-Weight 2-Edge-Connected Spanning Network Problem.
Let $G = (V, E)$ be a connected undirected graph. Given a non-negative weight function $C : E \to \mathbb{R}$ associated with triangle inequality of costs between edges, the Minimum-Weight

2-Edge-Connected Spanning Network Problem $MW2ECSN(V, C)$ consists in finding a 2-edge-connected subgraph $H \subset G$ of minimum cost that covers V [22].

Proposition 4.4.9 *MW2ECSN is a **NP-Complete** problem [22].*

Theorem 4.4.10 *There exist a reduced-MORN instance which can be reduced to a $MW2ECSN$ problem.*

Proof. In the conditions of the demonstration of Theorem 4.4.7, but with $\hat{V}_D = V_D$, the problem is reduced to the problem $MW2ECSN(V_D, C)$, which is a **NP-Complete** problem. We conclude that the resulting problem of this instantiation is **NP-Complete**. □

4.4.5. SPG relation

Definition 4.4.11 SPG : Steiner Problem in Graphs.
Let $G = (V, E)$ be a connected undirected graph. Given a non-negative weight function $C : E \to \mathbb{R}$ associated with its edges and a subset $T \subset V$ of terminal nodes, the Steiner Problem in Graphs $SPG(V, E, C, T)$ consists in finding a minimum weighted connected subgraph of G spanning all terminal nodes in T [15].

Proposition 4.4.12 *SPG is a **NP-Complete** problem [9, 16].*

Theorem 4.4.13 *There exist **NP-Complete** instances for relaxations of the reduced-MORN problem.*

Proof.
Let us consider the resulting reduced-MORN problem that comes up by relaxing the constraint of simple failure in G_T links. Let us call this problem RMORNNFR (the reduced-MORN without the failure constraint). Then, let the instance of RMORNNFR with the following parameter values :

1. $\exists$ a set $\hat{V}_D \subset V_D$ such that : $\begin{cases} \dot{m}_{ij} = \ddot{m}_{ij} = 1 & \forall i, j \in \hat{V}_D \\ \dot{m}_{ij} = \ddot{m}_{ij} = 0 & \text{otherwise.} \end{cases}$
2. $\hat{B} = \{b_0, b_1\}$ with $b_0 = 0$ and $b_1 \gg \sum_{\dot{m}_{ij} \neq 0} 2\dot{m}_{ij}$.
3. G_T and G_D are complete graphs such that G_D "copies" the transport network G_T.
4. $T(b_{ij}) = \begin{cases} 1 & \text{if } b_{ij} = b_1 \\ 0 & \text{otherwise.} \end{cases}$
5. The cost matrix $C = \{r(i, j)\}_{(i,j) \in E_T}$ is real and positive. These are the only costs that intervene in the objective function of RMORNNFR.

This instantiation of RMORNNFR is equivalent to solving the problem $SPG(V_T, E_T, \hat{V}_T, C)$ where $V_T = \{tns(v) : v \in \hat{V}_D\}$. The elimination of the constraint of simple failure survival in G_T makes the 2-edge-connectivity condition unnecessary in order to be a feasible solution to the RMORNNFR problem.

Finally, as the SPG is a **NP-Complete** problem, then this instantiation of RMORNNFR is **NP-Complete**.

□

4.4.6. ANDP relation

The Access Network Design Problem is usually defined in the context of a wide area network (WAN), which can be seen as a set of sites and a set of communication lines that interconnect the sites. A typical WAN is organized as a hierarchical structure integrating two levels : the backbone network and the access network composed of a certain number of local access networks.

We introduce the notation used to formalise the problem :
- S_T is the set of terminal sites or clients (nodes),
- S_C is the set of sites where concentrator equipment can be installed to diminish the cost of the Network (Steiner nodes),
- z is a node that represents the backbone,
- $S = S_T \cup S_C \cup \{z\}$ is the set of all nodes.
- $C = \{c_{ij} : i, j \in S\}$ is the matrix which gives for any pair of sites of S, the cost of laying a line between them. When the direct connection between i and j is not possible, we set $c_{ij} = \infty$.
- $E = \{(i, j); \forall i, j \in S : c_{ij} < \infty\}$ is the set of feasible connections between sites of S.
- $G_A(S, E)$ is the graph of feasible connections on the Access Network.

Definition 4.4.14 ANDP : Access Network Design Problem.
We define the Access Network Design Problem $ANDP(G_A(S, E), C)$ as the problem of finding a subgraph $\mathcal{T} \subset G_A$ of minimum cost such that $\forall s_t \in S_T$ there exists a unique path from s_t to node z and such that terminal sites can not be used as intermediate nodes (they must thus have degree 1 in the solution). We will denote by Γ_{ANDP} the space of feasible solutions associated with the problem [4].

Proposition 4.4.15 *ANDP is a **NP-Complete** problem [4, 28].*

Theorem 4.4.16 *There exist **NP-Complete** instances for relaxations of the MORN problem.*

Proof. Let us consider the original MORN problem. We relieve the constraint of resistance of the Data Network to simple failures of the Transport Network.

Under this conditions, we consider the following parametrization :

1. $\bar{M}_0 = ((\vec{m}_{ij}))_{i,j \in V_A}$, where $\vec{m}_{ij} = (\dot{m}_{ij}, \ddot{m}_{ij})$, is the only point to point traffic matrix between the access nodes. There are no matrices $\bar{M}_e \; \forall e \in E_T$ because of the assumption that relaxes the problem.
2. $\exists$ a node $\hat{z} \in V_E$ such that : $\begin{cases} \dot{m}_{i\hat{z}} = \ddot{m}_{i\hat{z}} = 1 & \forall i \in V_A \\ \dot{m}_{ij} = \ddot{m}_{ij} = 0 & \text{otherwise.} \end{cases}$

3. $\hat{B} = \{b_0, b_1\}$ with $b_0 = 0$ and $b_1 \gg \sum_{\dot{m}_{ij} \neq 0} 2\dot{m}_{ij}$.
4. G_T and G_D are complete graphs such that G_D "copies" the transport network G_T.
5. $T(b_{ij}) = \begin{cases} 1 & \text{if } b_{ij} = b_1 \\ 0 & \text{otherwise.} \end{cases}$
6. The cost matrix $C = \{r(i,j)\}_{(i,j) \in E_T}$ is real and positive. These are the only costs that intervene in the objective function of MORN.

The formulation of original MORN, with the relaxation that does not require robustness to simple failures and with this parametrization, makes the resulting problem equivalent to solving $ANDP(\hat{H}, C)$ where :

- $\hat{H} = (\hat{V}, E_T)$ and $\hat{V}$ is discomposed in :
  - $V = S_T \cup S_C \cup \{z\}$ is the set of all nodes,
  - $S_T = \{tns(v) : v \in V_A\}$,
  - $S_C = \{tns(v) : v \in V_E \setminus \{\hat{z}\}\}$,
  - $z = tns(\hat{z})$.

The S_T nodes are terminals in $\hat{H}$. The S_C nodes are the Steiner nodes in $\hat{H}$. The node z is the one to which all nodes in S_T must connect at minimal cost in $\hat{H}$.

Let $\hat{S}_T \subset \hat{H}$ an optimal solution to $ANDP(\hat{H}, C)$. A global optimal solution to the relaxed MORN that satisfies conditions 1-6, is given by $\hat{S} = (\hat{H}_T, \hat{H}_D, \hat{B})$ where :

- $\hat{H}_D$ is the "copy" of $\hat{H}_T$ in G_D, and
- $\hat{B} = \{b_{ij} = b_1 : (i,j) \in \hat{H}_D\}$.

Finally, given the generality of $ANDP(\hat{H}, C)$ in this context, and as the ANDP is a **NP-Complete** problem, then this instantiation for relaxations of MORN is **NP-Complete**.

□

Chapter 5

Non-linear binary integer programming model

5.1. Introduction

In this chapter, it is presented the binary integer programming model designed by the project team. This is an interesting approach due to the common usage of computational solvers for this kind of models. We remark that this model is non-linear and we will discuss its implications later.

It is also observed that an abuse of notation is undertaken in this chapter, noting nodes v_i as i, due to the amount of sub-indexes and super-indexes used for integer variables.

5.2. Problem variables

Despite the beginning of last chapter was flooded by definitions of elements, sets, functions and so on, a several more are required for this chapter. Specially binary integer variables, which have not been considered up to now.

- $x_{ij}^{kl} = \begin{cases} 1, & \text{If the link } (i,j) \text{ of the Data Network is} \\ & \text{used to send traffic from } k \text{ to } l\,;\, k,l \in V_D\,; \\ 0, & \text{Otherwise.} \end{cases}$
- $y_{ij}^{kl} = \begin{cases} 1, & \text{If the link } (i,j) \text{ of the Transport Network is} \\ & \text{used to send data flow from } k \text{ to } l\,;\, k,l \in V_T\,; \\ 0, & \text{Otherwise.} \end{cases}$
- $y_{ij} = \begin{cases} 1, & \text{If } i \in V_A \text{ is connected to node } j \in V_E\,; \\ 0, & \text{Otherwise.} \end{cases}$
- $w_{ij}^{t} = \begin{cases} 1, & \text{If } (i,j) \in E_D \text{ uses capacity } \hat{b}_t \in \hat{B}\,; \\ 0, & \text{Otherwise.} \end{cases}$

These are the binary integer problem variables. This is to say that in order to solve the problem, one has to return the values of each variable for all the possible indexes. It is mandatory

to mention that given a solution expressed in terms of these binary integer variables, it is completely determined the structure of the solution data and transport network, altogether with its associated capacities. Conversely, given a solution network, it is easy to calculate the values of the variables. It is clear (and logical) then that there is bijective function between them.

Next, even at the risk of arising the reader's hatred, a few more definitions of order are introduced.

Definition 5.2.1 *Let* $\chi_{ij} = \begin{cases} 1, & \textit{If there is a data link between the nodes } (i,j) \in E_D\textit{;} \\ 0, & \textit{Otherwise.} \end{cases}$

χ_{ij} acts as an indicator function of presence of a link between two data nodes.

Definition 5.2.2 *Let* $f_{G_D} : E_T \to E_D^*$*, such that for* $l \in E_T$*,* $f_{G_D}(l)$ *returns the set of the data network links which are impacted by the failure of link* l *in the transport network.*

A similar way of expressing these sets using the variables already presented is :

Definition 5.2.3 *Let* $l = (p,q) \in E_T$*, then the set* U_l *is defined as :*

$$U_l = f_{G_D}((p,q)) = \{(i,j) \in E_D | y_{pq}^{tns(i)tns(j)} = 1\}.$$

Another equivalent definition for the flow implementation is as follows :

Definition 5.2.4 *Let* $i,j \in V_D$*,* $\rho_T^{i,j} = \{(u,v) \in E_T | y_{uv}^{tns(i)tns(j)} = 1\}$.

Definition 5.2.5 *Let* $i \in V_A$ *an access node, the predefined set of edge nodes to which it is able to directly connect is* $W_i \subseteq V_D$.

5.3. Complete binary integer model

Here we introduce the mathematical binary integer model. Finding a solution to the model means that given an input set of parameters : the transport and data network graphs, node distances, traffic matrix, z-function, set of bandwidths and its cost, one has to obtain the values

for the variables, which is the same as getting the solution network.

$$\min \sum_{i,j \in V_D} \left(\sum_{(u,v) \in E_T} y_{uv}^{tns(i)tns(j)} \cdot r_T(u,v) \right) \cdot T\Big(\sum_{\hat{b}_t \in \hat{B}} w_{ij}^t \cdot \hat{b}_t \Big) \tag{5.1a}$$

$$\text{s.t.}: \ \chi_{ij} = \sum_{\hat{b}_t \in \hat{B}} w_{ij}^t \leq 1 \qquad \forall (i,j) \in E_D. \tag{5.1b}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(tns(\hat{i}),k) \in E_T} y_{tns(\hat{i}),k}^{tns(\hat{i})tns(j)} - y_{i\hat{i}} \cdot (1 - ns(i,j)) \right) \geq 0 \qquad \begin{array}{l} \forall i \in V_A, \forall j \in V_E, \\ \forall \hat{i} \in W_i. \end{array} \tag{5.1c}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(k,tns(j)) \in E_T} y_{k,tns(j)}^{tns(\hat{i})tns(j)} - y_{i\hat{i}} \cdot (1 - ns(i,j)) \right) \geq 0 \qquad \begin{array}{l} \forall i \in V_A, \forall j \in V_E, \\ \forall \hat{i} \in W_i. \end{array} \tag{5.1d}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(tns(\hat{i}),k) \in E_T} y_{tns(\hat{i}),k}^{tns(\hat{i})tns(\hat{j})} - (2 \cdot (y_{i\hat{i}} + y_{j\hat{j}}) - 3) \cdot (1 - ns(i,j)) \right) \geq 0 \qquad \begin{array}{l} \forall i,j \in V_A, \\ \forall \hat{i} \in W_i, \forall \hat{j} \in W_j. \end{array} \tag{5.1e}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(k,tns(\hat{j})) \in E_T} y_{k,tns(\hat{j})}^{tns(\hat{i})tns(\hat{j})} - (2 \cdot (y_{i\hat{i}} + y_{j\hat{j}}) - 3) \cdot (1 - ns(i,j)) \right) \geq 0 \qquad \begin{array}{l} \forall i,j \in V_A, \\ \forall \hat{i} \in W_i, \forall \hat{j} \in W_j. \end{array} \tag{5.1f}$$

$$\dot{m}_{ij} \cdot ns(i,j) \cdot \left(\sum_{(p,k) \in E_T} y_{pk}^{tns(\hat{i})tns(j)} - \sum_{(k,p) \in E_T} y_{kp}^{tns(\hat{i})tns(j)} \right) = 0 \qquad \begin{array}{l} \forall i \in V_A, \forall j \in V_E, \\ \forall \hat{i} \in W_i, \\ \forall p \in V_T \backslash \{tns(\hat{i}), tns(j)\}. \end{array} \tag{5.1g}$$

$$\dot{m}_{ij} \cdot ns(i,j) \cdot \left(\sum_{(p,k) \in E_T} y_{pk}^{tns(\hat{i})tns(\hat{j})} - \sum_{(k,p) \in E_T} y_{kp}^{tns(\hat{i})tns(\hat{j})} \right) = 0 \qquad \begin{array}{l} \forall i,j \in V_A, \\ \forall \hat{i} \in W_i, \forall \hat{j} \in W_j, \\ \forall p \in V_T \backslash \{tns(\hat{i}), tns(\hat{j})\}. \end{array} \tag{5.1h}$$

$$\sum_{(tns(i),k) \in E_T} y_{tns(i),k}^{tns(i)tns(j)} \geq \chi_{ij} \qquad \forall i,j \in V_E. \tag{5.1i}$$

$$\sum_{(k,tns(j)) \in E_T} y_{k,tns(j)}^{tns(i)tns(j)} \geq \chi_{ij} \qquad \forall i,j \in V_E. \tag{5.1j}$$

$$\sum_{(p,k) \in E_T} y_{pk}^{tns(i)tns(j)} - \sum_{(k,p) \in E_T} y_{kp}^{tns(i)tns(j)} \geq 1 - \chi_{ij} \qquad \begin{array}{l} \forall i,j \in V_E, \\ \forall p \in V_T \backslash \{tns(i), tns(j)\}. \end{array} \tag{5.1k}$$

$$\sum_{(p,k) \in E_T} y_{pk}^{tns(i)tns(j)} - \sum_{(k,p) \in E_T} y_{kp}^{tns(i)tns(j)} \leq \chi_{ij} - 1 \qquad \begin{array}{l} \forall i,j \in V_E, \\ \forall p \in V_T \backslash \{tns(i), tns(j)\}. \end{array} \tag{5.1l}$$

$$\sum_{u,v \in V_D} x_{ij}^{uv} \cdot \dot{m}_{uv} \leq \sum_{\hat{b}_t \in \hat{B}} w_{ij}^{t} \cdot \hat{b}_t \qquad \begin{matrix} \forall l \in E_T, \\ \forall (i,j) \in E_D \setminus U_l. \end{matrix} \tag{5.1m}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(i,k) \in E_D \setminus U_l} x_{ik}^{ij} - 1 \right) \geq 0 \qquad \begin{matrix} \forall l \in E_T, \\ \forall i,j \in V_D. \end{matrix} \tag{5.1n}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(k,j) \in E_D \setminus U_l} x_{kj}^{ij} - 1 \right) \geq 0 \qquad \begin{matrix} \forall l \in E_T, \\ \forall i,j \in V_D. \end{matrix} \tag{5.1ñ}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(p,k) \in E_D \setminus U_l} x_{pk}^{ij} - \sum_{(k,p) \in E_D \setminus U_l} x_{kp}^{ij} \right) = 0 \qquad \begin{matrix} \forall l \in E_T, \\ \forall i,j \in V_D, \\ \forall p \in V_D \setminus \{i,j\}. \end{matrix} \tag{5.1o}$$

$$\sum_{k,l \in V_D} x_{ij}^{kl} \cdot \dot{m}_{kl} + z(\sum_{k,l \in V_D} x_{ij}^{kl} \cdot \ddot{m}_{kl}) \leq \sum_{\hat{b}_t \in \hat{B}} w_{ij}^{t} \cdot \hat{b}_t \qquad \forall (i,j) \in E_D. \tag{5.1p}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(i,k) \in E_D} x_{ik}^{ij} - 1 \right) \geq 0 \qquad \forall i,j \in V_D. \tag{5.1q}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(k,j) \in E_D} x_{kj}^{ij} - 1 \right) \geq 0 \qquad \forall i,j \in V_D. \tag{5.1r}$$

$$\dot{m}_{ij} \cdot \left(\sum_{(p,k) \in E_D} x_{pk}^{ij} - \sum_{(k,p) \in E_D} x_{kp}^{ij} \right) = 0 \qquad \begin{matrix} \forall i,j \in V_D, \\ \forall p \in V_D \setminus \{i,j\}. \end{matrix} \tag{5.1s}$$

$$\sum_{j \in W_i} y_{ij} = 1 \qquad \forall i \in V_A. \tag{5.1t}$$

$$\dot{m}_{ij} \cdot \left(x_{i\hat{i}}^{ij} - y_{i\hat{i}} \right) = 0 \qquad \begin{matrix} \forall i \in V_A, \forall j \in V_D, \\ \forall \hat{i} \in W_i. \end{matrix} \tag{5.1u}$$

$$\dot{m}_{ij} \cdot \left(x_{\hat{j}j}^{ij} - y_{j\hat{j}} \right) = 0 \qquad \begin{matrix} \forall i \in V_D, \forall j \in V_A, \\ \forall \hat{j} \in W_j. \end{matrix} \tag{5.1v}$$

$$w_{ij}^{t} \in \{0,1\}, x_{ij}^{kl} \in \{0,1\}, y_{ij}^{kl} \in \{0,1\}, y_{ij} \in \{0,1\}. \tag{5.1w}$$

5.4. Constraints

The model by itself might be found rather unamiable. In order to turn it a little more pleasant, a proper explanation for the constraints is described next.

5.1a : The first term is again the objective function to optimise. The aim is to minimise the cost of the total network. So that, the combination of all data nodes is considered to calculate the cost. If there is a data link between two nodes, then its bandwidth is not null and it has an associated cost per length. For that data link, only the transport links which are used to send data flow are taken into account in order to measure their separation.

It is also possible to express the objective function as in MORN model (4.1.2) in the way

$$\sum_{i,j \in V_D} r_T(\rho_T^{ij}) T(b_{ij})$$

because ultimately the integer variable present in the objective function indicates the transport path of a data link.

The remaining equations are nothing but constraints.

5.1b : The first constraint means that for every pair of data nodes, there is only one capacity available for the link between them, or otherwise there is no link between them, in which case the sum is zero. In this way, χ_{ij} acts as an indicator function of presence of a link between two data nodes.

5.1c, 5.1d : $\forall i \in V_A$, $\forall j \in V_E$ such that $ns(i,j) = 0$ and $\dot{m}_{ij} > 0$, there must exist in G_T a connecting path from $tns(\hat{i})$ to $tns(j)$, where $\hat{i} \in V_E$ is the edge node to which the access node i is connected.

5.1e, 5.1f : Similarly, $\forall i, j \in V_A$ such that $ns(i,j) = 0$ and $\dot{m}_{ij} > 0$, there must exist in G_T a connecting path from $tns(\hat{i})$ to $tns(\hat{j})$, where $\hat{i} \in V_E$ and $\hat{j} \in V_E$ are the edge nodes to which the access nodes i and j are respectively connected.

5.1g, 5.1h : This constraint are balance equations, meaning that all the traffic travelling in a path from $tns(\hat{i})$ to $tns(j)$, which enters each non-terminal transport node p must leave it.

5.1i, 5.1j : $\forall i, j \in V_E$ such that $b_{ij} > 0$ there must exist in G_T at least one connecting path between $tns(i)$ and $tns(j)$.

5.1m, 5.1n, 5.1ñ, 5.1o : Failure sceneries. For every single failure in the Transport Network, the Data Network ought to be designed such that the entire exchange of traffic could be carried out using only the remaining links. So $\forall l \in E_T$, let $f_{G_D}(l) = U_l \subseteq E_D$ the Data Network links set that are affected by the failure of the Transport Network link l. Then, the graph $G_l^* = (V_D, E_D \setminus U_l)$ must fulfil the following :

a) The Data Network sizing must support the projection of the matrix $\{\dot{m}_{uv}\}_{u,v \in V_D}$ over the residuary graph G_l^* at the $l-th$ failure scenery in the Transport Network :

$$\sum_{k,l \in V_D} x_{ij}^{kl} \cdot \dot{m}_{kl} \leq b_{ij}, \quad \forall (i,j) \in E_D \setminus U_l.$$

b) Furthermore, $\forall i,j \in V_D$ such that $\dot{m}_{ij} > 0$, in G_l^* there must exist at least one connecting path between i and j :

$$\sum_{(i,k)\in E_D\setminus U_l} x_{ik}^{ij} \geq 1,$$
$$\sum_{(k,j)\in E_D\setminus U_l} x_{kj}^{ij} \geq 1,$$
$$\sum_{(p,k)\in E_D\setminus U_l} x_{pk}^{ij} - \sum_{(k,p)\in E_D\setminus U_l} x_{kp}^{ij} = 0, \quad \forall p \in (V_D \setminus \{i,j\}).$$

5.1p : The capacity of a Data Network link must be able to support the whole committed traffic carried by it, plus a certain percentage of the peak traffic :

$$\sum_{k,l\in V_D} x_{ij}^{kl} \cdot \dot{m}_{kl} + z(\sum_{k,l\in V_D} x_{ij}^{kl} \cdot \ddot{m}_{kl}) \leq b_{ij}, \forall (i,j) \in E_D.$$

5.1q, 5.1r, 5.1s : $\forall i,j \in V_D$ such that $\dot{m}_{ij} \neq \vec{0}$, in G_D (flawless scenery) there must exist at least one connecting path between i and j :

$$\sum_{k,l\in V_D} x_{ij}^{kl} \cdot \dot{m}_{kl} \leq b_{ij}, \qquad \forall (i,j) \in E_D,$$
$$\sum_{(i,k)\in E_D} x_{ik}^{ij} \geq 1,$$
$$\sum_{(k,j)\in E_D} x_{kj}^{ij} \geq 1,$$
$$\sum_{(p,k)\in E_D} x_{pk}^{ij} - \sum_{(k,p)\in E_D} x_{kp}^{ij} = 0, \quad \forall p \in (V_D \setminus \{i,j\}).$$

5.1t : $\forall i \in V_A$, the access node i must be effectively connected to one and only one Data Network edge node. It is known the set of available edge nodes $W_i \subseteq V_E$ to which the access node i is due to be connected.

5.1u, 5.1v : This constraints imply that the traffic sent and received by an access data node to any other node of the network is only permitted to reach the former through the unique edge node to which it is linked.

5.1w : Clearly, this formal constraint remarks the binary integer character of the variables which act as indicator functions.

A few relevant comments before ending the chapter. Firstly, we emphasize that there is no unique mathematical model that includes all the information and constraints of our problem. Many variations can be taken into account to obtain a better time or memory performance. Even though, the one described in this chapter accomplishes the goal of modelling the complete problem using binary integer variables.

Secondly, notice that the title of the chapter actually states that the model is non-linear. In accordance, the objective function involves the product of the variables, turning the model non-linear. Consequently, several modifications have presently been undertaken in order to utilize any optimization software package like CPLEX as solver.

Besides, despite the resolution of this model should yield the optimal solution, the size of the problem —that in this case refers to the amount of nodes and links involved—, turns it impossible to solve in a lifetime. As a result, a number of relaxations of the constraints and improvement on this model are being carried out by the project team as a parallel line of work.

Part III

METAHEURISTICS

Chapter 6

VNS description

6.1. Introduction

In chapter 4 it has been shown the high complexity of the problem. In chapter 5 we discussed that the integer binary programming model is non-linear and untreatable unless several relaxations are admitted. It becomes clear hence the motivation to undertake VNS perspective in sight of the success of metaheuristics when dealing with **NP-Hard** combinatorial optimisation problems.

In this chapter, the generalities of the selected metaheuristic for the approach of the optimisation problem are introduced. We start by presenting metaheuristics and after that we focus on VNS, the adopted metaheuristic, reviewing its simplicity and effectiveness.

6.2. Metaheuristics

Metaheuristics are general high-level procedures that coordinate simple heuristics and rules to find good (often optimal) approximate solutions to computationally difficult combinatorial optimisation problems [26], usually based upon a basic idea, or analogy [11].

The first works based in stochastic optimisation methods appeared in 1952. Over the years, the main ideas have been developing into the presently well-known metaheuristics. Among them, we find genetic algorithms (1957) [13, 29], simulated annealing (1983) [18], tabu search (1986) [10], ant colonies (1991) [6, 7], GRASP (1995) [4, 8, 25, 26, 28], VNS (1995) [11, 12, 14, 20, 21, 23], and others. They are based on distinct paradigms and offer different mechanisms to escape from locally optimal solutions.

6.3. VNS generalities

Variable Neighbourhood Search (VNS) is a recent [20, 21] and effective metaheuristic for solving combinatorial and global optimisation problems. It is capable of escaping from the local

optima by systematic changes of the neighbourhood structures within the search [11, 12, 23].

6.3.1. Combinatorial optimisation problems

An optimisation problem consists in finding the minimum or maximum of a function $f(x)$ called the *objective function*. We denote $x = (x_1, x_2, \ldots, x_n)$ the *problem variables* and they stand for an alternative solution to the problem living in a set X, the *solution space*. Generally, the problem variables must obey certain rules expressed as a convention $g(x) \leq 0$ named *constraints* of the problem, which restrict the possible solutions to $\Omega = \{x \in X : g(x) \leq 0\}$, the *feasible solution space*. An optimal solution x^* (or global minimum) of the problem is a feasible solution where the minimum of

$$\begin{cases} \min\limits_{x \in X} f(x) \\ \text{s.t.} \quad x \in \Omega \end{cases} \tag{6.1}$$

is reached. That is, $x^* \in X$ has the property that $f(x^*) \leq f(x), \forall x \in \Omega$. An analogue definition is used for a maximization problem.

For some problems, the consideration of a solution as an ordered $n-tuple$ is too restrictive and the following generalization is made. A solution x is defined as a subset of a ground set $E = \{e_1, \ldots, e_n\}$ and the solution space becomes $X = 2^E$.

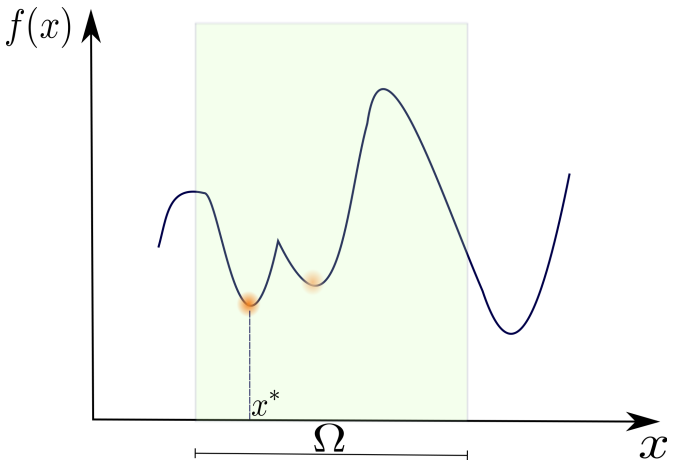


Figure 6.1 – Objective function

Image 6.1 shows a local minimum (right) and a global minimum (left). Notice that the pointed x^* is a minimum of f constrained to the set Ω, even though there may be values of x outside Ω where the function has lower values.

6.3.2. Neighbourhoods

Definition 6.3.1 *A* neighbourhood structure *in the solution space X is a function $\mathcal{N} : X \to 2^X$ such that it assigns to a given solution x a solution neighbourhood $\mathcal{N}(x) \subset X$.*

It is noteworthy that neighbourhoods may eventually contain feasible as long as infeasible solutions. Each solution $x' \in \mathcal{N}(x)$ is generally reached from x by an operation called a *move*. Usually, if two solutions belong to the same neighbourhood, they either differ by only a few elements, or by a small amount in a suitable defined *distance*. So neighbours $\mathcal{N}_k$ may be induced for example by one or more metrics introduced in the solution space X or by the definition of distinct movements. Then, we define by $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, a finite set of neighbourhood structures en X.

Definition 6.3.2 *A solution x^* is a local minimum with respect to a given neighbourhood $\mathcal{N}_k$ if $f(x^*) \leq f(x), \forall x \in \mathcal{N}_k(x^*)$.*

6.3.3. Local search procedure

The first thing needed to apply VNS to an optimisation problem is to obtain a feasible solution, generally by means of a GRASP (Greedy Randomised Adaptive Search Procedure) procedure. Once such a solution is found, a local search phase looking for improvement in the objective function begins. A local search determines a best solution in the neighbourhood of the current solution.

Pseudocode 1 : "Local Search"

```
1: x ← GenerateInitialSolution() ;
2: Repeat
3:    x ← Improve(x, N(x)) ;
4: until No improvement is possible.
5: Return x.
```

The typical greedy descendent search consist of iteratively replace the current solution by the result of the local search while there exist cost reduction (path method). The movements are made only if there is an improvement in the solution (iterative improvement), see Alg-1.

The scheme of a local search is shown in Fig-6.2. It starts with an initial solution x_1 and the neighbourhood $\mathcal{N}(x_1)$ is explored for a better solution. The exploration may conclude when the first solution is found (first improvement scheme) or the best solution is found (best improvement scheme). In the figure, a new solution x_2 of lesser cost is found. Next, the local search seeks $\mathcal{N}(x_2)$. This is repeated iteratively until no improvement is possible, reaching a final solution x_6. This solution is optimal in the sense that there is no better solution "near" it, with the notion of proximity induced by the neighbourhood structure at issue. The drawback with this approach is that the procedure may get trapped to the local minimum "closest" to the

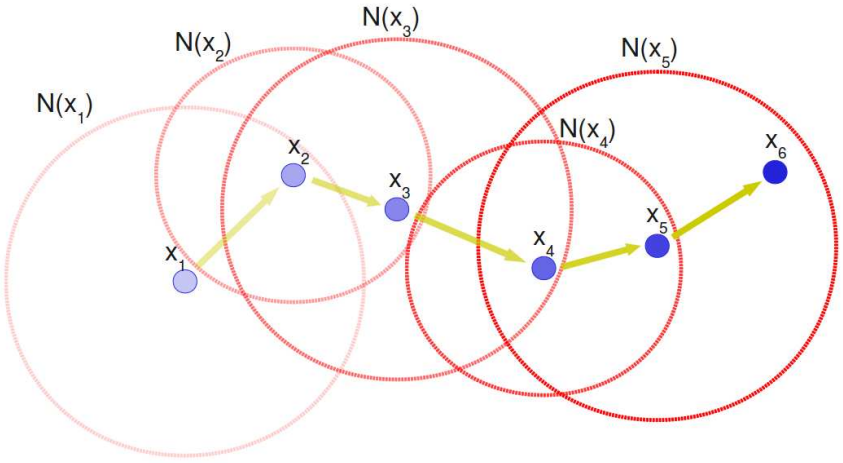


Figure 6.2 – Local search scheme

initial solution as Fig-6.3 exemplify.

The systematic changes of the neighbourhood structures brought in by VNS introduces the possibility of escaping from the local optima.

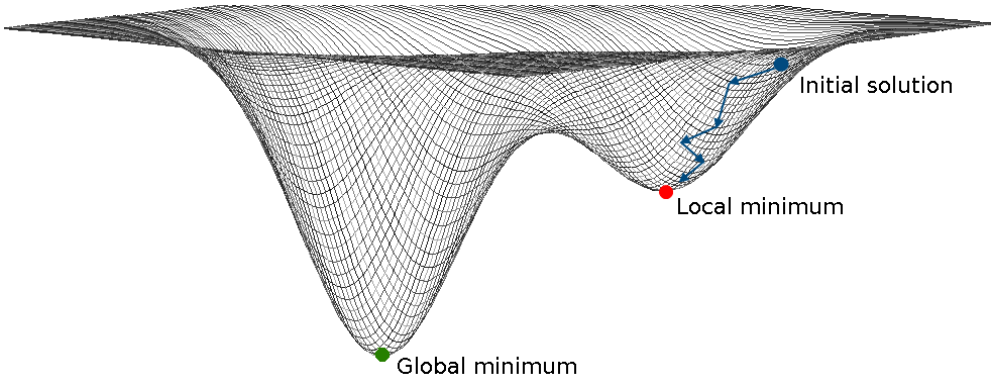


Figure 6.3 – Local search falling in local optimum

VNS is based in three simple facts [12] :

– A local minimum with one neighbourhood structure is not necessarily so with another one.
– A global minimum is a local minimum with respect to all possible neighbourhood structures.
– For many problems, local minima with the same or different neighbourhood structure

are relatively close.

As stated before, most heuristics make use of only one local search, consequently they get stuck at the closest local minimum of the initial solution. In this point is evident the importance of the development of several different neighbourhood structures.

6.4. Fundamental schemes

6.4.1. Descendent VNS

The Variable Neighbourhood Descendent, VND, looks for the best solution in a particular neighbourhood. In case there is no possible improvement in the solution, the algorithm carries on iteratively searching in the following neighbourhood structures. On the contrary, if a new solution with a lesser cost is found, the previous solution is updated with the new one and the procedure starts over taking the first neighbourhood structure. The steps of VND are shown in Pseudo-2.

Pseudocode 2 : "Descendent VNS ; VND"

Initialisation :

- determine a set of neighbourhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, used for the descent ;
- find an initial solution x.

Iterations :

- repeat, until no improvement is obtained, the following sequence :

1: Do $k \leftarrow 1$;
2: **Repeat**
3: (a) *Neighbourhood exploration :* Find the best solution x' in the $k - th$ neighbourhood of x, i.e. $x' \in \mathcal{N}_k(x)$;
4: (b) *Move or not :* If the obtained solution x' is better than x, set $x \leftarrow x'$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$.
5: **until** $k = k_{max}$.
6: **Return** The best solution found x.

The final solution brought about by the algorithm is a minimum to all neighbourhood structures considered. As regards, the probability of a global minimum is higher than using only one structure.

6.4.2. Reduced VNS

The Reduced Variable Neighbourhood Search, RVNS, is obtained by taking random solutions of $\mathcal{N}_k(x)$, without applying a descent to them. The steps of RVNS are given in Pseudo-3.

RVNS is useful for instances of big problems where each local search is very expensive. The optimal value of the parameter k_{max} is frequently 2.

Pseudocode 3 : "Reduced VNS ; RVNS"

Initialisation :

- determine a set of neighbourhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, used for the search ;
- find an initial solution x ;
- choose a stopping condition.

Iterations :

- repeat, until stopping condition holds, the following sequence :

1: Do $k \leftarrow 1$;
2: **Repeat**
3: (a) *Shaking :* Randomly choose a solution x' in the $k-th$ neighbourhood of x, i.e. $x' \in \mathcal{N}_k(x)$;
4: (b) *Move or not :* If the obtained solution x' is better than x, set $x \leftarrow x'$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$.
5: **until** $k = k_{max}$.
6: **Return** The best solution found x.

The stopping condition may be the maximum number of iterations between two improvements of the solution. We remark that even though RVNS is not exhaustive in the search, a "good" solution is expected to be found.

6.4.3. Basic VNS

The Basic Variable Neighbourhood Search, BVNS, combines deterministic and randomic changes in the neighbourhood structures. The steps of BVNS are presented in Pseudo-4.

Pseudocode 4 : "Basic VNS ; BVNS"

Initialisation :

- determine a set of neighbourhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, used for the search ;
- find an initial solution x ;
- choose a stopping condition.

Iterations :

- repeat, until stopping condition holds, the following sequence :

1: Do $k \leftarrow 1$;
2: **Repeat**
3: (a) *Shaking :* Randomly choose a solution x' in the $k-th$ neighbourhood of x, i.e. $x' \in \mathcal{N}_k(x)$;
4: (b) *Local search :* Apply some local search method with x' as initial solution : denote x'' the local minimum obtained this way ;
5: (c) *Move or not :* If the obtained solution x'' is better than x, set $x \leftarrow x''$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$.
6: **until** $k = k_{max}$.
7: **Return** The best solution found x.

The step (a) of the procedure, called shaking, is the same than step (a) of RVNS. This random way to generate $x' \in \mathcal{N}_k(x)$ is useful to avoid cycling that may appear by following the deterministic nested sequence of neighbourhood structures.

BVNS stopping condition may be the allowed CPU time, maximum iteration number, or maximum number of iterations between two improvements of the solution.

6.4.4. General VNS

The General Variable Neighbourhood Search, GVNS, changes the step (b), called local search, of BVNS by a VND. The steps of GVNS are given in Pseudo-5.

Pseudocode 5 : "General VNS ; GVNS"

Initialisation :

- determine a set of neighbourhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, used for the agitation ;
- determine a set of neighbourhood structures $\mathcal{N}'_j$, $j = 1, \ldots, j_{max}$, used for the descent ;
- find an initial solution x ;
- choose a stopping condition.

Iterations :

- repeat, until stopping condition holds, the following sequence :

1: Do $k \leftarrow 1$;
2: **Repeat**
3: (a) *Shaking :* Randomly choose a solution x' in the $k-th$ neighbourhood of x, i.e. $x' \in \mathcal{N}_k(x)$;
4: (b) *Local search :* Apply VND with neighbourhood structures $\mathcal{N}'_j$, $j = 1, \ldots, j_{max}$, with x' as initial solution ; denote x'' the local minimum obtained this way ;
5: (c) *Move or not :* If the obtained solution x'' is better than x, set $x \leftarrow x''$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$.
6: **until** $k = k_{max}$.
7: **Return** The best solution found x.

6.4.5. Scheme selected

We dare say that the possible combinations of improvements or variations of VNS is illimited. For example, nesting, randomisation, order of neighbourhood structures, stopping conditions, decompositions, first/best improvement interchange, skewing, paralellisation, and hybrids to mention some. We cite [12] as a basic reference.

As we see, the gamma of variations to study exceeds the aim of this work. In order to being executive, we take an engineering point of view. We select VND scheme to solve our problem in the sake of simplicity.

We have adapted the local searches described in Chapter-8 to VND scheme. There, it is presented a block diagram that illustrates VNS customisation. Finally, in Chapter-10, the obtained results in the cost of the solution and computational time are shown.

6.5. Virtues of VNS as metaheuristic

It is convenient to point out several properties that highlight VNS among the other metaheuristics. These characteristics evince the reasons why a metaheuristic become popular and widely selected for the approach to theoretical and practical problems. Some expected qualities that guarantee the general interest in a metaheuristic are :

1. *Simplicity :* The metaheuristic should be based in a simple and clear principle.
2. *Precision :* The steps of the metaheuristic should be formulated in very precise terms.
3. *Coherence :* All steps of the heuristic algorithms should follow naturally from the metaheuristic's principle.
4. *Effectiveness :* The procedures should return optimal or near-optimal solutions for most practical cases.
5. *Efficiency :* The algorithms should run in a moderate computational time.
6. *Robustness :* The heuristics performance should be consistent over a variety of instances.
7. *User-friendliness :* The heuristics should be clearly expressed and should be easy to understand and easy to use. This means that it should have as few parameters as possible.
8. *Innovation :* The principles of a metaheuristic, or the efficiency or effectiveness of the heuristics derived from it should contribute to new kinds of applications.

It appears that VNS possesses all the qualities listed to a large degree. Despite it exceeds the scope of this work, in [12] or [11] it is briefly analysed the manner VNS fits this properties considering the diversity of problems addressed by it, though we think some of them are evident.

Chapter 7

MORN Heuristic

In this chapter we will explain the principal aspects of the heuristic implemented to solve Reduced-MORN. We have already stated that the approach in this thesis will be VNS. So, the initial feasible solution is acquired by a GRASP algorithm constructed by the project team. Despite we will give an outline of this procedure, we will constantly refer to [27], where a thorough explanation is included, in sight that a deeper description exceeds the scope of this thesis.

In first place, we will generalise the process of optimisation by dividing it into two main blocks as shown in Fig-7.1. The remaining of this chapter we will concentrate on the description of the GRASP construction. In the following chapter we will focus on the VNS scheme employed and carefully review the local searches implemented in order to optimise a feasible solution by reducing its cost as much as possible.

7.1. GRASP construction

The GRASP process takes as input all data related to the problem, and returns an initial feasible solution. Even though the VNS stage does not care whether the initial solution is nearly optimal or not, we point out that the solution returned has a cost more than satisfactory. In fact, for ANTEL test case, any feasible solution is nearly optimal.

The block diagram for the algorithm is presented in Fig-7.2. Now we will comment on each one.

This implementation has an *Initialisation* phase, not included in the diagram for simplicity. Here, invariant data which will be used through the algorithm is precomputed. After that, the *Access-Edge Assignation* between data nodes is carried out. Next, the block $G_E sol$ *routing over* G_T has to do with finding transport paths for edges in the Data Network included for the solution. As the diagram shows, this is performed until the solution is feasible, asking a function *Is_*$G_E sol$*_feasible ?*. Finally, the procedure G_E *reduction* refines the solution returned by GRASP. In the subsequent sections each of these method will be frugally developed.

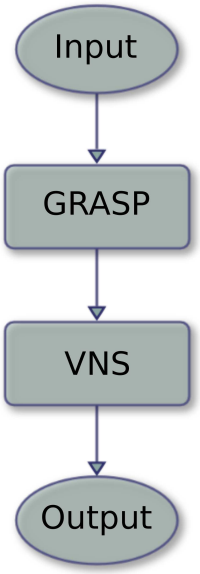


Figure 7.1 – General metaheuristic block diagram

7.1.1. Initialisation

During the initialisation phase, several data is calculated to be used during the rest of the GRASP algorithm. The most important are :

- Distances between nodes.
- Structural bonds calculation.
- Topological association calculation.

Distances between nodes

All the shortest paths between pairs of nodes in G_T are calculated using the Dijkstra algorithm. The corresponding distances for nodes $v_i, v_j \in V_D$ are inferred by the paths of $tns(v_i), tns(v_j) \in V_T$.

Bonds

Let $G = (V, E)$ a plane graph. Then, its *dual graph* $G^* = (V^*, E^*)$ is such that there is one vertex v^* for each face f of G, and for every edge $e \in E$ there exists an edge $e^* \in E^*$ such that the ends of e^* are the vertex in G^* that come from the faces adjacent to e.

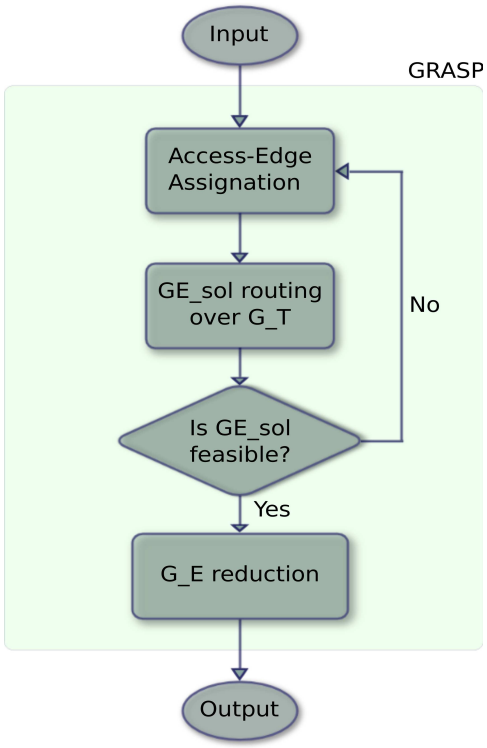


Figure 7.2 – GRASP block diagram

The definition of a dual graph does not depend only on the graph structure, but also on its plane representation. In addition, the dual graph is in general a multigraph. For our purpose, the plane representation to adopt is clearly the actual disposition of the Transport Network itself. Because of this, we require the geographic coordinates of every transport node as input to our algorithm. Besides, in sight that in our problem G_T does not have loops because it is 2-edge-connected, its dual, G_T^*, does not have leaves.

The algorithm that list the bonds in the Transport Network make use of the dual graph and the following property. It is known that edges in a plane graph G form a bond if and only if the corresponding dual edges form a cycle in the dual graph G^* [31].

Consequently, it is sufficient to find all cycles of the dual graph. However, even in a plane cubic (3-regular) graph, this task is **NP-Hard**. Effectively, the problem of finding a Hamiltonian circuit in a plane cubic and connected graph is **NP-Complete** [9], which is a much simpler problem than the previous one. Hence, we cannot expect an efficient algorithm for listing the cuts and then obtain the bonds. Though, in our case, the existence of a few nodes of degree greater than two facilitates the issue.

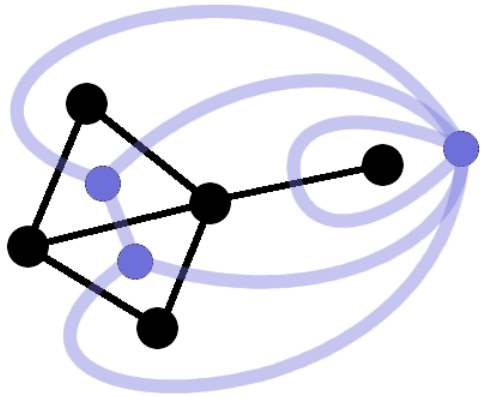


Figure 7.3 – Dual Graph Example

Firstly, all nodes of degree equal two are eliminated and the path they conformed is changed for a simple edge. We obtain a new graph G'_T. Then, we construct the dual graph. In the example, $G'^*_T = (V'^*_T, E'^*_T)$, were $V'^*_T = \{A, B, C, D, E, F\}$ and the edges are drawn in blue.

Its elementary cycles are those that does not contain cycles inside. They are defined by the faces of the dual graph G'^*_T [1]. However, one of them, for example the outer face, must be left aside because it can be obtained from the rest as we will explain. In the example they are ACBA, ACDA, ABEA, CDBC, and ADFA. All the cycles of the graph can be obtained by combining the elementary graphs. Then, we have at most $2^5 = 32$ possible cycles in the dual graph. This number is quite small. We must observe that despite ACBA + ACDA = ABCDA is a valid cycle, ACDA + ABFA = ACDABFA is not a cycle. In fact, ACDABFA conforms an "eight", two cycles that share only a vertex.

Once a cycle is selected, for instance, the cycle ABCDA, the final step consists of returning which is the bond in the primal graph. The way to achieve this is as follows. For every edge of the cycle, we must find the corresponding edge in the primal. For instance, in the corresponding edge for C-D is PAY-ALG. But we must remember that PAY-ALG represents a condensed path between them which in this case it is PAY-PCL-ALG. Because of this, we choose from this path one edge to remove (PAY-PCL or PCL-ALG. Doing this for every edge of the cycle in the dual graph, we obtain a bond in the primal graph. [2] To complete the example, one particular combination of transport edges that constitute one bond for the cycle ABCDA in the dual graph may be : SNR-FRB for AB, YOU-ALG for BC, PAY-PCL for CD, and ART-TQR for DA.

1. Notice that there are three edges (multi-edge) between A-B which may lead to think there are more elementary cycles.
2. Several bonds may be constructed from the same cycle in the dual graph.

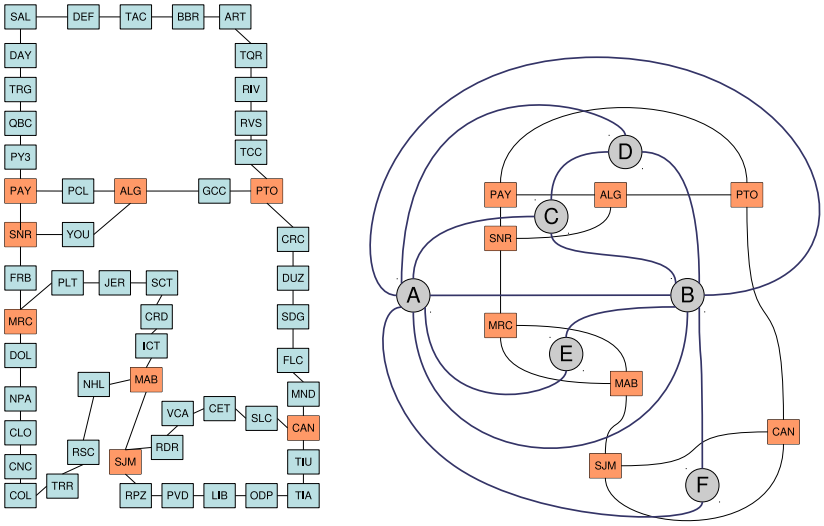


Figure 7.4 – Primal West Transport Network → Dual West Transport Network

The algorithm itself is relatively simple. It may also be implemented efficiently using bitmap vectors for every edge. Even though, as the number of faces increase, the number of cycles grow exponentially. Our implementation takes into account the existence of a vertex of high degree in the dual graph which corresponds to the outer face in order to improve performance.

In sight that bonds are used for the *Is_$G_E sol$_feasible ?* function, the bonds and the path of transport edges in G_T associated to every edge in $G_T'^*$ are precomputed at the initialisation of the GRASP algorithm.

Topological association

In the same way that with the calculation of the dual graph in last section, this procedure builds a new graph G_T'' collapsing the nodes of degree two in G_T with two exceptions : a transport node that share a station with an edge node, or a transport node whose elimination generates a multiedge in the resulting graph.

A new distance function $r' : E_T'' \to \mathbb{R}$, such that aggregates the distances of the collapsed edges.

When in doubt of the creation of multiedges in G_T'', the nodes to be collapsed in G_T should be those resulting in a transport edge in G_T'' with the longer distance associated.

The purpose of this process is to omit from G_T all the stations and nodes that do not change the solution found. This is because once the access had been assigned to their respective edge nodes, they do not provide any information to the solution.

7.1.2. Access-Edge Assignation

The Access-Edge assignation was discussed in Section-4.2. In order to review the particular implementation for the connection we refer to [27]. We do not employ this algorithm directly in sight that the Access-Edge assignation is also precomputed. We work with the demand matrices $\{\bar{M}'_0, \bar{M}'_1, \ldots, \bar{M}'_{|E_T|}\}$ that summarise the traffic that exchange each edge node in every possible failure scenery. Those matrices contemplate the traffic of the access nodes aggregated to their corresponding edge nodes.

7.1.3. $G_E sol$ routing over G_T

The core of this algorithm tries to route the links of G_E through the underlay network in a balanced and economical way.

The balanced approach bears upon avoiding the overload of links over one transport edge. This situation would make impossible the rerouting of traffic in the failure scenario when that particular edge is the one which fails. So that, it is important that all the connections form an edge node toward its neighbours were established in a balanced manner.

In addition, the transport paths in G_T are carefully chosen in order to keep the cost of the solution low. As the Minimum Cost Disjoint Paths problem is **NP-Complete** [9], the algorithm proceeds looking for an almost optimal solution to this problem by means of an heuristic that penalises the repetition of transport edges.

7.1.4. Is $G_E sol$ feasible ?

The *Is_$G_E sol$_feasible ?* function is one of the most important functions of the whole metaheuristic. It is used both during the GRASP construction and also in all local searches. As its name alludes, it evaluates whether a solution is feasible or not.

The algorithm lists the data tunnels ordering them according to a decreasing traffic requirement value. The tunnels are routed one by one through paths with sufficient capacity. Moreover, the procedure tries to make use of routes in G_E with the minimum amount of links as possible. And among routes with equal amount of links, it chooses those with the shortest length in G_T.

In sight that this function is called many times in each iteration, it was important that it possessed a good performance. For this reason, it deals with efficient memory handling, optimised data structures, caches and precomputed values. The actual function validates a solution for a

real network (as large as the network of ANTEL) in a few seconds, and refute a solution in a shorter time. [3]

7.1.5. G_E reduction

This last stage of the Construction Phase consist on building G_E, defining the capacities $B : E_E \rightarrow \hat{B}$ and data tunnels $\Phi : (\emptyset \cup E_T) \rightarrow 2^{P_D}$, with the routing $\Psi : E_E \rightarrow 2^{P_T}$ already determined. On the grounds that doing so is computationally expensive, the approximation adopted to accelerate the construction goes as follows.

In first place, the structural bonds are taken at random one by one. Then, a specific bond, $bond_{G_T} \subset G_T$, is taken at random. Its associated transport path is known. Once obtained the structural bonds, the calculation of all particular bonds, $bond_{G_T}$, is very fast.

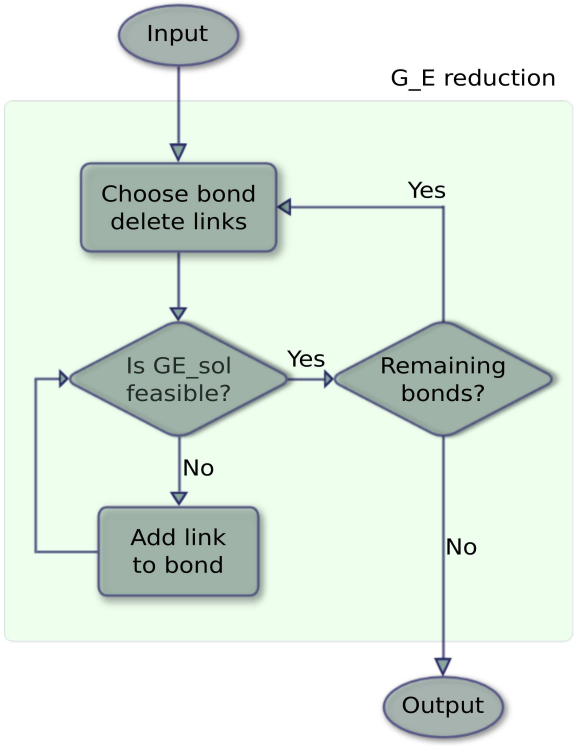


Figure 7.5 – G_E reduction block diagram

Furthermore, it is assumed that $B(e) \in \{0, \hat{b}_{\hat{B}}\}$ builds reasonable good solutions. In other words, if a link is used, it is designed with maximum capacity. The other elements of B are left

3. The boolean value returned by this function might have a margin of error, though small, due to the **NP-Complete** characteristic of this problem.

to be evaluated in local searches. However, despite of simplicity and speed in the algorithms, the spirit is that Data Networks are more economical when there are few links of high capacity than a lot of links of low capacity.

Investigations carried out by the project gave some insight and some mathematical results that the distribution of flows in $bond_{G_T}$ should be as homogeneous as possible for expecting feasible solutions, being a necessary but not sufficient condition. For that purpose, the links that cross the bond are ranked according to their cost in G_T and the most expensive ones are eliminated until an established minimal cut condition is reached.

If the solution is feasible, we keep it and continue with the algorithm analysing another bond. If the solution is not feasible, we start a roll-back process reintegrating the extracted links to the solution until it becomes feasible.

The reason for a massive link elimination against a consecutive removal of links one at a time is efficiency. The computational cost of checking that a solution is feasible is higher than dismissing a solution. As a result, it takes less time to prune all possible links and then return the links to the solution than eliminate one by one while the solution is feasible.

Chapter 8

VNS customisation of the problem

8.1. Introduction

Last chapter treated VNS metaheuristic in an abstract manner. Now is time to show how to adapt it to the given problem yielding the heuristic. Due to the kindness of the method it allow us to develop a relatively simple algorithm adjusted to the characteristics of the problem and reaching highly satisfactory solutions.

8.2. LS : Edge Elimination

The new feasible solution is created either from the initial GRASP construction or from a previous local search. Although its not a really elegant local search, it turns out to be a very efficient one. It is based on removing all data edges that keep the solution feasible.

The idea is this. Let a data edge $e = (u, v) \in \hat{E}_D$. Imagine that both data nodes v and u have degree greater than two. Then, that link could probably be removed due to the fact that e is an arch in a cycle of $\hat{G}_D$. If the Data Network remains two-edge-connected and its is possible to reroute the traffic between all data nodes that used that data link to send traffic in every failure scenario, then, the solution remains feasible and the cost has been reduced.

The algorithm is presented in Alg-6 and explained next. Firstly, each data edge $e \in \hat{E}_D$ is taken out from the initial solution. Or equivalently, a new graph $\hat{H}_D$ is created with all the edges of $\hat{G}_D$ except from e. As it can be seen in step 4 of the algorithm, this local search makes use of the construction procedure for an initial feasible solution but with a reduced amount of links. In particular, with the entire current solution minus the data edge e. The transport paths are created and the conditions for exchange of traffic in every failure scenario are tested. So if the resulting solution happen to be feasible, the graph is updated. Otherwise the next data edge is taken. The search continues until all data edges are analysed.

By the end of the local search we get a minimal Data Network and with a reduced cost.

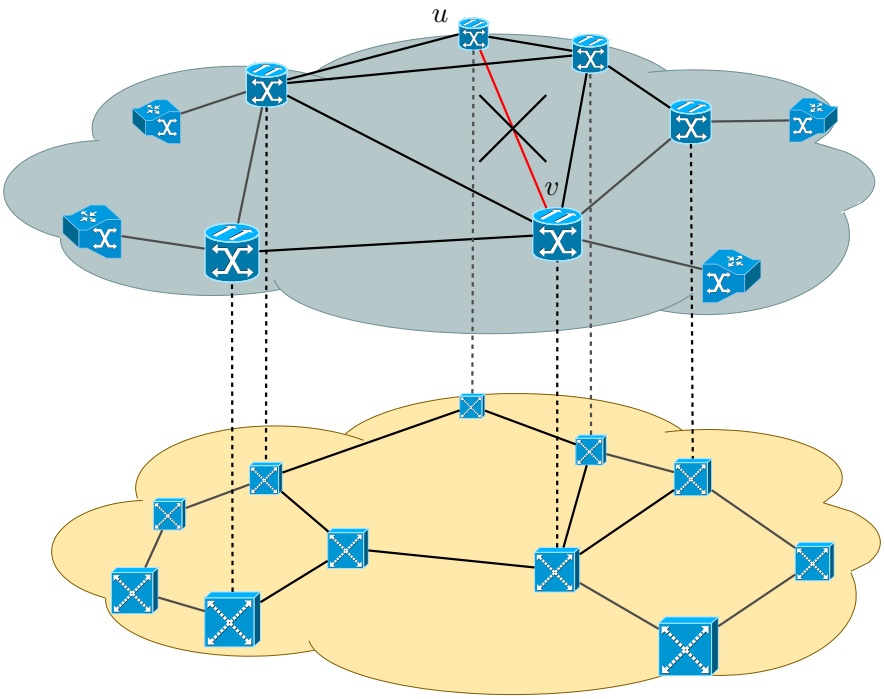


Figure 8.1 – LS : Edge Elimination

Pseudocode 6 : "LS : Edge Elimination"

Input :

- Initial feasible solution $(\hat{G}_D, B, \Phi_D, \Psi_T) = const(G_D, G_T, \hat{B}, \bar{M})$, obtained from GRASP or a previous local search.

Output :

- Final solution $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

Iterations :

1: **For all** $e \in \hat{E}_D$ not analysed **do**
2: Let $e \in \hat{E}_D$ a non-analysed edge ;
3: Let $H_D = \hat{G}_D \setminus \{e\}$;
4: Let $(\hat{H}_D, B, \Phi_D, \Psi_T) = const(H_D, G_T, \hat{B}, \bar{M})$;
5: **If** $cost(\hat{H}_D, B, \Phi_D, \Psi_T) < cost(\hat{G}_D, B, \Phi_D, \Psi_T)$ **then**
6: Do $\hat{G}_D \leftarrow \hat{H}_D$;
7: Goto 1 ;
8: **End if**
9: **End For**
10: **Return** $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

8.3. LS : Path Reduction

This local search tries to reroute data edges through less expensive transport paths by means of a simple movement. The movement, illustrated in Fig-8.2, is to remove a transport node of degree equal two in the solution and restore the path from which it was part of by inserting the transport edge between its previous adjacent nodes. So, the the notion of neighbourhood between two solutions is condensed in the difference in one transport path.

We need to point out that it is necessary that if there is a data node associated to the transport node to extract, such data node does not send or receive traffic. Otherwise, we could not remove the transport node from the solution. Another fundamental, and perhaps obvious condition is the existence of the transport edge linking the neighbours of the transport node to extract.

Pseudocode 7 : "LS : Path Reduction"

Input :

- Initial feasible solution $(\hat{G}_D, B, \Phi_D, \Psi_T) = const(G_D, G_T, \hat{B}, \bar{M})$, obtained from GRASP or a previous local search.

Output :

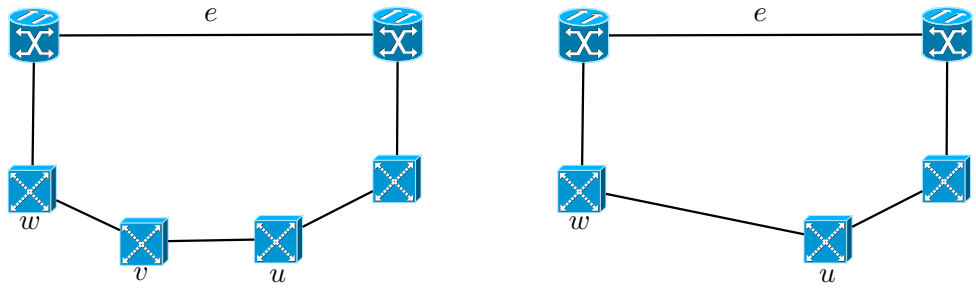
- Final solution $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

Iterations :

1: **For all** $e \in \hat{E}_D$ not analysed **do**
2: Let $e = (v_i, v_j) \in \hat{E}_D$ a non-analysed edge ;
3: Let ρ_T^{ij} the transport path associated to e ;
4: Find $v \in \rho_T^{ij}$ such that $deg(v) = 2$;
5: Find $u, w \in \rho_T^{ij}$ neighbours of v ;
6: **If** $(u, w) \in E_T$ **and** $m_{kl} = \vec{0}\, \forall v_k, v_l \in V_D$ such that $tns(v_k) = v$ **then**
7: Let $\hat{\rho}_T^{ij} = \left(\rho_T^{ij} \setminus \{(v, u); (v, w)\}\right) \cup \{(u, w)\}$;
8: Let $\hat{\Psi}_T = \left(\Psi_T \setminus \rho_T^{ij}\right) \cup \hat{\rho}_T^{ij}$;
9: **End if**
10: **If** $cost(\hat{G}_D, B, \Phi_D, \hat{\Psi}_T) < cost(\hat{G}_D, B, \Phi_D, \Psi_T)$ **then**
11: Do $\Psi_T \leftarrow \hat{\Psi}_T$;
12: **End if**
13: **End For**
14: **Return** $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

The algorithm consists on the following steps, see Alg-7. First, obtain all data edges in the solution. For each data edge, we analyse its corresponding transport path. In the path, we look for a transport node $v \in V_T$ of degree equal two. Let us call $u, w \in V_T$ its neighbours. Then, we make sure it is possible to extract v from the solution. We take the data nodes associated to v, if any, and verify the traffic they exchange with rest of the network is null. In that case and if the the transport edge between u and w exists, then, we remove the node v and its adjacent transport edges (v, u) and (v, w) from the path. We reconstitute the path by adding the trans-

port edge $(u, w) \in E_T$.



(a) Initial solution (b) Improved solution

Figure 8.2 – Path Reduction Scheme

Finally, we must check if the solution obtained this way is both feasible and of an inferior cost.

8.4. LS : Path Reallocation

This local search is aimed to route every data edge in the best possible way through the Transport Network. The neighbourhood structures vary now in the transport paths that the solutions have, i.e. in the way the data edges are routed underlay.

Again we could choose to modify only one path related to one data edge in the local search, or all the paths possible. We have determined to modify every data edge transport path.

Path Reallocation takes into account that the GRASP construction does not necessarily return the best, minimal or inexpensive paths set in the Transport Network, so that optimisations in this aspect ought to be intended.

The pseudocode is found in Alg-8. For every data edge $e \in \hat{G}_D$ we get its transport path associated and find the alternative shortest transport path by using Dijkstra algorithm. We keep the new path if the cost of the solution decreases, otherwise we maintain the previous path. We repeat the procedure for all data edges in $\hat{G}_D$.

One important thing to point out is that this local search properly includes the Path Reduction one. For this reason, it is only sensible to use Path Reallocation after Path Reduction has been run. The motivation for keeping the other local search are valid indeed. One thing is that in sight of its simplicity, Path Reduction might perform a primary optimisation, and then Path Reallocation finish its work when the first one has no effect. On the other side, while it is extremely clear how Path Reduction changes the transport paths, Path Reallocation may return new paths with no correlation to the previous ones.

Pseudocode 8 : "LS : Path Reallocation"

Input :

- Initial feasible solution $(\hat{G}_D, B, \Phi_D, \Psi_T) = const(G_D, G_T, \hat{B}, \bar{M})$, obtained from GRASP or a previous local search.

Output :

- Final solution $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

Iterations :

1: **For all** $e \in \hat{E}_D$ not analysed **do**
2: Let $e = (v_i, v_j) \in \hat{E}_D$ a non-analysed edge ;
3: Let ρ_T^{ij} the transport path associated to e ;
4: Let $\hat{\rho}_T^{ij}$ the Dijkstra shortest transport path from v_i to v_j ;
5: Let $\hat{\Psi}_T = \left(\Psi_T \setminus \rho_T^{ij}\right) \cup \hat{\rho}_T^{ij}$;
6: **If** $cost(\hat{G}_D, B, \Phi_D, \hat{\Psi}_T) < cost(\hat{G}_D, B, \Phi_D, \Psi_T)$ **then**
7: Do $\Psi_T \leftarrow \hat{\Psi}_T$;
8: **End if**
9: **End For**
10: **Return** $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

8.5. LS : Link Decomposition

The basic idea consists of replacing a data edge by a sequence of two data edges, for which the traffic path matches the traffic path of the sequence.

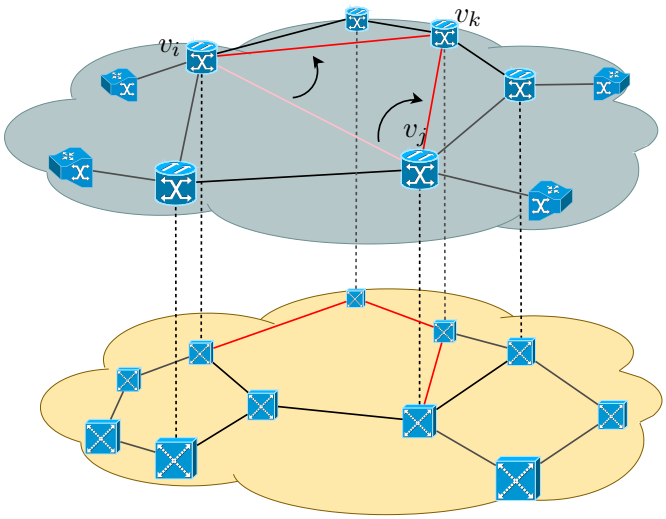


Figure 8.3 – LS : Link Decomposition

In the sake of simplicity, the implementation is a bit different in the way that it makes use of the GRASP construction. This modification arise to avoid the time consuming task of locating matching paths. Instead of doing so, we will extend the idea to replace a data edge by a sequence of two data edges disregarding of its associated path. Therefore, the resulting path may or may not be the conjunction of the previous. Nevertheless, in as much as we care, if the resulting network has an inferior cost, our job is well done.

Actually, if a new graph is found with the same cost, it is also preferable to keep the new solution because as long as we have adjoined two data edges in place of one, a more dense and robust topology is obtained. Besides, we repeat the same argument that a new input for the rest of the local searches is acquired.

Another possible extension, that we prefer not to introduce since it would become too time consuming for a local search, is this. Ultimately, this local search insert two data links and removes one, so we could rename it "2Insertion/Elimination". However, we do neither insert any unrelated two edges not included nor we eliminate any other, but the inserted and removed edges conforms a triangle, a three-edge cycle.

Pseudocode 9 : "LS : Link Decomposition"

Input :
- Initial feasible solution $(\hat{G}_D, B, \Phi_D, \Psi_T) = const(G_D, G_T, \hat{B}, \bar{M})$, obtained from GRASP or a previous local search.

Output :
- Final solution $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

Iterations :

1: **For all** $e \in \hat{E}_D$ not analysed **do**
2: Let $e = (v_i, v_j) \in \hat{E}_D$ a non-analysed edge ;
3: Let $V_i \subset V_D$ the set of adjacent nodes of v_i ;
4: **For all** $v_k \in V_i$ such that v_k is adjacent to node v_j **do**
5: $H_D = \left(\hat{G}_D \setminus \{e\}\right) \cup \{(v_i, v_k); (v_k, v_j)\}$;
6: Let $(\hat{H}_D, B, \Phi_D, \Psi_T) = const(H_D, G_T, \hat{B}, \bar{M})$;
7: **If** $cost(\hat{H}_D, B, \Phi_D, \Psi_T) \leq cost(\hat{G}_D, B, \Phi_D, \Psi_T)$ **then**
8: Do $\hat{G}_D \leftarrow \hat{H}_D$;
9: **End if**
10: **End For**
11: **End For**
12: **Return** $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

The steps of the algorithm are delineated in Alg-9 and ilustrated in Fig-8.3. Firstly, for every data edge $e = (v_i, v_j) \in \hat{E}_D$, we try to find a two-edge chain $v_i - v_k - v_j$ such that (v_i, v_k) and (v_k, v_j) in E_D are not included in $\hat{G}_D$ yet. Then, we remove (v_i, v_j) from $\hat{G}_D$ and add (v_i, v_k) and (v_k, v_j). We turn the set of edges into a feasible solution in step 6. Finally,

if the new solution has a cost less or equal than the current one, we actualise the solution. Otherwise, we try with another chain $v_i - v_l - v_j$, until all nodes in V_i have been chosen. We repeat the procedure for every data edge in $\hat{E}_D$.

We can claim that the movement is valid only (v_i, v_k) and (v_k, v_j) are not in $\hat{G}_D$ only if we have applied the other local searches previously. If both edges were part of $\hat{E}_D$ and when removing (v_i, v_j) the cost would be reduced, then the Elimination LS should have noticed before. If only one edge were part of $\hat{E}_D$, say (v_i, v_k) and when removing (v_i, v_j) the cost would be reduced, then the Insertion/Elimination LS should have noticed before.

8.6. LS : Insertion/Elimination

Even though this local search might seem similar to the previously explained one in Section-8.2, the reached solutions may vary. The idea now is to insert a data edge left outside the current solution, and remove one edge of those used for the solution. The concept now is to avoid to fall into a local minimum. Think that inserting and removing links may lead to non-minimal Data Networks, for which the entire VNS process can be restarted. For this reason, despite the cost of a different newly found solution by this process may be the same, it is as well convenient to take it in.

Pseudocode 10 : "LS : Insertion/Elimination"

Input :

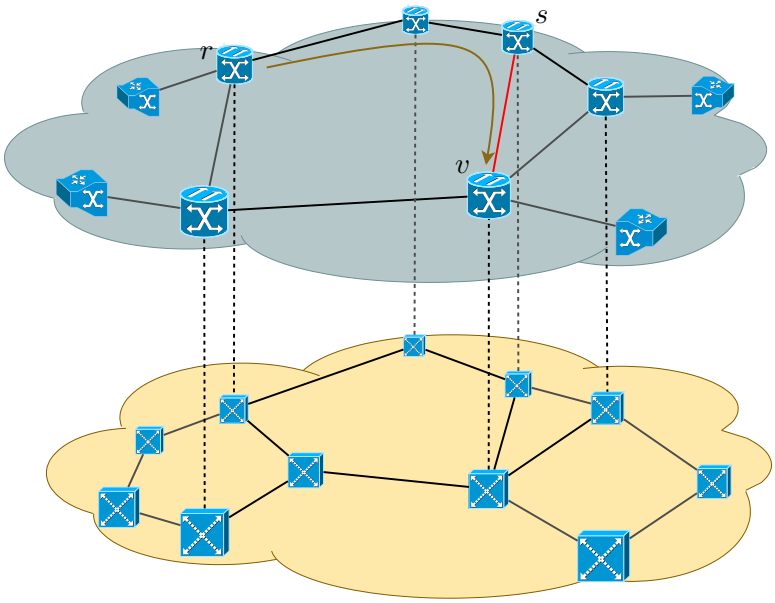
- Initial feasible solution $(\hat{G}_D, B, \Phi_D, \Psi_T) = const(G_D, G_T, \hat{B}, \bar{M})$, obtained from GRASP or a previous local search.

Output :

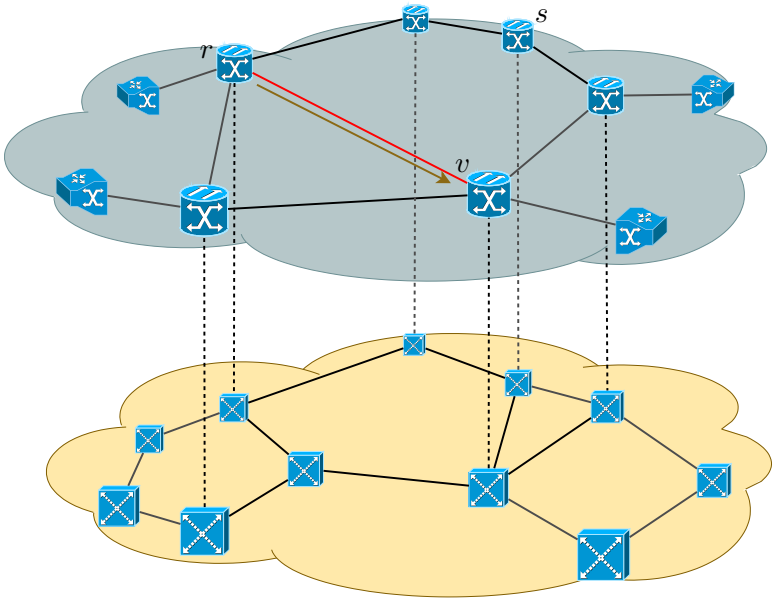
- Final solution $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

Iterations :

1: **For all** $e_o \notin \hat{E}_D$ not analysed **do**
2: Let $e_o \notin \hat{E}_D$ a non-analysed edge ;
3: **For all** $e_i \in \hat{E}_D$ not analysed **do**
4: Let $e_i \in \hat{E}_D$ a non-analysed edge ;
5: Let $H_D = \hat{G}_D \cup \{e_o\} \setminus \{e_i\}$;
6: Let $(\hat{H}_D, B, \Phi_D, \Psi_T) = const(H_D, G_T, \hat{B}, \bar{M})$;
7: **If** $cost(\hat{H}_D, B, \Phi_D, \Psi_T) \leq cost(\hat{G}_D, B, \Phi_D, \Psi_T)$ **then**
8: Do $\hat{G}_D \leftarrow \hat{H}_D$;
9: Goto 1 ;
10: **End if**
11: **End For**
12: **End For**
13: **Return** $(\hat{G}_D, B, \Phi_D, \Psi_T)$.



(a) Initial solution



(b) Improved solution

Figure 8.4 – LS :Insertion/Elimination

The procedure, Alg-10, is very much alike the elimination one. In this occasion, however, we firstly insert one data edge outside $\hat{G}_D$, that we call e_o. And then we apply the same algorithm than in the Elimination Local Search. The difference is that now the feasible solution created must contain the edge e_o. If no data edge can be removed, another edge outside the solution is analysed. The procedure is repeated until all edges are analysed or another solution with cost less or equal to the present solution is encountered.

Some comments are noteworthy. Firstly, notice that this local search also makes use of the construction procedure for an initial feasible solution. Secondly, several other local searches, which are really intuitive, are properly contained by this one. This means that Insertion and Elimination is quite powerful.

8.7. LS : Capacity Reduction

This local search tries to improve the capacities of the data links, remembering that the initial feasible solution returned by the GRASP construction leaves the data links at its maximum capacity, which also has the maximum cost. This means that the cost of that solution may be refined by optimising them. Therefore, it is reasonable to adapt those capacities to a minimum value, bearing in mind that in every failure scenario the capacity of a link should be enough to transfer its according traffic anyway.

Pseudocode 11 : "LS : Capacity Reduction"

Input :
- Initial feasible solution $(\hat{G}_D, B, \Phi_D, \Psi_T) = const(G_D, G_T, \hat{B}, \bar{M})$, obtained from GRASP or a previous local search.

Output :
- Final solution $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

Iterations :

1: **For all** $e \in \hat{E}_D$ not analysed **do**
2: Let $e = (v_i, v_j) \in \hat{E}_D$ a non-analysed edge ;
3: Let $b_{ij} \in B$ the capacity associated to e ;
4: Let t_{ij} the maximum traffic carried by e among all failure scenarios ;
5: Let $\tilde{b}_{ij} = \max_{k=0..\bar{B}}\{\hat{b}_k \in \hat{B} : \hat{b}_k < b_{ij}\}$;
6: Let $\tilde{B} = (B \setminus \{b_{ij}\}) \cup \{\tilde{b}_{ij}\}$;
7: **If** $b_{ij} \geq t_{ij}$ **then**
8: Do $B \leftarrow \tilde{B}$;
9: **End if**
10: **End For**
11: **Return** $(\hat{G}_D, B, \Phi_D, \Psi_T)$.

There are several variations for the local search and the neighbourhood structure. On one

hand, all links in a solution can be optimised or just one. On the other hand, a link can be reduced to its minimum value, or just reduced to the closest minor value. We have chosen to optimise the minimum unit possible per time, this is to say that we reduce if possible only one link only one unit.

The algorithm steps go as described in Alg-11.

Firstly, we take each data edge included in the present solution. For every data edge we analyse whether its assigned capacity is likely to be brought down until one is found. The capacity of that edge is modified to the next minor value from the set $\hat{B}$ only if the link is still able to carry its corresponding traffic in every failure scenario, even with its new reduced capacity value.

Part IV

RESULTS

Chapter 9

Test cases description

The test cases introduced in this chapter were created by the project team in order to evaluate all work and present all results over a consistent and permanent basis. In the interest of outlining different situations and realities of ANTEL, several variations in parameters that are under their control have been considered and are detailed next.

9.1. Scenarios Set 1

9.1.1. Demand

The Network shape is mainly defined by Residential Internet traffic. Other kinds of traffic such as 'Enterprise traffic' or 'Added Value traffic' (IPTV, VoIP, Cardales) has shown to be minor. Residential Internet traffic is dominant in the way that the presence or absence of the latter types of traffic in prototype problems only affects G_E in one of five times, and when G_E did changed, it was in a minimal way [27].

The reasons for such a low impact are the relative low number of clients or low bandwidth demanded in the case of Enterprise services. However, when it comes to Added Value services, the multicast nature of the majority of this traffic appear to spread efficiently in the backbone.

Consequently, the important parameters identified in order to test performance are clients and bandwidth, i.e. the amount of sold services and the associated bandwidth consumption to each service.

We will consider two test cases that take into account different Network demands that will be called `demL` and `demH`. The first one represents a low demand scenario and the second one a high demand scenario. Both are extreme but plausible scenarios in the context of ANTEL reality.

9.1.2. Requirements

In addition to demand requirements, the resulting Network form depends on the consumption of products offered by the telecommunication enterprise. In particular, on the variety of free and flexible products.

This concept was modelled as $z_Q(bw)$. Then, another variable we will deal with in order to build the test cases and analyse its dependence with results is the function z_Q.

We will name `zQL` and `zQH` the scenarios for a low z_Q design and a high z_Q design respectively. The data and functions are introduced in Section-9.2.

9.1.3. Contents

The cost of utilization of international links plays an important role when it comes to the origin or location of the content that clients demand in sight that ANTEL has to pay foreign companies to access content from abroad. On the other hand, if the content required by clients is national, then its cost is inferior.

As a result, a strategy followed in order to avoid part of the over-cost of hiring international connections is the existence of "caches" or other forms of national Internet content. So, the scenarios brought about acknowledge the percentage of national content demanded. The two alternatives are denoted `intL` and `intH`. The former stands for a 25 % of Internet content ending nationally in TI infrastructure located in AMM, while the latter represents the totality of Internet traffic being international.

9.1.4. Architecture

Another scheme to explore is related to different Network architectures. Nowadays, the ATM and IP/MPLS Networks have the primary objective of concentrate the HSI traffic towards another network, known as "Public IP Network". This network is the part of Internet inside Uruguay. It is physically located in four AMM stations called Aguada, Centro, Cordón and Unión, and the Americas NAP settled in Miami. The Public IP Network concentrate all international links and is responsible of routing Internet traffic.

Previously, with only ATM Network there were no options about this. However, the new MPLS Network has the potential feature of performing the functions of concentrating the HSI traffic and routing the Internet traffic, i.e. collapsing ATM and Public IP Networks in only one MPLS Network. Owing to this, it is interesting to evaluate if there is any economical benefit by merging the networks.

We will call `napL` the scheme where the present architecture is maintained or, contrarily, and `napH` the scheme where the IP/MPLS Network extends taking up the international spots of traffic exchange and performing both functions : gathering and routing of traffic.

9.2. Data for the problem

All data for the problem was supplied by ANTEL. The Transport Network G_T (nodes, edges, distances), available capacities $\hat{B}$ and their cost per kilometre $T : \hat{B} \to \mathbb{R}_0^+$. The Data Network has been mostly established by ANTEL and modified to be in accordance with some abstractions of the model and scenarios. The function z_Q and demand expectations were derived by the project team, see [27].

The Transport Network used as input for the run of algorithms and attainment of results is presented in Fig-9.1.

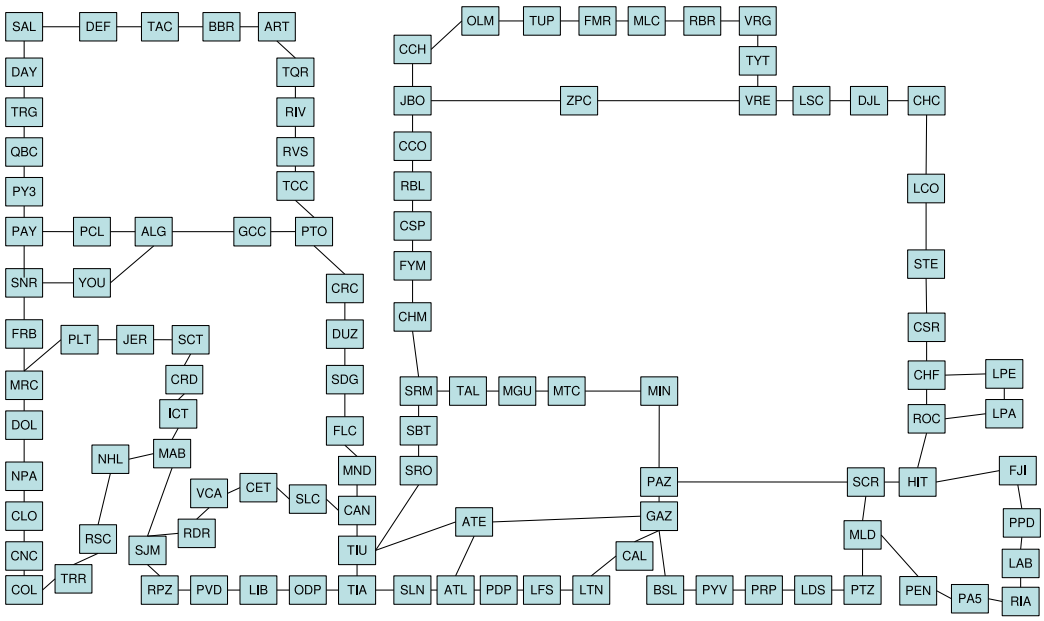


Figure 9.1 – ANTEL Transport Network - Uruguay

The Data Network consisting of data nodes and data edges was also established by ANTEL. Access nodes were set in almost every network station where ADSL service is offered over ATM infrastructure. Different edge nodes situation schemes have been handled as to represent AMM and Internet. AMM is represented by a fixed node and another fixed node called Internet stands for the place where international traffic ends. The rest of the nodes are optional. They include departmental capitals, important cities of the country, nodes with a transport station of degree three or more.

$\hat{B}$	**Speed** (Mbps)	**VCAT/DWDM Map**	**Useful Capacity** (Mbps)	**E1 Equivalent** (E1)	**Cost** (US$/km)
b_0	0	-	0	0	0.00
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	560	4 VC4	528	256	177.20
b_9	1,000	7 VC4	924	441	305.25
b_{10}	10,000	1 λ	10,000	5,263	104.00

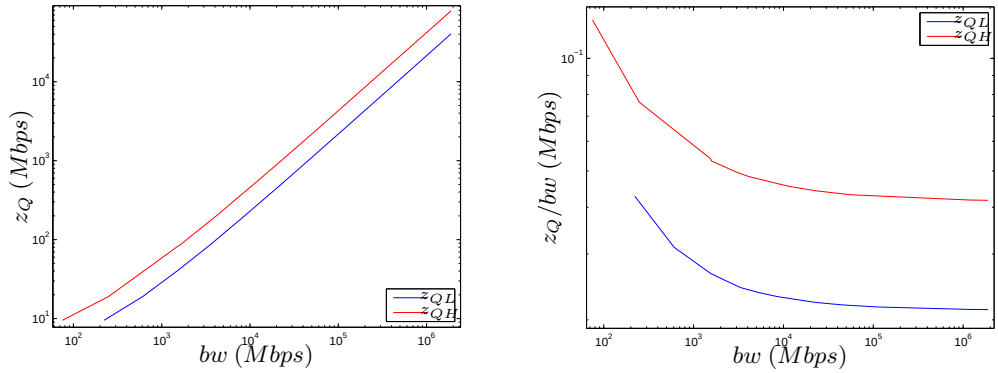
Table 9.1 – Transport Network Capacities and Costs.

In Table-9.1 the set $\hat{B}$ of available speeds and their costs are shown. In the algorithm executions b_8 and b_9 have been omitted due to the inconsistency in the assumption of a $T : \hat{B} \rightarrow \mathbb{R}_0^+$ strictly increasing function.

BW (Mbps)	$\mathbf{z_{QL}}$ (Mbps)	**BW** (Mbps)	$\mathbf{z_{QH}}$ (Mbps)
0	0	0	0
222	9.5	75	9.5
608	19.0	249	19.0
1547	41.2	1547	83.4
3095	76.1	1580	84.0
3450	84.0	2521	127.5
5390	127.5	3095	153.2
8670	200.0	4142	200.0
22900	510.0	11197	510.0
45588	1000.0	22587	1000.0
110792	2400.0	55548	2400.0
1161000	24777.0	1161000	48560.3
1896300	40411.9	1896300	79159.3

Table 9.2 – z_Q function for `zQL` and `zQH` scenarios.

In Table-9.2 we present the function brought about as statistical multiplexing factor that we called z_Q. Perhaps Fig-9.2 illustrate the function better.



(a) $z_Q(bw)$ for `zQL` and `zQH` (b) $z_Q(bw)/bw$ for `zQL` and `zQH`

Figure 9.2 – $z_Q(bw)$ and $z_Q(bw)/bw$ for `zQL` and `zQH` scenarios.

The `demL` and `demH` scenarios have the lowest demand projections of 1,084,860 Mbps sold and the highest demand projections of 1,896,300 Mbps sold respectively.

We have already explained `intL` and `intH`, and also `napL` and `napH` scenarios. Finally, combining `dem`, `zQ`, `int`, and `nap` low and high, we construct 16 different test cases to analyse. In Fig-9.3 we illustrate for the test case 02 (`02-demH_zQH_intH_napL`), the topology of the Transport Network, the data nodes and the potential data edges to include in the solution.

9.3. Scenarios Set 2

With the objective of testing the performance with another set of parameters, we have created a new set of test cases. For this purpose, we have divided the original Transport Network in the East Region and in the West Region, as shown in Fig-9.4, maintaining the ring transport topology and keeping the nodes TIU and TIA in both infrastructures. We have accordingly kept the data nodes of the each region and the induced data edges. We will distinguish between them in Table-10.1 by the names `east_copy` and `west_copy`.[1]

Then, we have introduced modifications. For example, we have created a new gamma of bandwidths going form 1000 to 10000 Mbps with a step of 1000 Mbps and cost proportional to bandwidth, see table Table-9.3 for a better understanding. This change is inspired on the distance between the largest capacity value and the next lower value in Set 1. This would allow for testing the local search that shrinks the data edges capacities to a minimum. Even though, we suggest the reader to be careful in sight we assume the results will be improved. We do

1. Despite we believe it would improve results, we did not created mesh topologies in the Transport Network to be coherent with the problem formulation.

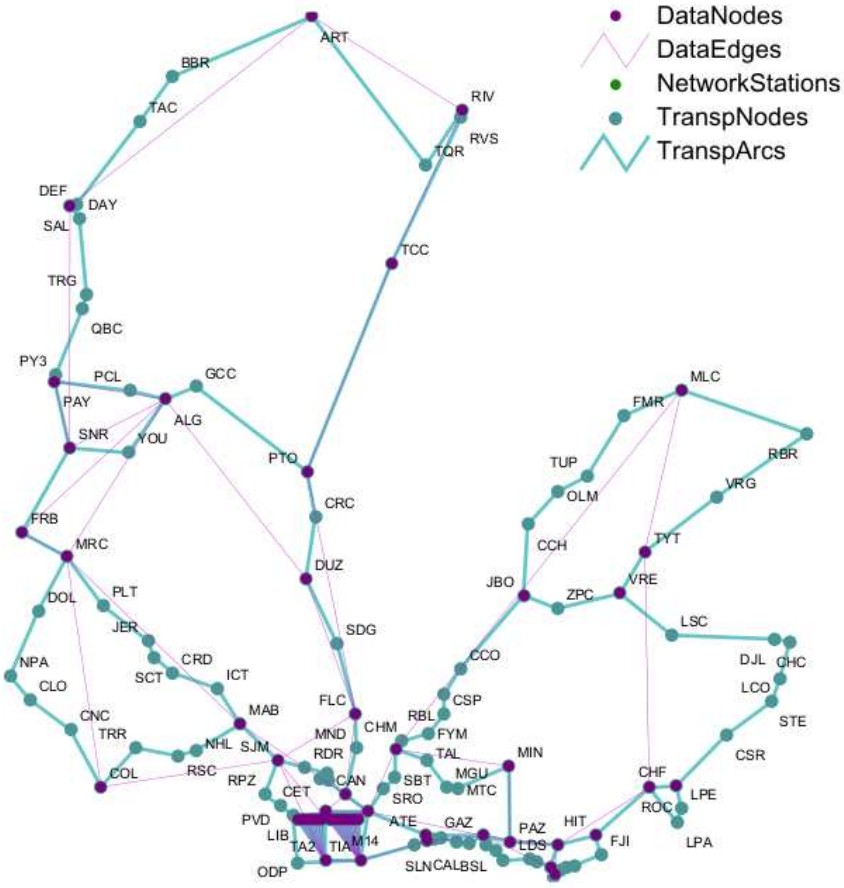
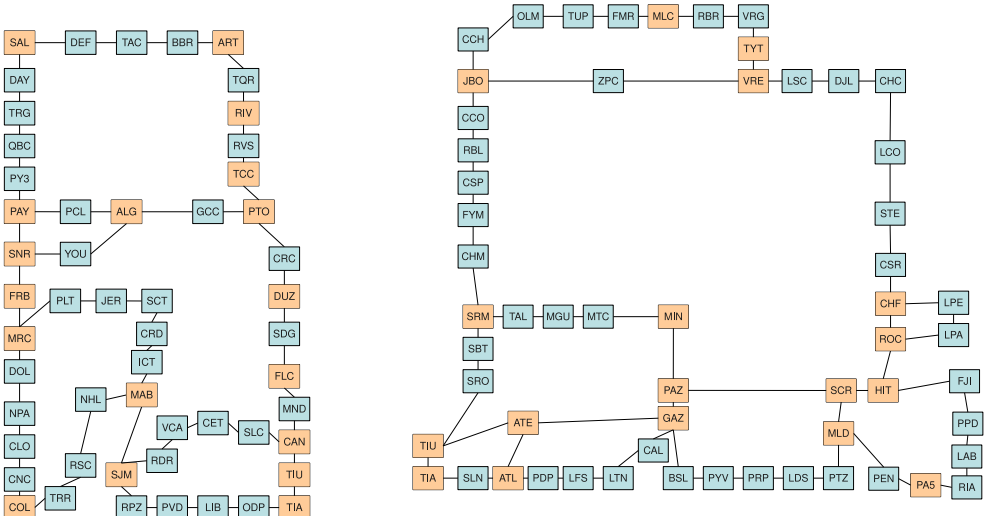


Figure 9.3 – ANTEL Transport Network - Test case 02

not want the reader to misunderstand the results, so we clarify that if the initial Construction Phase designed the edges with the maximum capacity and the charge of the network were small enough as to permit the local searches decrease all capacities to the minimum value, then the gap will be superior to approximately 90 %. Though we tried to avoid this extreme scenario. We recognise this test cases where new bandwidths were introduced by the string `cap` in its name.

On the other hand, we have built test cases where we completely modified the data graph, trying full-mesh topologies, `fm`, and "half-full-mesh" topologies, `hfm`. For the latter scheme, we have removed half the totality of data edges of the complete graph. Here, we expect that the introduction of potential data tunnels will also diminish the cost, as the local searches would have to evaluate more available movements between solutions.

At last, for some of the new scenarios, we have changed its original traffic demand. In



(a) West Region (b) East Region

Figure 9.4 – ANTEL Transport Network.

comparison with the initial state, we have increased the traffic weight of the network. We have randomly assigned traffic uniformly distributed from 0 to 30,000 Mbps to couples of nodes. The number of nodes with traffic demand is half the number of data edges. For those test cases we will append the string `charge`, giving the notion that in this networks almost all edges will be designed and charged, if feasible, near to its maximum capacity.

$\hat{\mathbf{B}}$	**Speed** (Mbps)	**Cost** (US$/km)
b_0	0	0
b_1	1,000	10
b_2	2,000	20
b_3	3,000	30
b_4	4,000	40
b_5	5,000	50
b_6	6,000	60
b_7	7,000	70
b_8	8,000	80
b_9	9,000	90
b_{10}	10,000	100

Table 9.3 – Linear Transport Network Capacities and Costs.

Chapter 10

Performance tests and analysis of results

10.1. Parameters of testing

In this chapter we present the experimental results obtained when applying the algorithms implemented with the scheme previously described.

The results obtained for the first set of test cases introduced, Set 1, are similar to those presented in [27]. There, results are presented form an economical point of view, analysing the benefits and strategies that ANTEL could follow by making use of a tool as powerful as this one. This analysis, numbers, data, results, recommendations and insight generated has already been elevated to the enterprise counterpart.

In this work, we preferred to focus on a more computational perspective, building different semi-random test cases comprising the Set 2.

Another point is the choice of the programming language. This is a fundamental decision because it strongly determines the process of development of an algorithm and bounds its performance. There is a compromise among factors to consider. This problem needs to be implemented in a language that efficiently caters for the treatment of large amounts of data. It is also desirable that it were a structured language and object oriented. Finally, despite sometimes difficult or delicate, the requirements aforementioned turn *C++* the sensible programming language for combinatorial optimisation algorithms par excellence.

Hence, all the algorithms were implemented in *C++* language, and tested to be compiled and run on both *Windows* and *Linux* operating systems. The actual results included in this thesis were obtained on an *Intel Core2 Duo* processor with 2 Gbytes of RAM, running under *Ubuntu 9.10, GNULinux* operating system with kernel *2.6.31-22-generic*.

In the performance testing phase all instances were solved with the same GRASP parame-

ter settings. The number of iterations for the VNS phase will be called $MaxIter$.

The parameters we will employ for showing results are :
– Gap of the Construction Phase and the Local Search Phase for problem p :

$$gap(p) = \frac{CC - LC}{CC},$$

CC and LC being the cost of the solution of the the Construction Phase and the Local Search Phase respectively.
– Total Running time, T.
– Besides, we will display the average of the defined parameters over all test cases.

10.2. Results Set 1

We run our algorithm over the Set 1 with just a couple of iterations. The average gap for the first set of test cases is $Av\_gap = 3\,\%$. The average total running time is $Av\_T = 17'7''$. We recognise we did not expected more satisfactory results in sight of the narrow margin of action, since we glimpsed any feasible solution for ANTEL networks were nearly optimal. Furthermore, considering the size of the networks we are treating, the times involved are proficient.

We point out that the knowledge of the test cases and the algorithm for the construction of an initial feasible solution enable us to draw conclusions of this results. Firstly, we remark that the capacity reduction local search is the most efficient. This is due to the fact that the initial construction design data edges with maximum capacity. Moreover, the other local searches related to data edge movements reduces the cost of the solutions in most cases too. However, we notice that the remaining local searches, which attempt to modify the transport paths do not improve the solution almost in any case. The reason for this is the ring topology used for the Transport Network. When a short transport path is found, it is rare to find a better one. Besides, we know the Construction Phase utilise the Dijkstra algorithm for routing, so in this conditions the transport paths are seldom adjusted.

10.3. Results Set 2

For this set of test cases, the number of iterations was established in a previous tuning phase with a $MaxIter$ value. In example networks the Local Searches did not achieve further improvements beyond 10 iterations. Table-10.1 summarises the information gathered and the parameters calculated.

We begin announcing that the previous tendency in the performance of the local searches is preserved. Since we continued with ring transport topologies, the local searches that try to change transport paths are useless. Nevertheless, we gladly say that the local searches concerning data edge modifications or bandwidth adjustment achieve improvements iteratively.

Test case	**Feasible ?**	**gap**	T
`east_copy`	Yes	13 %	5”
`east_copy_cap`	Yes	13 %	7”
`east_copy_charge_cap`	Yes	0 %	4’
`east_fm`	Yes	64 %	1’50”
`east_fm_cap`	Yes	64 %	2’16”
`east_fm_charge`	Yes	27 %	2’54”
`east_fm_charge_cap`	Yes	39 %	8’56”
`east_hfm`	No	-	0.6”
`east_hfm_cap`	No	-	0.5”
`east_hfm_charge`	No	-	0.6”
`east_hfm_charge_cap`	No	-	0.6”
`west_copy`	Yes	44 %	45”
`west_copy_cap`	Yes	52 %	33”
`west_copy_charge_cap`	Yes	54 %	34”
`west_fm`	Yes	56 %	13’43”
`west_fm_cap`	Yes	59 %	13’34”
`west_fm_charge`	Yes	27 %	5’41”
`west_fm_charge_cap`	Yes	31 %	2’28”
`west_hfm`	No	-	0.5”
`west_hfm_cap`	No	-	1.3”
`west_hfm_charge`	No	-	0.6”
`west_hfm_charge_cap`	No	-	1.6”
Average *	-	**39 %**	**2’50**

$*$. Average over feasible scenarios.

Table 10.1 – Set 2 results.

As we can see, some of the networks generated are not feasible, which is coherent by the charge imposed, and was found by our algorithm. An interesting result is that those ones were the scenarios with half the possible data edges. Another positive thing is the short time, about one second, the algorithm takes to decide an input is a non-feasible network.

The average time it takes the entire algorithm to return an optimal solution relative to all local searches is about $Av\_T = 2'50''$, and the average gap is $Av\_gap = 39\,\%$. In spite of prising this number, it is of order to be critical stating that we varied the networks introducing the test cases of Set 2 specially with the aim of allowing the local searches to shine a little and demonstrate they work when there is any chance.

Entering into detail, we perceive that in general, if there are more data edges to choose, the gap is better and it algorithm takes more time. This is confirmed by the obtained results comparing the `copy` test cases with only some potential data edges, with the full-mesh `fm` test

cases with all the potential data edges to select.

Besides, we notice that the results are better in most `cap` cases as expected.

Chapter 11

Conclusions

11.1. Conclusions of the implemented solution

Firstly, we firmly believe that given the scope and complexity of the problem, the presented solution accomplishes the objectives settled at the beginning of this work. Especially, we have created a set of local searches that manage to performably reach relative optimal solutions in a short time for real networks.

Clearly, all the classes and functions implemented are likely to be improved. Although a great effort and care has been taken to efficiently control speed and memory, the amount of routines, structures and processes turned it quite a complicated task.

Besides, we think relevant that the global solution accepts any type of network as input, admitting any set of distances, capacities, topologies, traffic, etc. The only constraints are the planarity of the Transport Network and two-edge-connectivity of both networks. Even though, this is not an actual restriction as the algorithm would characterise the network as non-feasible.

To conclude, we remark that this thesis solves a difficult real problem, developing a tool that may improve the decision taking policy of the National Telephone enterprise. For instance, it could be viable to identify structural problems in the network and simulate the performance of a plausible solution beforehand. It could help to determine the economically best strategy of business to follow with regards quality of services. It could justify inversions objectively and measure its impact, or evaluate robustness and cost of an infrastructure modification. It could serve for sizing and routing planning over the Transport Network or provide strategic guidelines based on optimality principles to medium or long term projects. Finally, another possible utilization may be for example to evaluate quantitatively the cost of expanding ANTEL services to each point in the country.

11.2. Extensions and future work

Concerning the work of this thesis, we sense that the local searches implemented tried to take into account all the variables of the problems reducing the cost of each dimension as simple and efficiently as possible. Even though, we have not been exhaustive when experimenting variations. We judge that the local search implementations may be improved, other VNS schemes may return better results, in shorter times, or other local searches may turn out to outperform those of Chapter-8. So we feel that this material is of interest up to a point and there is lot of work which exceeds the academic expectations and remains to be continued as a future extension.

Neither the time nor the scope of this thesis or the embracing project has allowed to study the problem in a deeper manner as we may have desired and the problem deserved. However, we know that several different techniques are being pursued in a parallel way for the solution of MORN. Different metaheuristics are being implemented, the exact solution to relaxations of the constrains are also being tested. We believe that this project has originated a great amount of different courses of action that look for a consistent solution.

As future work, it is always possible to continue polishing the algorithms implemented in order to diminish execution times, CPU and memory load.

Another imaginable extension could incredibly enhance the amiability of the program. The functions implemented were not formerly thought to be used by an enterprise employee but they were planned for academic purpose to be handled by an expert user. If a graphical user interface were implemented, with a friendly process for data input and a simple graphical output, showing maps, routes and capacities, appended with a User Manual, this application could be introduced to any levels of the enterprise. We believe this would impressively exploit the potential of this tool.

Ultimately, as stated in last section, the most important tasks to conclude are tied to the analysis of results applied to the enterprise network. A huge variety of new scenarios can be thought for employing this developed tool which may lead to crucial decisions and future strategies.

11.3. Personal experience

Even though we had already gone through some diverse labouring experiences or through a few Engineering projects from beginning to end, the experience of carrying out this work has been invaluably enriching. After all, every new experience has its challenges, sacrifices and fruitful profits. In my particular case, since I had my degree in Electrical Engineering, I had never faced a real world computational enterprise of such large-scale like this one.

Moreover, all the process of resolution for this problem has been a novelty indeed. The

initial analysis of this kind of problem, the inspection of the main parameters and abstractions, the mathematical model formulation and study of complexity, the optimisation techniques and schemes, the implementation, the data acquisition and test cases creation are all generally quite far from the formation received during degree studies.

From the mathematical standpoint, it has been notably positive. During this thesis several interesting issues have appeared related to graph theory and topological design of networks, computational complexity theory, metaheuristics and combinatorial optimization to name just a few examples.

On the other hand, this thesis has had high requirements of software development. Being this aspect a fundamental component in the professional life of an Engineer, this work has favoured and stimulated the consolidation of this area, sometimes poor in the Electrical Engineering curricula.

Finally, we mention briefly that being this thesis part of a major Engineering project that involves both enterprise interests and academical ones, the aptitudes for team work, oral and written expression in presentations and documentation, have been nourished by this activity.

Appendix A

Test case examples

A.1. East Network

```
# Problema OPTMPLS (Formato estandar)
#-----------------------------------
# Las definiciones de problemas se establecen en archivo de texto
# con estructura de modelo relacional (RDBMS).
# Para varias estructuras se establece explicitamente el cardinal
# de elementos a modo de control y resumen.
# La notacion usada es
#  '#' como comentario hasta fin de linea.
#  'name' como palabra clave previa al nombre de estructura
# ':=' como separador entre nombre y datos
# ';' como final de datos
# Entre parentesis se establece el formato de las tuplas
# con los nombres de dominio separados por comas
# ' ' y/o '\t'  como separadores de datos en la tupla
# '\n' o '\r\n' como separadores de tuplas


# Nombre de la Red de Datos
name DataName := CasoDePrueba1;
# Nombre de la Red de Transporte
name TranspName := Transporte_este;
# Usuario generador
name User := acorez;
# Fecha de generacion
name Date := 09Set2010;
# Observacion
name Obs := Red este;

# Cantidad de nodos de Datos
name nDataNodes := 18;
# Nodos de Datos
name DataNodes (DataNodeId, DataNodeName, IsReal) :=
```

```
0 EDGENDATE 1
1 EDGENDATL 1
2 EDGENDCHF 1
3 EDGENDGAZ 1
4 EDGENDHIT 1
5 EDGENDJBO 1
6 EDGENDMIN 1
7 EDGENDMLC 1
8 EDGENDMLD 1
9 EDGENDPA5 1
10 EDGENDPAZ 1
11 EDGENDROC 1
12 EDGENDSCR 1
13 EDGENDSRM 1
14 EDGENDTIA 1
15 EDGENDTIU 1
16 EDGENDTYT 1
17 EDGENDVRE 1
;

# Nodos Edge de Datos
name DataEdgeNodes (DataNodeId) :=
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
;

# Nodos Acceso de Datos
name DataAccessNodes (DataNodeId) :=
;

# Cantidad de Aristas de Datos
name nDataArcs := 15;
# Aristas de Datos
```

```
name DataEdges (DataEdgeId, OrgDataNodeId, DestDataNodeId) :=
0 1 3
1 1 14
2 1 15
3 3 8
4 6 10
5 6 13
6 7 13
7 7 16
8 8 9
9 9 12
10 10 12
11 11 12
12 11 16
13 13 15
14 14 15
;

# Cantidad de Requisitos de Trafico
name nTraffic := 11;
# Trafico
name Traffic (TrafficId, OrgDataNodeId, DestDataNodeId, Committed, Excess) :=
0 1 3 450 0
1 1 15 450 0
2 3 8 450 0
3 6 10 450 0
4 6 13 450 0
5 7 16 450 0
6 8 9 450 0
7 10 12 450 0
8 11 12 450 0
9 11 16 450 0
10 13 15 450 0
;

# Cantidad de Nodos de Transporte
name nTranspNodes := 58;
# Nodos de Transporte
name TranspNodes (TranspNodeId, TranspNodeName, CoordX, CoordY) :=
0 SRO 478.17 6183.25
1 SBT 485.24 6189.78
2 SRM 485.42 6206.56
3 CHM 489.22 6211.45
4 FYM 505.05 6215.95
5 CSP 513.95 6227.38
6 RBL 513.92 6238.58
7 CCO 523.46 6253.90
8 JBO 560.64 6297.05
9 CCH 563.05 6338.70
```

```
10 OLM 581.05 6358.30
11 TUP 597.9 6367.00
12 FMR 620.05 6402.65
13 MLC 653.73 6417.96
14 RBR 727.5 6392.40
15 VRG 674.06 6354.70
16 TYT 632.05 6322.75
17 VRE 617.6 6298.28
18 LSC 647.77 6273.70
19 DJL 708.35 6271.42
20 CHC 717.42 6269.90
21 LCO 711.35 6247.65
22 STE 706.98 6234.75
23 CSR 679.93 6215.05
24 CHF 650.45 6185.16
25 LPE 653.5 6171.67
26 LPA 650.75 6163.92
27 ROC 634.68 6184.18
28 HIT 603 6156.00
29 FJI 606.23 6144.56
30 PPD 590.4 6138.35
31 LAB 586 6136.80
32 RIA 582.27 6135.42
33 PA5 578.77 6133.24
34 PEN 578.31 6131.42
35 MLD 576.5 6137.50
36 PTZ 568.35 6140.70
37 LDS 564.37 6142.48
38 PRP 548.15 6140.95
39 PYV 544.57 6147.23
40 BSL 538.6 6150.10
41 GAZ 536.55 6155.80
42 CAL 528.51 6150.82
43 LTN 521.5 6152.18
44 LFS 511.88 6154.60
45 PDP 507.74 6153.88
46 ATL 503.62 6152.56
47 SLN 496.12 6150.0
48 TIA 464.7 6141.26
49 TIU 469.5 6170.0
50 ATE 503.38 6156.24
51 TAL 503.71 6200.23
52 MGU 515.46 6184.31
53 MTC 522.04 6183.85
54 MIN 552 6196.65
55 PAZ 552.7 6152.28
56 SCR 580.93 6150.30
57 ZPC 580.53 6289.45
;
```

```
# Cantidad de Aristas de Transporte
name nTranspArcs := 65;
# Aristas de Transporte
name TranspArcs (TranspArcId, OrgTranspNodeId, DestTranspNodeId, Length) :=
0 0 1 11.28
1 1 2 19.08
2 2 3 7.53
3 3 4 11
4 4 5 10.6
5 5 6 12.76
6 6 7 23.67
7 7 8 82.12
8 8 9 52.4
9 9 10 31.23
10 10 11 21.9
11 11 12 48.9
12 12 13 42.74
13 13 14 91.52
14 14 15 47
15 15 16 58
16 16 17 31.6
17 17 18 41.27
18 18 19 77.34
19 19 20 10.5
20 20 21 25.35
21 21 22 20
22 22 23 40
23 23 24 49.85
24 24 25 20
25 25 26 11.45
26 26 27 26
27 27 28 45.66
28 28 29 22.84
29 29 30 20.14
30 30 31 6.38
31 31 32 6.26
32 32 33 6.56
33 33 34 2.42
34 34 35 8.78
35 35 36 11.3
36 36 37 5
37 37 38 24
38 38 39 7.5
39 39 40 7
40 40 41 7.67
41 41 42 15.4
42 42 43 9.44
43 43 44 10.5
```

```
44 44 45 5.98
45 45 46 4.98
46 46 47 19
47 47 48 66.46
48 48 49 177.94
49 49 0 51.37
50 49 50 77.78
51 50 46 25
52 50 41 42.15
53 2 51 20.5
54 51 52 21.94
55 52 53 13.4
56 53 54 47.81
57 54 55 49.88
58 55 41 21.34
59 55 56 37.26
60 56 35 81.97
61 56 28 25.5
62 24 27 20.05
63 17 57 51.92
64 57 8 28.8
;

# Cantidad de Tecnologias
name nTechnology := 2;
# Tecnologias
name Techno (TechnoId, Bandwidth, Price)
0 1000 59
1 10000 104
;

# Cantidad de NetworkStations
name nNetworkStation := 18;
# NetworkStations
name NetworkStations (NetworkStationId, DataNodeId, TranspNodeId) :=
0 0 50
1 1 46
2 2 24
3 3 41
4 4 28
5 5 8
6 6 54
7 7 13
8 8 35
9 9 33
10 10 55
11 11 27
12 12 56
13 13 2
```

```
14 14 48
15 15 49
16 16 16
17 17 17
;

# Factor de multiplexado estadistico
name ZetaQ (BW, ZQ) :=
0 0
75 9.5
249 19.0
1547 83.4
1580 84.0
2521 127.5
3095 153.2
4142 200.0
11197 510.0
22587 1000.0
55548 2400.0
1161000 48560.3
1896300 79159.3
;
```

A.2. West Network

```
# Problema OPTMPLS (Formato estandar)
#-----------------------------------
# Las definiciones de problemas se establecen en archivo de texto
# con estructura de modelo relacional (RDBMS).
# Para varias estructuras se establece explicitamente el cardinal
# de elementos a modo de control y resumen.
# La notacion usada es
#  '#' como comentario hasta fin de linea.
#  'name' como palabra clave previa al nombre de estructura
# ':=' como separador entre nombre y datos
# ';' como final de datos
# Entre parentesis se establece el formato de las tuplas
# con los nombres de dominio separados por comas
# ' ' y/o '\t'  como separadores de datos en la tupla
# '\n' o '\r\n' como separadores de tuplas


# Nombre de la Red de Datos
name DataName := CasoDePrueba1;
# Nombre de la Red de Transporte
name TranspName := Transporte_oeste;
# Usuario generador
name User := acorez;
# Fecha de generacion
```

```
name Date := 09Set2010;
# Observacion
name Obs := Red oeste;


# Cantidad de nodos de Datos
name nDataNodes := 18;
# Nodos de Datos
name DataNodes (DataNodeId, DataNodeName, IsReal) :=
0 EDGENDALG 1
1 EDGENDART 1
2 EDGENDCAN 1
3 EDGENDCOL 1
4 EDGENDDUZ 1
5 EDGENDFLC 1
6 EDGENDFRB 1
7 EDGENDMAB 1
8 EDGENDMRC 1
9 EDGENDPAY 1
10 EDGENDPTO 1
11 EDGENDRIV 1
12 EDGENDSAL 1
13 EDGENDSJM 1
14 EDGENDSNR 1
15 EDGENDTCC 1
16 EDGENDTIA 1
17 EDGENDTIU 1
;

# Nodos Edge de Datos
name DataEdgeNodes (DataNodeId) :=
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

```
;

# Nodos Acceso de Datos
name DataAccessNodes (DataNodeId) :=
;

# Cantidad de Aristas de Datos
name nDataArcs := 47;
# Aristas de Datos
name DataEdges (DataEdgeId, OrgDataNodeId, DestDataNodeId) :=
0 0 2
1 0 4
2 0 5
3 0 6
4 0 8
5 0 9
6 0 14
7 1 11
8 1 12
9 1 15
10 2 5
11 2 7
12 2 10
13 2 16
14 2 17
15 3 6
16 3 7
17 3 8
18 3 13
19 4 5
20 4 10
21 4 17
22 5 10
23 5 13
24 5 15
25 5 16
26 5 17
27 6 7
28 6 8
29 6 9
30 6 13
31 7 8
32 7 13
33 7 14
34 7 16
35 7 17
36 8 13
37 8 14
38 9 12
```

```
39 9 14
40 10 11
41 10 14
42 10 15
43 11 15
44 12 14
45 13 16
46 13 17
;

# Cantidad de Requisitos de Trafico
name nTraffic := 26;
# Trafico
name Traffic (TrafficId, OrgDataNodeId, DestDataNodeId, Committed, Excess) :=
0 0 2 450 0
1 0 4 450 0
2 0 5 450 0
3 0 9 450 0
4 1 11 450 0
5 1 12 450 0
6 2 7 450 0
7 2 17 450 0
8 3 7 450 0
9 3 8 450 0
10 3 13 450 0
11 4 5 450 0
12 5 10 450 0
13 5 15 450 0
14 5 17 450 0
15 6 8 450 0
16 6 9 450 0
17 7 13 450 0
18 7 16 450 0
19 7 17 450 0
20 9 12 450 0
21 10 14 450 0
22 10 15 450 0
23 11 15 450 0
24 12 14 450 0
25 13 17 450 0
;

# Cantidad de Nodos de Transporte
name nTranspNodes := 53;
# Nodos de Transporte
name TranspNodes (TranspNodeId, TranspNodeName, CoordX, CoordY) :=
0 SAL 294.04 6526.38
1 DAY 299.45 6518.73
2 TRG 303.77 6474.14
```

```
3 QBC 301.53 6465.88
4 PY3 285.55 6426.53
5 PAY 284.75 6422.78
6 SNR 293.75 6383.83
7 FRB 265.53 6333.98
8 MRC 292.03 6319.61
9 DOL 275.3 6288
10 NPA 258.71 6249.75
11 CLO 270.35 6235.83
12 CNC 294.3 6218.1
13 COL 312.05 6184.55
14 TRR 332.95 6207.57
15 RSC 357.1 6202.8
16 NHL 367.85 6205.75
17 MAB 394.1 6221.4
18 SJM 416.15 6200.45
19 RPZ 408.51 6179.97
20 PVD 417.55 6173.35
21 LIB 425.06 6167.98
22 ODP 427.73 6140
23 TIA 464.7 6141.26
24 TIU 469.5 6170
25 CAN 455.85 6179.94
26 MND 462.1 6207.3
27 FLC 461.85 6227.5
28 SDG 450.9 6269.1
29 DUZ 432.7 6307.15
30 CRC 438.35 6343.55
31 PTO 433.33 6370.05
32 TCC 482.85 6492
33 RVS 523.87 6577.78
34 RIV 524.73 6582.28
35 TQR 503.25 6550
36 ART 436.18 6637.4
37 BBR 353.88 6601.95
38 TAC 335.2 6575.8
39 DEF 297.84 6526.78
40 RDR 431.9 6195.8
41 VCA 445.02 6192.64
42 CET 441.12 6189.42
43 SLC 445.54 6188.28
44 ICT 380.75 6242.15
45 CRD 354.52 6251.71
46 SCT 343.32 6260.72
47 JER 339.65 6270.38
48 PLT 313.33 6291.1
49 GCC 368.05 6419.78
50 ALG 349.93 6412.4
51 YOU 328.43 6381
```

```
52 PCL 329.63 6417.33
;

# Cantidad de Aristas de Transporte
name nTranspArcs := 57;
# Aristas de Transporte
name TranspArcs (TranspArcId, OrgTranspNodeId, DestTranspNodeId, Length) :=
0 0 1 40.14
1 0 39 90.3
2 1 2 135.23
3 2 3 60.66
4 3 4 164.1
5 4 5 89.53
6 5 6 14.96
7 5 52 43.64
8 6 7 110.5
9 6 51 118.2
10 7 8 35.49
11 8 9 139.92
12 8 48 144.38
13 9 10 64.35
14 10 11 168.79
15 11 12 85.88
16 12 13 10.31
17 13 14 113.74
18 14 15 38.18
19 15 16 142.61
20 16 17 67.4
21 17 18 163.14
22 18 19 88.57
23 19 20 13
24 20 21 117.43
25 21 22 42.87
26 22 23 138.96
27 23 24 63.39
28 24 25 167.83
29 25 26 92.26
30 25 43 65.12
31 26 27 16.69
32 27 28 112.78
33 28 29 37.22
34 29 30 141.65
35 30 31 66.8
36 31 32 170.52
37 32 33 95.95
38 33 34 12.4
39 34 35 116.47
40 35 36 41.91
41 36 37 145.34
```

```
42 37 38 70.77
43 38 39 165.86
44 18 40 15.73
45 40 41 119.16
46 41 42 44.6
47 42 43 140.69
48 17 44 169.55
49 44 45 94.99
50 45 46 19.42
51 46 47 123.85
52 47 48 40.95
53 31 49 68.81
54 49 50 172.25
55 50 51 97.68
56 50 52 14.77
;

# Cantidad de Tecnologias
name nTechnology := 2;
# Tecnologias
name Techno (TechnoId, Bandwidth, Price)
0 1000 59
1 10000 104
;

# Cantidad de NetworkStations
name nNetworkStation := 18;
# NetworkStations
name NetworkStations (NetworkStationId, DataNodeId, TranspNodeId) :=
0 0 50
1 1 36
2 2 25
3 3 13
4 4 29
5 5 27
6 6 7
7 7 17
8 8 8
9 9 5
10 10 31
11 11 34
12 12 0
13 13 18
14 14 6
15 15 32
16 16 23
17 17 24
;
```

```
# Factor de multiplexado estadistico
name ZetaQ (BW, ZQ) :=
0 0
75 9.5
249 19.0
1547 83.4
1580 84.0
2521 127.5
3095 153.2
4142 200.0
11197 510.0
22587 1000.0
55548 2400.0
1161000 48560.3
1896300 79159.3
;
```

Bibliography

[1] D. Alevras, M. Grötschel, and R. Wessäly. A network dimensioning tool. In *Preprint SC 96-49, Konrad-Zuse-Zentrum für Informationstechnik*, Berlin, 1996.

[2] M. Baïou. *Le problème du sous-graphe Steiner 2-arête-connexe : approche polyédrale*. PhD thesis, Université de Rennes I, Rennes, France, 1996.

[3] M. Baïou. On the dominant of the steiner 2-edge connected subgraph polytope. *Discrete Applied Mathematics*, 112(1-3) :3 – 10, 2001.

[4] H. Cancela, F. Robledo, and G. Rubino. A GRASP algorithm with RNN based local search for designing a WAN access network. *Electronic Notes in Discrete Mathematics*, 18 :59–65, 2004.

[5] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, third edition, 2005.

[6] M. Dorigo. *Optimization, Learning and Natural Algorithms (in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[7] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

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

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[10] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[11] P. Hansen and N. Mladenović. Variable neighborhood search, a chapter of "handbook of metaheuristics", 1997.

[12] P. Hansen, N. Mladenović, and J.A. Moreno. Búsqueda de entorno variable. *Inteligencia Artificial, Revista Iberoamericana de IA*, 7(19) :77–92, 2003.

[13] J. H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. The University of Michigan Press, Ann Arbor, MI, 1975.

[14] A. Imran, S. Salhi, and N. A. Wassan. A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, 197(2) :509–518, 2009.

[15] Y. B. Ji and M. L. Liu. A new scheme for the steiner problem in graphs. In *Circuits and Systems, 1988., IEEE International Symposium on*, pages 1839 –1842 vol.2, 7-9 1988.

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

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

[18] P. J. M. Laarhoven and E. H. L. Aarts, editors. *Simulated annealing : theory and applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[19] Z. Ma, J. Chen, Y. R. Yang, and A. Krishnamurthy. Optimal capacity sharing of networks with multiple overlays. In *IWQoS*, pages 72–81, 2006.

[20] N. Mladenović. A variable neighborhood algorithm - a new metaheuristic for combinatorial. abstracts of papers presented at optimization days, 1997.

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

[22] C. L. Monma, B. S. Munson, and W. R. Pulleyblank. Minimum-weight two-connected spanning networks. *Math. Program.*, 46(2) :153–171, 1990.

[23] J. A. Moreno-Pérez, P. Hansen, and N. Mladenović. Parallel variable neighborhood searches, 2004.

[24] C. R. Reeves, editor. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., New York, NY, USA, 1993.

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

[26] M. G. C. Resende and C. C. Ribeiro. An annotated bibliography of grasp. Technical report, AT&T Labs Research, Florham Park, NJ 07932, 2004.

[27] C. Risso. Optimización de costos en redes multicapa robustas. Master's thesis, Facultad de Ingeniería, UdelaR, 2010.

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

[29] Sivanandam and Deepa. *Introduction to Genetic Algorithms*. Springer Publishing Company, Incorporated, 2007.

[30] M. Stoer. *Design of Survivable Networks*, volume 1531 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1992.

[31] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 1995.

List of figures

2.1. OSI Model Layers . . . 14
2.2. Red Overlay . . . 15

3.1. Access and edge data node representation . . . 20
3.2. Data node chart . . . 21
3.3. Example: Data Network . . . 22
3.4. Transport node representation . . . 22
3.5. Example: Traffic Network . . . 23
3.6. Cost function . . . 26
3.7. Routing example . . . 27
3.8. Simple transport failure . . . 27

4.1. Example: Disbalanced Transport Network . . . 35
4.2. Example: Balanced Transport Network . . . 36
4.3. Illustration of elements in proof of Theorem 4.4.4 . . . 40

6.1. Objective function . . . 58
6.2. Local search scheme . . . 60
6.3. Local search falling in local optimum . . . 60

7.1. General metaheuristic block diagram . . . 66
7.2. GRASP block diagram . . . 67
7.3. Dual Graph Example . . . 68
7.4. Primal West Transport Network $\rightarrow$ Dual West Transport Network . . . 69
7.5. G_E reduction block diagram . . . 71

8.1. LS: Edge Elimination . . . 74
8.2. Path Reduction Scheme . . . 76
8.3. LS: Link Decomposition . . . 77
8.4. LS:Insertion/Elimination . . . 80

9.1. ANTEL Transport Network - Uruguay . . . 87
9.2. $z_Q(bw)$ and $z_Q(bw)/bw$ for zQL and zQH scenarios. . . . 89
9.3. ANTEL Transport Network - Test case 02 . . . 90
9.4. ANTEL Transport Network. . . . 91