JOINT ROUTING AND RESOURCE ALLOCATION FOR WIRELESS BACKHAULING OF SMALL CELL NETWORKS

A Degree Thesis submitted to the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Dídac Surís

In partial fulfilment

of the requirements for the degree in

Science and Telecommunication Technologies Engineering

Advisor: Josep Vidal

Co-advisor: Adrián Agustín

Barcelona, June 2016

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

ABSTRACT – RESUMEN – RESUM

The future communication networks are destined to support an increasingly large amount of data traffic, and for that reason, efficient mechanisms to manage them are necessary. Based on a backhaul network, and starting from specific scenarios, we develop methods to jointly optimize the routing parameters and resources of this network. We relate this optimization with the *Software Defined Networks* and *network virtualization* concepts, which allow us to have an overall vision of the network, and lead us to study its decomposition. To do this, we use convex optimization techniques, which have very efficient resolution mechanisms, and help us to obtain tools for interpreting the obtained results and perform analysis on the network parameters. The achieved results show a great improvement in relation to the non-optimized case in terms of carried traffic, which is an assessment we make in the final economic analysis.

Las redes de comunicaciones del futuro están destinadas a soportar una cantidad de tráfico de datos cada vez más elevada, y por eso son necesarios mecanismos eficientes para gestionarlas. Basándonos en una red de *backhaul* y partiendo de escenarios concretos, desarrollamos métodos para optimizar conjuntamente los parámetros de enrutamiento y los recursos de esta red. Esta optimización la ligamos con los conceptos de *Software Defined Networksk y* de *network virtualization*, que nos permiten tener una visión general de la red, y nos conducen a estudiar su descomposición. Esto lo hacemos usando técnicas de optimización convexa, que tiene mecanismos de resolución muy eficientes, y nos ayuda a obtener herramientas para interpretar los resultados obtenidos y hacer análisis de los parámetros de la red. Los resultados conseguidos muestran una gran mejora con relación al caso no optimizado en términos de tráfico transportado, valoración que recogemos en un análisis económico final.

Les xarxes de comunicacions del futur estan destinades a suportar una quantitat de trànsit de dades cada cop més elevada, i per això són necessaris mecanismes eficients per a gestionar-les. Basant-nos en una xarxa de *backhaul* i partint d'escenaris concrets, desenvolupem mètodes per a optimitzar conjuntament els paràmetres d'encaminament i els recursos d'aquesta xarxa. Aquesta optimització la lliguem amb els conceptes de *Software Defined Network* i de *network virtualization*, que ens permeten tenir una visió general de la xarxa, i ens condueixen a estudiar-ne la seva descomposició. Tot això ho fem utilitzant tècniques d'optimització convexa, que té mecanismes de resolució molt eficients, i ens ajuda a obtenir eines per a interpretar els resultats obtinguts i fer anàlisis dels punts forts i febles de la xarxa. Els resultats aconseguits mostren una gran millora respecte el cas no optimitzat en termes de trànsit transportat, valoració que recollim en una anàlisi econòmica final.

ACKNOWLEDGEMENTS

Firstly, I would like to express my gratitude to my advisors, Josep Vidal and Adrián Agustín, for their continuous support, advice and guidance throughout the project, and for spending some of their precious hours listening to my (sometimes senseless) ideas. To Josep Vidal, also, for offering me a wide variety of ideas for the project in the first place, and for giving me access to his server, without which the realization of this project would have been much more difficult.

My sincere thanks also go to Anna Lladó for her charming support and her patient explanations on graph theory, and to Sandra Lagén, for sharing with me her approach on using graph-coloring theory to model our network restrictions.

Last but not the least, I would also like to thank Daniel Camps and August Betzler, from the i2CAT foundation, for their feedback on our theoretical developments, from their expertise in real network deployments.

REVISION HISTORY AND APPROVAL RECORD

Revision	Date	Purpose
0	06/04/2016	Document creation
1	07/04/2016 – 25/05/2016	Document writing
2	27/05/2016	Document revision
3	03/06/2016	Document revision
4	04/06/2016 – 12/06/2016	Document writing
5	13/06/2016	Document revision
6	14/06/2016 – 25/06/2016	Document writing
7	27/06/2016	Document final revision and closing

Document distribution list:

Name	e-mail
Dídac Surís Coll-Vinent	didac.suris@alu-etsetb.upc.edu
Josep Vidal Manzano	josep.vidal@upc.edu
Adrián Agustín de Dios	adrian.agustin@upc.edu

Written by:		Reviewed and approved by:	
Date	27/06/2016	Date	27/06/2016
Name	Dídac Surís	Name	Josep Vidal
Position	Project Author	Position	Project Supervisor

CONTENT

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Objectives and problem formulation

1.1.1 Statement of purpose and motivation

This project addresses the challenge of optimizing a wireless network backhaul, an important issue in mobile communications because the trend of future networks is to generate dense deployments of base stations (BS) to improve coverage and per-area capacity, which implies, in turn, the need of a reliable, larger capacity backhaul. The costs associated with it can be significantly reduced by using a wireless network instead of a wired one. Moreover, the vast majority of the current backhaul networks are designed as static, meaning that the resources given to each node (or BS) and each link are fixed, and do not depend on the momentary network requirements; as opposed to this strategy, we propose to optimize the efficiency of the system dynamically (as opposed to statically) optimizing the resource allocation as a function of the current traffic demand.

To support all that, we introduce a closely related tool: the network virtualization, which enables abstraction and sharing of infrastructure and radio spectrum resources, reducing significantly the overall expenses of wireless network deployment and operation [1]. This work can only be understood in a Software Defined Network (SDN) context, whereby the control and data plane are separated, the network functions are less dependent on specific hardware, and all the network variables are optimized at the same time. If the reader is unfamiliar with the concepts of virtualization or SDN, it is highly recommendable to read the Appendix III. Virtualization and SDN.

Taking into account all the above-mentioned, *the purpose of this project is to optimize the efficiency of a wireless network backhaul through the virtualization of this network, in a dynamic way, and interpreting the efficiency from different points of view.* This will be done using several tools such as convex optimization, graph colouring, interference-aware resource allocation and Multiple Input Multiple Output (MIMO) approaches.

The results this thesis provides can be seen as tools for the network administrators to use depending on the scenario they are considering. As this work covers a wide variety of cases, they can always find the one that suits their case best, or at least adapt the given tools to it. These tools permit knowing which values should be assigned to the different network variables, how and what parameters should be modified in order to improve the network functioning, and what results should be expected from these changes.

1.1.2 Achievements and specifications

Schematizing the previous ideas, the project has to achieve the following goals:

- The project has to include a system definition, contemplating the considered scenarios, and the definition of the key performance indicators (KPI).
- It has to be developed based on the proper theoretical background, and include a thorough literature reviewing.
- It has to demonstrate that there are efficient ways of distributing the radio resources that lead to a substantial improvement of the KPI, using interference management, MIMO and/or graph colouring.
- It has to clearly explain the considered cases, the rationale behind their chose, the problems we want to solve associated with each case and scenario, and the mathematical model associated to them.

- It has to include a solution to each of the previous problems, justifying the validity of that solution using different mathematical tools such as convex optimization, communication theory and graph theory.
- The project will include a Matlab simulation of each of the cases, which will evaluate the KPI and will provide all the necessary parameters in order to evaluate the correctness of the solution, as the parameters provided by the simulation have to make sense altogether.
- It has to include a study on the interpretation of the results, providing guidelines and lessons learnt in terms of the most restricting conditions are, and how to improve the functioning of the network.
- A mathematical study of the problem decomposition in subproblems.

1.2 Document structure & State of the art

In this section, we introduce the remaining chapters and the state of the art for each part of the thesis.

In chapter 2 System model and mathematical formulation, the considered scenarios, based on [2] and [3], will be introduced, and the model of the system and the mathematical structure of the problem is presented. The paper from which this project departs is [4], and from its results and procedures, we build the rest of the project, so the next chapter presents this paper's content, along with other model considerations, obtained from [5] and [6]. The convex optimization theory is explained in [7].

In chapter 3, the interferences are introduced, together with the explanations and resolutions of different cases involving them. First, an interference avoidance case is presented, where the graph theory (see, e.g., [8]) takes relevance. Then for the interference management in a SISO case we use the approach given in [9], and for the MIMO case, our main reference is [10].

After that, some decomposition methods for our problem are presented in chapter 0. The mathematical background for the decomposition of a convex problem has been obtained mainly from [11]. In this chapter, the virtualization of the resources plays an important role. A good reference is [1].

Chapter 5, Constraints analysis and interpretation, develops a study of the sensitivity and infeasibility of the problem as a function of the constraints. For this study, we used ideas from [12] and [13].

The readers are recommended to go through the appendices for more information on graph theory and convex optimization, if some of these fields are unknown to them. Apart from that, the thesis is self-contained.

1.2.1 Key Performance Indicators

When comparing results from different optimization methods, scenarios or cases, different KPI can be considered: the minimum use of bandwidth, the minimization of the total power, the delay between the sender and the receiver, the resource utilization, or any utility function of the traffic or any other parameter. These are, at the same time, possible *objective functions* we can optimize (any convex function can be used depending on the purpose of the user), as will be explained in 2.5.

This relation "KPI – objective functions" means that when comparing results in this thesis, the KPI we will consider is simply the value of the objective function. This can only be done when all the restrictions to the compared problems are the same or equivalent (this is, the comparison is fair), and obviously the different results are obtained for the same objective function.

2 SYSTEM MODEL AND MATHEMATICAL FORMULATION

2.1 Problem formulation and considered scenarios

In order to start from a specific and clear working basis, we defined two different user scenarios, which share the same basic characteristics, and are based in [3]. We consider an urban communications network, consisting of several Small Cells that provide access to the subscribers (mobile devices), and which in an urban context, can be mounted on lampposts or street furniture [2]. The access network is the one that connects the subscribers to their immediate service provider[1], and the connection between the BS of these Small Cells and the Core Network is the Backhaul Network[2]. This is the one we based our project on: we consider a wireless mesh network, composed by several nodes. All of them are routers in the Backhaul Network, and can also serve as entry (and exit) points for access traffic, or as a connection to the wired Core Network.

Please note that some nodes have the possibility of accepting some traffic coming from the outside of the network (like uplink access traffic generated in base stations), and also of delivering this traffic to the outside of the network (like downlink access traffic designated to base stations). This traffic will have a source and a destination, and between them, several different paths may be possible, depending on the (wireless) connections between the routers, so the objective is to choose the most appropriate one. For the purpose of this project, both voice and data traffic can be considered.

2.1.1 Scenario 1: Mobile network operator running the backhaul + radio access network

In this scenario, we consider a Mobile Network Operator (MNO) that controls both the backhaul and the access networks, including its traffic and resources (namely, transmitted power of each of the antennas, time-slot fractions and bandwidth allocation). This means that the MNO will be able to choose which the most important traffic is, and where to assign more resources.

2.1.2 Scenario 2: Infrastructure provider renting backhaul to mobile network operators

In this second scenario, instead of having a single MNO that controls all the network parameters, the presence of several MNOs is considered. Each MNO controls its access network, and therefore, the traffic it introduces to the backhaul network, which is controlled by an Infrastructure Provider (InP). The InP has control over all the backhaul network, including the routing of the traffic and the power and bandwidth allocation.

Following a network virtualization approach, from the MNOs point of view, the total capacity of each link, the total power transmitted from the antennas or the traffic from the other MNOs should be simply invisible. The different MNOs are transparent to each other: they only see their part of the network, consisting of their part of traffic and the capacity they can use: they only see a *virtual network*, the one the InP allows them to see.

When considering this scenario, we will derive the techniques required by the Infrastructure Provider to allocate backhaul resources to MNO tenants. In these relationships between InPs and MNOs there is always a Service Level Agreement (SLA), which is a manifesto stating the conditions and the rights of each part, such as the scope, quality or responsibilities associated to the service. Typical values taken into account are the minimum traffic per MNO the InP has to provide, the maximum peak traffic an MNO can transmit and its duration, and the recovery time taken after a service outage.

[1] https://en.wikipedia.org/wiki/Access_network
[2] https://en.wikipedia.org/wiki/Backhaul_(telecommunications)

2.2 Network flow model

Our system model, which we have taken from (and is perfectly explained in) [4], can be divided in two separated parts. The first one is the network flow model, and the second one is the communications model.

The network flow model is based on a directed graph (which we assume is always connected), where each edge represents a link ($l = 1, \dots, L$), and each vertex, a node ($n = 1, \dots, N$). A link is an ordered pair (i, j) of distinct nodes, and its presence means that the network is able to send data from the start node i to the end node j. The network topology is represented by the *node-link incidence matrix* $\boldsymbol{A} \in \Re^{N \times L}$, whose entry A_{nl} is associated with the node n and link l via

$$A_{nl} = \begin{cases} 1 & \text{if } n \text{ is the start node of link l} \\ -1 & \text{if } n \text{ is the end node of link l} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

We define $\mathfrak{O}(n)$ as the set of links that are outgoing from node n, and $\mathfrak{I}(n)$ as the set of links that are incoming to the node n.

In our model, each node can send (different) data to many destinations and receive data from many sources. The data flows are identified by their destinations, so the data in a link is treated separately depending on its destination, being its source irrelevant, as all the flows going to the same destination node are treated as one. We assume that the destination flows are labeled $d = 1, \dots, D$, where $D \leq N$. For each destination d, we define a *source-sink vector* $s^{(d)} \in \Re^N$, whose nth entry ($n \neq d$) $s_n^{(d)}$ denotes the nonnegative amount of flow (data rate in bits/s) injected into the network at node n (the source) and destined to node d (the sink). In order to accomplish the flow conservation law, the sink flow at the destination is given by $s_d^{(d)} = -\sum_{n, n \neq d} s_n^{(d)}$, which is in fact the (negative of the) total flow destined to d.

On each link l, we let $x_l^{(d)} \geq 0$ be the amount of flow destined for node d. We call $x^{(d)} \in \Re^L$ the *flow vector* for destination d. At each node n, components of the flow vector and the source-sink vector with the same destination satisfy the flow conservation law:

$$\sum_{l \in \mathfrak{O}(n)} x_l^{(d)} - \sum_{l \in \mathfrak{I}(n)} x_l^{(d)} = s_n^{(d)} \rightarrow \left\{ \begin{matrix} \text{compactly} \\ \text{written} \end{matrix} \right\} \rightarrow \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)}, \quad d = 1, \dots D \tag{2.2}$$

where $\boldsymbol{A}$ is the node-link incidence matrix previously defined. Finally, we impose capacity constraints on the individual links. Let c_l be the capacity of link l and $t_l = \sum_d x_l^{(d)}$ be the total amount of traffic on link l. We then require that $t_l \leq c_l$.

In summary, our network flow model imposes the following group of constraints on the network flow variables $x^{(d)}, s^{(d)}$ and t:

$$\begin{aligned} &\boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)}, \quad \boldsymbol{x}^{(d)} \succcurlyeq 0, \boldsymbol{s}^{(d)} \succcurlyeq_d 0 \quad d = 1, \dots D \\ &t_l = \sum_d x_l^{(d)}, \quad l = 1, \dots, L \\ &t_l \leq c_l. \quad l = 1, \dots, L \end{aligned} \tag{2.3}$$

where $\succcurlyeq$ means component-wise inequality, and $\succcurlyeq_d$ means component-wise inequality except for the dth component (the sink flow $s_d^{(d)}$ is always negative). We will use x to denote the collection of flow vectors $\boldsymbol{x}^{(d)}$ and use s to denote the collection of source vectors $\boldsymbol{s}^{(d)}$.

This model describes the average behavior of data transmissions, i.e., the average data rates on the communication links, and ignores packet-level details of transmission protocols and forwarding mechanisms. The link capacity in practical communication systems should be defined appropriately, taking into account packet loss and retransmission, so the flow conservation law holds for the effective throughput or goodput.

2.3 Communications model

We define as *communication variables* the critical parameters on which the capacities of individual links depend, such as the transmit powers, bandwidths, or time-slot fractions, and denote the vector of communication variables by $\boldsymbol{r}$. Let r_l be a vector of communications variables associated with link l. In general, the capacity c_l depends not only on r_l, but also on communications resources allocated to other links in the network (due to interferences). To begin with, we will consider the case where the link capacity is only a function of the local resource allocation r_l:$c_l = \Phi_l(r_l)$. The functions Φ_l are concave and monotone increasing in r_l. The concavity of Φ_l implies that the first set of constraints are jointly convex in and $\boldsymbol{t}$ and $\boldsymbol{r}$. The monotonicity condition means that the link capacities increase with increasing resources.

The communications variables are themselves limited by various resource constraints, such as limits in the total transmit power at each node. We model these restrictions with the generic constraint $\boldsymbol{Fr} \preccurlyeq \boldsymbol{g}$. For example, if we want to limit the maximum transmit powers per node, we can considerate $\boldsymbol{r}$ as the vector of powers, and use $\boldsymbol{A}_+\boldsymbol{r} \preccurlyeq P_{\max}$, where $(\boldsymbol{A}_+)_{nl} = \max\{0, A_{nl}\}$, only identifying the outgoing links at each node.

The vector $\boldsymbol{r}$ includes the power ($\boldsymbol{p}$), bandwidth ($\boldsymbol{w}$) and time ($\boldsymbol{time}$) resources, $\boldsymbol{r}$ is just a general notation. Depending on the case, we will substitute $\boldsymbol{r}$ for any of these, or for two of them at the same time. The constraint $\boldsymbol{r} \succcurlyeq 0$ will always apply, and the constraint $\boldsymbol{Fr} \preccurlyeq \boldsymbol{g}$ will be modified depending on the meaning of $\boldsymbol{r}$.

2.4 Mathematical formulation

Combining the two previous models, a mathematical model for the wireless data network can be obtained. This model reflects how the link capacities depend on the allocation of communications resources, and how the overall optimal performance of the network can only be achieved by simultaneous optimization of routing and resource allocation (SRRA). Suppose that the objective is to minimize a convex cost function $f(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{r})$. We have the following generic formulation of the SRRA problem:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{r}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)}, \ \boldsymbol{x}^{(d)} \succcurlyeq 0, \quad \boldsymbol{s}^{(d)} \succcurlyeq_d 0, \quad d = 1, \dots, D \\ & t_l = \sum_d x_l^{(d)} \ , l = 1, \dots, L \\ & t_l \leq \ \Phi_l(r_l) \ , \ \ l = 1, \dots, L \\ & \boldsymbol{Fr} \preccurlyeq \boldsymbol{g}, \quad \boldsymbol{r} \succcurlyeq 0 \end{aligned} \tag{2.4}$$

Here, the optimization variables are the network flow variables s, x, t and the communications variables r. Since the constraints in (2.4) define a convex set and the objective function is convex, the SRRA problem is a convex optimization problem [3].

Vectors will be written in lower case, and matrices will be written in upper case, both in bold font. Scalars will be painted without bold. All variables are real. When using " $\log x$ " we will be referring to the binary logarithm " $\log_2 x$ ".

Variable	Description
$\boldsymbol{r}$	Resource vector. Each component represents the resource of a link.
$\boldsymbol{p}$	Power vector. Each component represents the transmitted power of a link.
$\boldsymbol{w}$	Bandwidth vector. Each component represents the bandwidth of a link.
$\boldsymbol{time}$	Time vector. Each component represents the time slot of a link.
h, H	Channel, understood as the *complex* constant gain between the transmitter and receiver.
k	Constant including channel losses and noise
$\boldsymbol{g}$	Resource limit. Vector representing the resource limits in each node. Each element represents a node.

Variable	Description
σ	Noise power
D	Number of destinations
N	Number of nodes
L	Number of links
$\Delta(G)$	Degree of the graph G
$\mu(G)$	Multiplicity of a graph G
$\chi'(G)$	Chromatic index of a graph G
$\boldsymbol{u}$	Receiver filter
$\boldsymbol{v}$	Precoding vector
$\boldsymbol{c}$	Capacity of a link, in bits/s
t_l	Total traffic in a link l
x_l	Traffic per user per link l
$s_n^{(d)}$	Traffic from a node n to a destination d

Table 2.1 Variable definitions

2.5 Objective functions

Different objective functions can be considered, depending on the necessities of the user or, in our case, the interests of the Infrastructure Provider. They are related to the KPI we want to improve: for example, if we want to study how we can reduce the transmitted power, compared to the non-optimized network, the objective function will be the total transmitted power. Some typical examples are:

- Minimize the total used power: $f(\boldsymbol{p}) = \sum_l p_l$
- Minimize the total bandwidth: $f(\boldsymbol{w}) = \sum_l w_l$
- Maximize a certain utility function of the traffic, if it is not fixed:

$$f(\boldsymbol{s}) = \sum_d \sum_{n,n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) = \sum_d \sum_{n,n\neq d} \log\left(s_n^{(d)}\right) \tag{2.5}$$

For all the following simulations, the previous utility function (2.5) *will be used, unless otherwise specified.*

One may impose that certain destinations are more important than others in terms of priorities, by adding a weighting parameter multiplying, for example $\sum_d \omega_d \cdot f(s^{(d)})$. The only restriction for these functions is to be convex (or concave if we want to maximize them) with respect to the optimizing variables. However, working with this restriction is not always so straightforward. Below, a couple of other useful examples are presented, and a convex solution is given to each one.

2.5.1 Delay minimization across multiple MNO

To begin with, *the destination nodes can be interpreted as the different MNOs* in the Scenario 2. This way, the traffic destined to a certain node is equivalent to the downlink traffic an operator wants to get to its BS.

A similar interpretation with the uplink traffic can be done by slightly modifying the system model, and instead of separating the traffic by its destinations, separate it by its source (or "entry"): we would have an entry e $(1, \dots, E)$ with an associated traffic in each link, and $\boldsymbol{x}^{(e)}$ would be the traffic "coming from e" instead of "going to d". The vector $\boldsymbol{s}^{(e)}$would be the traffic coming from the node e, and each component $s_n^{(e)}$, the traffic coming from e and destined to n. The only modification in the mathematical model would be that instead of having the restriction $\boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)}$, we would have $\boldsymbol{A}\boldsymbol{x}^{(e)} = -\boldsymbol{s}^{(e)}$.

A way of using this interpretation is to include in the objective function the minimum delay in the transmission of a message. As in our formulation there is no such thing as messages or packets, rather they are treated as a general and uncountable, but obviously quantifiable, concept of traffic, a common cost function used in the communication network literature is the *total delay function* [14]:

$$f(t) = \sum_l \frac{t_l}{c_l - t_l} \tag{2.6}$$

This function represents the delay of the link (which is closely related to the occupation (t/c) of the link, as the delay increases with the channel occupation) and it is a sum for all the links in order to consider all them (a weighed sum can also be used). It is convex with respect to *t,* but not jointly convex in *c* and *t*, so we prefer to use another cost function with similar qualitative properties, which is the *maximum link utilization* [14]:

$$f(t) = \max_l \frac{t_l}{c_l} \tag{2.7}$$

This function is quasi-convex [4], and therefore can be solved through a series of convex feasible problems ([7], section 4.2).

We can adapt our problem with a slight reinterpretation of the parameters: instead of assigning traffic to each MNO (destination) in each link, so that the sum of traffics is less or equal to the capacity of the link, we will assign a capacity to each MNO, so that the sum of capacities is less or equal to the total capacity of the link. This way, x is defined as the maximum traffic an MNO is allowed to receive. Then, we fix the (minimum) traffic each MNO (destination) is actually going to receive (s_{min}), and the network problem can be written as:

$$\text{minimize} \quad \sum_d \omega_d \cdot \max_l \frac{s_l^{(d)}}{x_l^{(d)}} \tag{2.8}$$

$$\begin{aligned} \text{subject to} \quad & s^{(d)} \succcurlyeq {s_{\min}}^{(d)}, \ \forall d \\ & Ax^{(d)} \succcurlyeq s^{(d)}, x^{(d)} \succcurlyeq 0, s^{(d)} \succcurlyeq_d 0, \forall d \\ & t_l = \sum_d x_l^{(d)} \ , t_l \leq \Phi_l(r), \qquad \forall l \end{aligned}$$

where ω_d is a weight associated to each destination (or MNO) depending on the importance we want to give them (the larger the weight, the lower the delay that MNO will bear in its transmissions). The main difference with respect to the original problem is the inequality $\succcurlyeq s$. When equality holds, the delay associated to that link and destination tends to be infinite in (2.6) (in the original problem (2.4) we consider that the traffic adds up to the capacity, and do not consider the occupation of the channel).

2.5.2 Reaching isolation between MNOs to accomplish with the SLA

In network virtualization, a very important requirement is the *isolation* between resources [1]. In our case, we do not want any MNO to notice that other MNOs operate in the same network, which means that they will always have the right to successfully transmit the minimum traffic stablished in the SLA (if SLA is defined in terms of minimum traffic), regardless of the other MNOs traffic. However, if one MNO wants to transmit more traffic in a specific moment, it could be allowed as long as its transmissions do not deprive other operators of their minimum traffic (and with a certain limit, depending on the SLA).

A way of dealing with it is using a *penalization* in the objective function, instead of a restriction. If a restriction is applied, the solution of our problem must fit the restriction or otherwise the problem is said *infeasible*. On the contrary, a penalization is just a term in the objective function that impedes the problem to return certain

unwanted results because they are penalized. However, if these results are the only option, the problem provides them as a solution that is not infeasible, as these results are not strictly forbidden, just undesired. As a way of example, the formulation could be like this:

$$\text{minimize} \quad f(\text{variables}) + \sum_{d} \sum_{n,n\neq d} k_n^{(d)} \left| s_{n\,\text{demanded}}^{(d)} - s_n^{(d)} \right|^{\alpha} \tag{2.9}$$

where $\alpha \geq 1$, the greater its value, the more it will penalize high differences, but still allow low differences. The $k_n^{(d)}$ term is just a calibration constant. A restriction will ensure that the traffic provided is always at least the minimum that the SLA states, such as $\sum_{n\neq d} s_n^{(d)} \geq s_{\text{minimum}}^{(d)}, \quad \forall d$. Note that in this formulation the traffic served may be lower than the traffic demanded if the link capacities are not enough.

2.6 Realistic case

In order to depart from a realistic case and simulate networks with practical meaning, the following prototype network was created, from the feedback from the i2CAT staff (see Appendix IV. Work plan and incidences).

We consider the IEEE 802.11ad protocol, promoted by the Wireless Gigabit Alliance (WiGig)[3], which works with wireless communications operating over the 60 GHz frequency band. A network of 10 nodes, with an approximate separation of 50 m between them, is considered. The maximum transmitting powers are 10 dBm [5], and there are four 2 GHz channels. The typical considered traffics will be of 1000 Mbit/s per link. The losses at these frequencies are very high, and following the free space path loss formula, $L = (4\pi df/c)^2$, for a 50 m distance, $L(dB) = 102\ dB$. Considering a noise power of $-174 + 10\log(W)\ dBm$ ([6, pp. 31, 32]), a $W = 2\ GHz$ bandwidth and antenna gains of 15 dBi at transmitter and receiver, the signal to noise ratio is:

$$SNR(dB) = 10\ dBm + 2 \cdot 15\ dBi - 102\ dB - (-81\ dBm) = 19\ dB.$$

For the network topology, we consider a network where the antennas have a radiation pattern spanning between 90º and 180º (in our example, we consider 180º), and they can be electronically steered in order to point to a node or another. We consider that there is a link between two antennas when they can potentially connect by changing their directivities and the nodes are at a maximum distance of 150m, and a single antenna can only transmit or receive from one link at a time. When facing a FDMA approach, we will modify this topology and will consider that each node has fixed several antennas, each one being the receiver or transmitter of a link. In the example in Figure 2.1, not all the possible links have been created (to eliminate the symmetry), the central node has a 360º vision angle (for example has 2 antennas), and the orange nodes are the destinations. *For all the following simulations, this network will be used unless otherwise specified.*

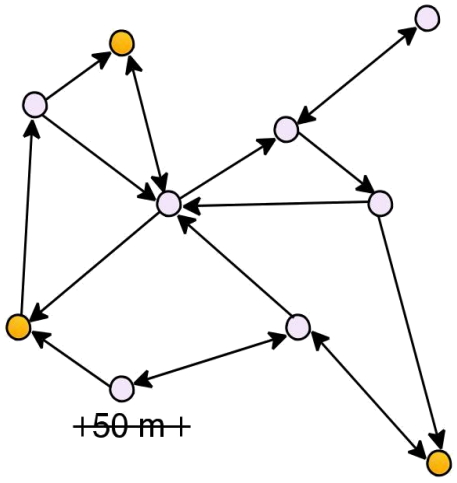

Figure 2.1 Example network

Despite this being a realistic case, our work is not limited only to this example, as we pretend to optimize parameters that are not currently optimized in general, such as the transmitting power of the antennas. This is why in our simulations we will respect as much as possible the previous realistic case, but will modify it when it limits us. For this project, we assume that we have knowledge of the wireless channels at the transmitter side, and that we can modify all the optimized parameters.

[3] http://www.wi-fi.org/?utm_source=wigig&utm_medium=referral&utm_campaign=wigig-redirect

3 JOINT ROUTING AND RESOURCE ALLOCATION IN WIRELESS INTERFERENCE NETWORKS

In this section, the main optimization problem (2.4) will be reformulated in order to introduce interference in our network model, with the objective of spatially reusing resources, and improving the spectral efficiency. Several approaches will be taken depending on the considered models, as there are many channel access methods and we would like to include into our solution a representative majority of them. The reference case that we will improve is the one where each link has a different time/frequency resource, with no reuse.

3.1 SISO Case. Interference avoidance based on graph coloring

In this section, we consider a model whereby all interference among links is avoided, as in TDMA or FDMA; we just need to ensure that all the resources are orthogonal between themselves. This section introduces the optimization of the frequency (or time) resource, together with the transmitted power.

3.1.1 FDMA model

A departing approach is to consider that all the receivers in our network are receiving signal from all the transmitters, so that when each transmitter sends a signal, it reaches each receiver (only one of them being the desired receiver). A trivial way to avoid interference is that all resources are orthogonally split among transmitters, for which a possible solution would be equally distributing the resources among the links. Since we are optimizing the resources, however, this is not the most efficient solution, and we will look for new ones. We will consider the FDMA approach, but it can be done similarly with the TDMA, as we will see.

First, we need to reformulate the problem. Now the capacity formula depends both on the bandwidth associated with each link ($\boldsymbol{w}$), and on the power $\boldsymbol{p}$, like:

$$\Phi_l(\boldsymbol{r}) \equiv \Phi_l(p_l, w_l) = w_l \cdot \log\left(1 + k_l \frac{p_l}{w_l}\right) \tag{3.1}$$

where the constant k_l takes into account the channel losses and the noise spectral density. Note that, as we are avoiding interference, the capacity of a link only depends on the resources allocated to that link. We will drop this assumption later in sections 3.2 and 3.3. Then, the only extra condition needed to reach the resources orthogonality would be:

$$\sum_{l=1}^{L} w_l \leq W_{total} \tag{3.2}$$

where W_{total} is all the bandwidth we can use. This constraint would be the specific formulation for the previous general "$\boldsymbol{Fr} \preccurlyeq \boldsymbol{g}$", and is a linear restriction, so the problem remains convex (see Appendix II. Mathematical background). Here we are considering a fixed total bandwidth we can use, but we could formulate the problem similarly was the objective function to minimize W_{total}; then it would be a variable.

Obviously, this solution does not allow making an efficient use of the spectrum, as it is considering that all the nodes can interfere with each other, which is not always the case in practice. Hence, we now consider a little more sophisticated model that will include the following: if two nodes are neighbors, they cannot share the same resource, because they would be interfering with each other. It is easy to see that if both the node A and the node B want to transmit to the node C, in C we will have the signals both from A and from B, so they

should use different bands (orthogonal resources). And in the opposite case, if B transmits both to A and C, we also consider that B cannot use the same band to transmit the two signals, as they would also interfere with each other. We consider that in the rest of the nodes the interference is negligible.

Hence, the new conditions we need to add to the problem are:

- All the links entering or leaving a node have to have different resources associated (with more than one possible resource for each link, even zero).

$$\sum_{l \in \mathfrak{O}(n), \mathfrak{I}(n)} w_l \leq W_{total} \rightarrow \text{necessary, not sufficient} \qquad (3.3)$$

- The links need the same resource at their two ends (the transmitter and the receiver).

The idea is to introduce certain restrictions (that would also be specific cases for the restriction "$\boldsymbol{Fr \preccurlyeq g}$" in (2.4)) into the optimization problem that ensure that the solution provided by the solver (see Appendix I. System Level Simulator) is implementable. This means that there is a way to distribute the resources (in this case the bandwidths) to each link so that they accomplish the requirements above.

To solve this problem we will introduce the graph theory. A short introductory explanation can be found in the Appendix II. Mathematical background. First of all we start from a graph G_1, where each BS is a node, and each link is an edge. We will use the example in Figure 3.1.

For the purposes of our problem, we can ignore the arrows (directions of the link) for a moment, as both the input and output links are considered as interference, so we would have a simple graph.

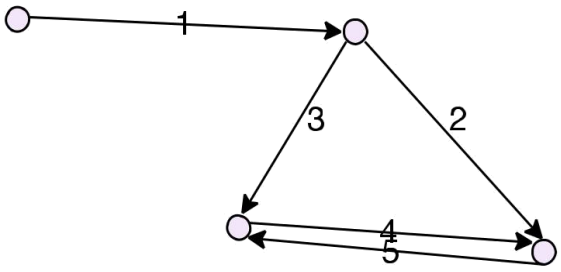

Figure 3.1 Example of a graph

We will model the different w_l as the *number of colors* associated with each link. To make it simple, we can think of W_{total} as being an integer representing the number of available frequency channels, and $w_l$an integer representing the number of frequency channels we can use in link l. A *color* would be the specific channel.

Then, we can create a new graph, which we will call *extended graph G*, from the first one, where each edge l (representing the link l) will be replicated w_l times, and hence will be transformed in a set of w_l edges, creating a multigraph. Imagine in the example that links 2 and 4 are assigned only one channel (or color), link 3 and 5 two channels, and the link 1, three channels, then the extended graph will result in the one in Figure 3.2.

We can now translate the previous idea to the graph theory language: the solution we obtain from the solver has to allow the extended graph to be ***edge-colorable***, as we do not want two links associated to the same node to share the same frequency channel (or color). It is important to remark that the optimization problem does not provide the specific colors or channels; it just ensures that we will be able to find them using a proper coloring algorithm.

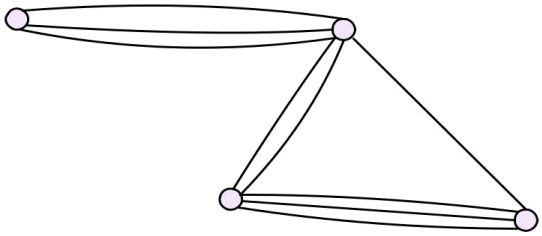
Figure 3.2 The extended graph from Figure 3.1

Now we need to come up with the restrictions we have to introduce to the problem for the extended graph to be edge-colorable. We will use the *chromatic index* concept for the extended graph. Note that is the extended

graph the one that needs to be colorable, so we have to ensure that $W_{total} \geq \chi'(G)$, as W_{total} is the total number of colors.

The problem is that finding the chromatic index of a graph is an NP-Problem, and thus very hard to solve, and even more complicated when we actually do not know how *G* (the extended graph) is. Because of that, we will work with several upper bounds, which are stated below, and can be found in [8]:

- (Shannon's Bound). Every colorable graph G satisfies:
$$\chi'(G) \leq \left\lfloor \frac{3}{2}\Delta(G) \right\rfloor \tag{3.4}$$
- (Vizing's Bound). Every colorable graph G satisfies:
$$\chi'(G) \leq \Delta(G) + \mu(G) \tag{3.5}$$
- (Favrholdt, Stiebitz and Toft 2006). Every colorable graph G satisfies:
$$\chi'(G) \leq \Delta(G) + \mu(G - v) \tag{3.6}$$
for every vertex $v \in V(G)$

The expression $\mu(G - v)$ means that we take into account the multiplicity of the graph $G - v$, which is the graph G without a node, whichever (the one we prefer), and also without all the edges incident to that node.

The tightness of these bounds depends on the particular graph, and we did not encounter a way of determining the best bound depending on the graph type. However, our simulations (in Figure 3.3) show that in general the Shannon's bound (3.4) is tighter than Vizing's (3.5). Mathematically, it only can be assured that the bound (3.6) is tighter than (3.5), as it is its extended version, and $\mu(G) \geq \mu(G - v)$.

These are theoretical bounds that assure that coloration is possible; they do not necessarily explain how to do it, the algorithm to apply. However, these particular bounds are demonstrated through the algorithm that makes them possible, so applying that algorithm we will always be possible to color the graph with the number of colors indicated by the bound [15].

Using the Shannon's Bound (3.4) we can transform the restriction to:

$$W_{total} \geq \left\lfloor \frac{3}{2}\Delta(G) \right\rfloor \geq \chi'(G) \tag{3.7}$$

We can use a simpler upper bound, considering that in the cases where $\Delta(G)$ is odd we do not floor it:

$$W_{total} \geq \frac{3}{2}\Delta(G) \tag{3.8}$$

Now we transform $\Delta(G)$ into the parameters we actually are working with, and arrange the inequation:

$$\sum_{l \in \mathfrak{O}(n), \mathfrak{I}(n)} w_l \leq \frac{2}{3} W_{total} \; \forall n \; \in V(G_1) \tag{3.9}$$

where $\sum_{l \in \mathfrak{O}(n), \mathfrak{I}(n)} w_l$ is $\delta(n)$ and $V(G_1)$ are all the nodes of the initial graph. Using a compact notation, the expression on the left is simply $|\boldsymbol{A}| \cdot \boldsymbol{w}$.

Similarly, we can use the Vizing's Bound (3.5) to obtain:

$$\begin{gathered} \sum_{l \in \mathfrak{O}(n), \mathfrak{I}(n)} w_l + M \leq W_{total} \quad \forall n \in V(G_1) \\ w_l \leq M \; \forall l \in E(G_1) \end{gathered} \tag{3.10}$$

where M is a represents the multiplicity of the extended graph, which is unknown. The definition of M as "multiplicity" is given in the second condition in (3.10), which defines that M is at least the maximum of the bandwidth resources in a link. If for a pair of nodes A and B there are two links, one going in each direction, instead of w_l we have to write the $w_1 + w_2$, where 1 and 2 are the links from A to B and B to A respectively.

Finally, the third option (3.6) would be:

$$\sum_{l \in \mathfrak{O}(n), \mathfrak{I}(n)} w_l + M \leq W_{total} \quad \forall n \in V(G_1) \qquad w_l \leq M \; \forall l \in E(G_1 - v) \tag{3.11}$$

This is, when applying the multiplicity definition we would do it over the initial graph with a node being subtracted (the one that we prefer), and obviously also its related edges.

The previous restrictions (3.9), (3.10) or (3.11) would be added to the initial problem (2.4).

A comparison of the impact of using one or another bound on the optimization problem can be seen in the following Figure 3.3. The simulation consists in using the previously defined network, and creating links depending on the *maximum link length* (all the pairs of nodes within that distance are connected with probability 0.5, and for each maximum link length, the simulation represents the mean of 25 simulation results). The increase of the maximum link length means there are more links and thus the graph is denser.

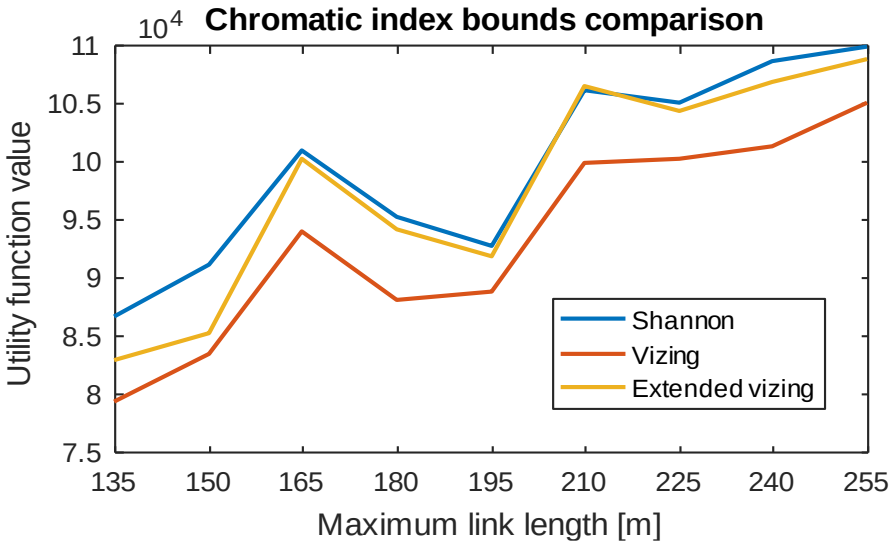

Figure 3.3 Comparison between chromatic index bounds

It can be appreciated that the improvement of the Shannon bound over the Vizing bound does not depend on the density of the graph. In addition, the option from (3.6) and (3.11), that we called "extended Vizing", always gives better or equal results than the Vizing one, as it has fewer restrictions. Intuitively, the more the number of nodes, the less the difference between these two methods will be, because a single node will not be so important. In these simulations, the deleted node has been chosen randomly, as we could not find a proper way to introduce its optimization in our convex model.

A comparison between optimizing (with the Shannon method) and non-optimizing the bandwidth, showing the improvement of the former respect to the latter, can be seen in the Figure 3.4. In the non-optimizing case, the total bandwidth is uniformly distributed among all the links in the network. The objective function in this case is the utility function of the traffic (2.5).

3.1.2 TDMA model

A similar reasoning can be done for the TDMA model, although now we slightly modify the network topology as explained in 2.1. In any case, the mathematical network model is the same as before. The main difference between the TDMA case and the FDMA case is that now the power cannot be optimized, it cannot be divided in the different subbands. A node can always send less power than its maximum, but the power that does not go to a link cannot go to some other link.

Before we implicitly considered the capacity was being multiplied by a unit time. Now we consider that every transmitter is using all the possible bandwidth, to be fair with the comparisons. The model only changes in its communication model, where we would have (for the Shannon case; the others would be equivalent):

$$t_l \leq \Phi_l(time_l) = time_l \cdot W_{total} \cdot \log\left(1 + k_l \cdot \frac{p_l}{W_{total}}\right), \quad time_l \geq 0 \ \ \forall l$$
$$\sum_{l \in \mathfrak{D}(n), \mathfrak{I}(n)} time_l \leq \frac{2}{3} \cdot (1s) \quad \forall n \in V(G_1) \tag{3.12}$$

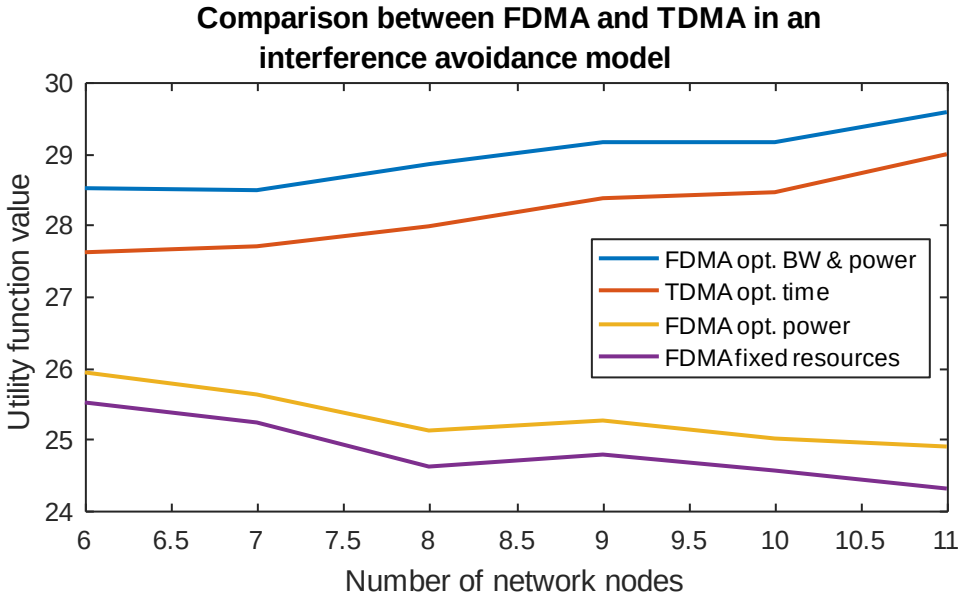

Figure 3.4 TDMA vs FDMA in randomly generated networks with increasing number of nodes (for the same area) and two random destinations. Mean for 5 independent realizations.

The blue curve represents the FDMA optimization, where both the bandwidth and the power are optimized. The TDMA case is painted in red, and comparing it with the FDMA method, the FDMA has better results because, as said before, the power per band can also be optimized.

After that, two more cases are represented. In yellow, the FDMA case where the bandwidth is equally distributed among all the links, and only the power resource is optimized, and in purple, the case where only the routing is optimized and all the resources are fixed (both power and bandwidth, equally distributed among links). As we could expect, the more resources we optimize, the better the results.

Note that it is representing the sum of the base 2 logarithm of the traffics: the difference between cases is actually very significant. The number of destinations D remains fixed ($D = 2$), so the utility objective function adds the same number of elements. The more the number of nodes, the more the paths: if we are optimizing them, it is like giving more options, so the objective value is greater; if the resources are equally distributed, there are less resources for each link, so the objective value is lower, decreases with the number of nodes.

3.2 SISO Case. Interference management

A new situation arises when, instead of trying to avoid interference, our model contemplates its management, meaning that some interferences can be accepted because they are treated as noise by the un-intended users.

The model formulation only varies in the definition of the capacity function. In this case, the resources are just the power $\boldsymbol{p}$, and we *assume unit total bandwidth in the theoretical development* (all the links share the same frequencies). Now, instead of having $\Phi_l(p_l)$ we have $\Phi_l(\boldsymbol{p})$, which means that the capacity of a link is affected not only by the resources assigned to that link, but also by the other links' resources. Because of this, now the capacity function is *not concave* in function of all the variables p_l:

$$\Phi_l(\boldsymbol{p}) = \log\left(1 + \frac{|h_{ll}|^2 p_l}{\sigma_l + \sum_{j\neq l}|h_{lj}|^2 p_j}\right) \tag{3.13}$$

where σ_l is the noise associated to the link l, and h_{lj} is the channel gain from the transmitter of the link j to the receiver of the link l.

In these cases, two different approaches to the problem can be adopted. The first one is trying to change the model by considering a different $\Phi_l(\boldsymbol{p})$ that is concave. The second one is rewriting the non-concave function so that we obtain a concave one. The two ideas can be put together, as in [9], where interference cancellation is considered, and using that consideration then the formula is modified so the convex problem appears.

We are going to use the second approximation of [9], where they do not consider interference cancellation, but the formula (3.13). Using a high SNR approximation, and considering $\log(1+x) \cong \log(x)$ when $x >> 1$ (say $x \geq 5$ or 10), it can be transformed to:

$$\begin{aligned}\Phi_l(\boldsymbol{p}) = \log\left(1 + \frac{|h_{ll}|^2 p_l}{\sigma_l + \sum_{j\neq l}|h_{lj}|^2 p_j}\right) &\cong \log\left(\frac{|h_{ll}|^2 p_l}{\sigma_l + \sum_{j\neq l}|h_{lj}|^2 p_j}\right) \\ = -\log\left(\frac{\sigma_l + \sum_{j\neq l}|h_{lj}|^2 p_j}{|h_{ll}|^2 p_l}\right) &= -\log\left(\frac{\sigma_l}{|h_{ll}|^2}p_l^{-1} + \sum_{j\neq l}\frac{|h_{lj}|^2}{|h_{ll}|^2}p_j p_l^{-1}\right)\end{aligned} \tag{3.14}$$

This approximation is always an underestimate for the link capacity, so the results obtained will always be implementable. A change of variables can be done:

$$Q_l = \log(p_l) \rightarrow p_l = e^{Q_l} \;\; \forall l \tag{3.15}$$

Then, in order to obtain a *log-sum-exp* expression, we define:

$$\psi_l(Q) = \Phi(e^Q) \cong -\log\left(\frac{\sigma_l}{|h_{ll}|^2}e^{-Q_l} + \sum_{j\neq l}\frac{|h_{lj}|^2}{|h_{ll}|^2}e^{Q_j - Q_l}\right) \tag{3.16}$$

The *log-sum-exp* expression is convex, as can be seen in [7, p. 74], so $\psi_l(Q)$ is concave. The general problem is:

$$\begin{aligned}\text{minimize} \quad & f(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{r}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)}, \quad \boldsymbol{x}^{(d)} \succcurlyeq 0, \quad \boldsymbol{s}^{(d)} \succcurlyeq_d 0 \; d = 1, \dots, D \\ & t_l = \sum_d x_l^{(d)} \;, l = 1, \dots, L \\ & t_l \leq \psi_l(\boldsymbol{Q}) \;, \quad l = 1, \dots, L \\ & \sum_{l\in\mathfrak{D}(n)} e^{Q_l} \leq P_{tot}^{(n)}, \quad n = 1, \dots, N\end{aligned} \tag{3.17}$$

The problem of this formulation is that the approximation is only valid when we have high SNIR for *every link*. A similar approach can be useful for very low SNIR: we can approximate $\log(1+x) \cong x$ (first order Taylor approximation). In that case, we would have:

$$\Phi_l(\boldsymbol{p}) = \log\left(1 + \frac{|h_{ll}|^2 p_l}{\sigma_l + \sum_{j\neq l}|h_{lj}|^2 p_j}\right) \cong \frac{|h_{ll}|^2 p_l}{\sigma_l + \sum_{j\neq l}|h_{lj}|^2 p_j} = \{p_l = e^{Q_l}\} = \frac{|h_{ll}|^2 e^{Q_l}}{\sigma_l + \sum_{j\neq l}|h_{lj}|^2 e^{Q_j}} \tag{3.18}$$

Then, instead of $t_l \leq \phi_l(\boldsymbol{p})$ and $\boldsymbol{Fr} \preccurlyeq \boldsymbol{g}$ we would have:

$$\begin{aligned} &\log(t_l) \leq \log\left(\Phi_l\left(e^{\boldsymbol{Q}}\right)\right), \quad l = 1, \dots, L \\ &\sum_{l\in\mathfrak{D}(n)} e^{Q_l} \leq P_{tot}^{(n)}, \quad n = 1, \dots, N \end{aligned} \tag{3.19}$$

The $\log\left(\Phi_l\left(e^{\boldsymbol{Q}}\right)\right)$ can be developed as before reaching a *–log-sum-exp*, which is concave.

However, the first inequality has the form {concave} ≤ {concave}, so it is *not* a convex problem formulation (it should be {convex} ≤ {concave}). Therefore, to apply this approximation, we have to consider that $\boldsymbol{t}$ is fixed rather than a variable. This can happen if we are given a specific flows routing, and are asked to minimize some function related to the communication resources, such as the total power (for example, if we separated the two subproblems and optimized them alternately, but then we could not assure the result to be optimal).

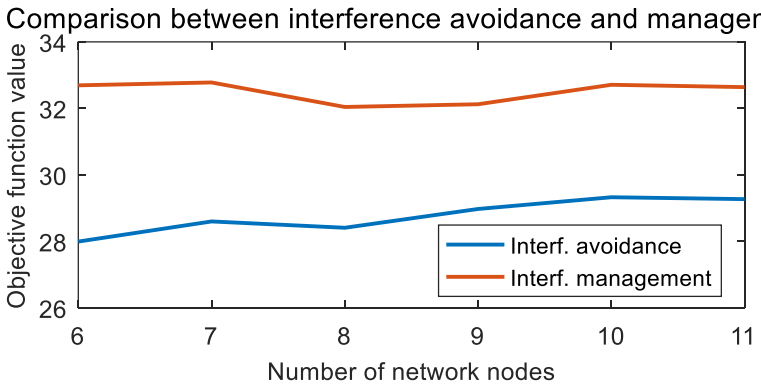

Figure 3.5 High SNIR interference management case compared to the interference avoidance case (Shannon bound)

The comparison between the interference avoidance and management cases depends very strongly on the considered interferences between links. If they are very strong, ($|h_{lj}|^2 \approx |h_{ll}|^2,\ l \neq j$), then the high SNIR assumption is not valid, the logarithm is negative and the problem is infeasible. And when the SNIR is high, and the current approach is valid, it gives much better results than the interference management case. It is difficult for an intermediate case to exist, because the high SNR assumption has to be fulfilled for all the links: if we consider a medium SNIR, as the channels are random, most likely one of them goes from medium to low SNR and makes the problem infeasible.

3.3 MIMO Case. Interference management

In the MIMO case, the previous steps *cannot* be used, as we have more than one variable in the numerator (one for each transmitting antenna), and could not reach the *log-sum-exp* expression. We use the approach explained in [10], where the interference alignment is exploited (it will permit some approximations). In the initial formulation, the *weight-sum-rate* maximization problem is considered. The objective function is $\sum_{k=1}^{K} \omega_k R_k$, where K is the number of users, being a user a pair sender-receiver, R_k is each user's rate, and ω_k a constant related to the priority of each user. A user sender can interfere with another user receiver, and that is the interference that needs to be minimized. The restriction applied is on the transmit power per user.

We can adapt this problem and think that each user in the previous problem is now a link in our problem, this is, instead of users interfering themselves, we have links interfering themselves. Instead of having a power constraint per link (as in the previous problem, where they have a power constraint per user), we would have a power constraint per node (which includes its outgoing links). The main idea is that we need to make the rate function concave in all the variables. The initial rate function is (*assuming single stream MIMO links*):

$$R_l = \log\left(1 + \frac{\left|\boldsymbol{u}_l^H \boldsymbol{H}_{ll} \boldsymbol{v}_l\right|^2}{\sum_{j \neq l} \left|\boldsymbol{u}_l^H \boldsymbol{H}_{lj} \boldsymbol{v}_j\right|^2 + \sigma_l |\boldsymbol{u}_l|^2}\right) \tag{3.20}$$

where $\boldsymbol{v}_l$ are the precoding vectors, of length M_l (number of transmit antennas), $\boldsymbol{u}_l$ the linear receiver filters, of length N_l (number of receive antennas), and $\boldsymbol{H}_{lj} \in \mathbb{C}^{N_l \times M_l}$ the constant complex channel matrix from transmitter j to receiver l. The idea is to use a distributed algorithm to optimize node by node (we cannot do it link by link because the links starting from the same node have a shared power restriction).

The procedure consists in alternatingly optimize the precoding vectors $\boldsymbol{v}$ and the receive filters $\boldsymbol{u}$, departing from a previous interference alignment phase. The interference alignment reduces the *observed* (leakage) interference coming from other users to levels comparable to the noise level, and this permits us doing certain approximations later. Given the precoding vectors, finding the optimal receive filters is just applying a lineal receive filter [10], so in the following steps we will focus on how to find the optimal precoding vectors $\boldsymbol{v}$, given *fixed* receive filters $\boldsymbol{u}$. Some considerations will be taken into account:

1. We will consider high SNR.
2. The links starting from the same node do not interfere with each other.
3. The power coming from one single node is much lower than the sum of the powers coming from the rest of the nodes (plus the noise).
4. We define $\forall l \; h_{lj} \triangleq H_{lj}^T u_l$
5. We define $n(l)$ as the node where the link l starts.

The objective function taken into account is the following. The approximation considers high SNIR (high SNR and successful interference alignment):

$$\sum_{l=1}^{L} \omega_l R_l = \sum_{l=1}^{L} \omega_l \log\left(1 + \frac{\left|\boldsymbol{h}_{ll}^H \boldsymbol{v}_l\right|^2}{\sum_{j \notin n(l)} \left|\boldsymbol{h}_{lj}^H \boldsymbol{v}_j\right|^2 + \sigma_l |\boldsymbol{u}_l|^2}\right) \cong \sum_{l=1}^{L} \omega_l \log\left(\frac{\left|\boldsymbol{h}_{ll}^H \boldsymbol{v}_l\right|^2}{\sum_{j \notin n(l)} \left|\boldsymbol{h}_{lj}^H \boldsymbol{v}_j\right|^2 + \sigma_l |\boldsymbol{u}_l|^2}\right) \tag{3.21}$$

Each node will optimize its links, which means that at each iteration all the nodes will optimize the variables of the objective function (3.21) that they can control, using the values provided by the other nodes. With some manipulation and considering the previous points, the objective function seen by a node n can be transformed in the following way, separating the links starting at n, and the links not starting at n. We need this separation because for each node n, we need to optimize the variables of the links starting at n, not the other links' variables, so the function has to be convex only respect to the variables of the links starting at n, and the approximations done in the two cases will be different.

Part of the objective function of the links starting at n:

$$\sum_{l \in n} \omega_l \log\left(\frac{\left|\boldsymbol{h}_{ll}^H \boldsymbol{v}_l\right|^2}{\sum_{j \notin n} \left|\boldsymbol{h}_{lj}^H \boldsymbol{v}_j\right|^2 + \sigma_l |\boldsymbol{u}_l|^2}\right) = \sum_{l \in n} \omega_l \left(\log\left(\left|\boldsymbol{h}_{ll}^H \boldsymbol{v}_l\right|^2\right) - \log\left(\sum_{j \notin n} \left|\boldsymbol{h}_{lj}^H \boldsymbol{v}_j\right|^2 + \sigma_l |\boldsymbol{u}_l|^2\right)\right) \tag{3.22}$$

which is concave respect to the vectors $\boldsymbol{v}_l$ for $l \in n$.

And for the links not starting at n:

$$\sum_{l\notin n}\omega_l\log\left(\frac{\left|\boldsymbol{h}_{ll}^H\boldsymbol{v}_l\right|^2}{\sum_{j\notin n}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2+\sigma_l|\boldsymbol{u}_l|^2}\right)=\sum_{l\notin n}\omega_l\left(\log\left(\left|\boldsymbol{h}_{ll}^H\boldsymbol{v}_l\right|^2\right)-\log\left(\sum_{j\notin n(l)}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2+\sigma_l|\boldsymbol{u}_l|^2\right)\right)$$
$$=\sum_{l\notin n}\omega_l\left(\log\left(\left|\boldsymbol{h}_{ll}^H\boldsymbol{v}_l\right|^2\right)-\log\left(\sigma_l|\boldsymbol{u}_l|^2+\sum_{\substack{j\notin n(l)\\ j\notin n}}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2+\sum_{j\in n}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2\right)\right) \quad (3.23)$$
$$=\left\{N_l\triangleq\sigma_l|\boldsymbol{u}_l|^2+\sum_{\substack{j\notin n(l)\\ j\notin n}}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2\right\}=\sum_{l\notin n}\omega_l\left(\log\left(\left|\boldsymbol{h}_{ll}^H\boldsymbol{v}_l\right|^2\right)-\log(N_l)-\log\left(1+\frac{\sum_{j\in n}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2}{N_l}\right)\right)$$

And considering that the interferences term $\sum_{j\in n}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2$ is lower than N_l for the previously commented interference alignment, and using the first order Taylor approximation $\log(1+x)\cong x$ for small x, the previous expression can be transformed to:

$$\sum_{l\notin n}\omega_l\left(\log\left(\left|\boldsymbol{h}_{ll}^H\boldsymbol{v}_l\right|^2\right)-\log(N_l)-\frac{\sum_{j\in n}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2}{N_l}\right) \quad (3.24)$$

which is concave respect to the vectors $\boldsymbol{v}_j$ for $j\in n$. Now we have a concave (respect to the variables of the node n) objective function, adding the expressions (3.24) and (3.22).

We can use this in two different ways. The first one is applying it directly, in the communications subproblem (better explained in section 0) where we have the same objective function (the sum-rate maximization).The algorithm is explained in [10], and we use the same procedure, just with a stronger approximation (the consideration number 3). It has to be noted that each node is not seeing exactly the same objective function, as different approximations are applied in each one. The steps would be the following:

Algorithm for weighted sum-rate maximization

1. Initialize precoding vectors $\boldsymbol{v}$ from an interference alignment phase.
2. Repeat until convergence
   a. For each node, calculate its optimal receiver filters $\boldsymbol{u}$ given the current $\boldsymbol{v}$
   b. For each node, optimize $\boldsymbol{v}$ using the explained objective functions.

Table 3.1 Algorithm for weighted sum-rate maximization

The second way we can use it is applying the approximations directly in the main problem, where we can use a distributed iterative algorithm, solving the problem node by node. We only change the restriction $t_l\leq\Phi_l(\boldsymbol{v})$, and for each $l\in n$ we put (from (3.22)):

$$\Phi_l(\boldsymbol{v})=\sum_{l\in n}\omega_l\left(2\log\left|\boldsymbol{h}_{ll}^H\boldsymbol{v}_l\right|-\log\left(\sum_{j\notin n}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2+\sigma_l|\boldsymbol{u}_l|\right)\right) \quad (3.25)$$

And for each $l\notin n$ (from (3.24)):

$$\Phi_l(\boldsymbol{v})=\sum_{l\notin n}\omega_l\left(\log\left(\left|\boldsymbol{h}_{ll}^H\boldsymbol{v}_l\right|^2\right)-\log(N_l)-\frac{\sum_{j\in n}\left|\boldsymbol{h}_{lj}^H\boldsymbol{v}_j\right|^2}{N_l}\right) \quad (3.26)$$

The followed algorithm would be equivalent to the one in Table 3.1.

4 DECOMPOSITION METHODS

Both the original problem (2.4) and our variations exposed in chapter 3 can be solved using decomposition methods, and actually, a first step is already developed in [4], where the problem is separated into two subproblems: the network and the communications subproblems, which are related by the links capacity constraint. That paper also suggests a further decomposition of these subproblems into smaller ones. A basic explanation of decomposition methods can be found at the appendices.

In this chapter, we are going to present different decompositions of the presented problems, and explain what could be practical interpretations of each one. First, a decomposition of the main problem (2.4) into two subproblems will be explained, following the explanation in [4], then these subproblems will be further decomposed, and finally a decomposition of the problem into subnetworks will be analyzed.

4.1 Review of the initial dual decomposition

Considering the original formulation (2.4), and the maximum-utility problem,

$$f(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t},\boldsymbol{r}) = \sum_{d}\sum_{n,n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) \tag{4.1}$$

we can form the dual problem, by introducing Lagrange multipliers $\boldsymbol{\beta} \in \Re^L$ only for the L coupling constraints between the network flow variables and the communications variables, which are the capacity constraints $t_l \leq \Phi_l(r_l)$, resulting in the *partial Lagrangian*:

$$L(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t},\boldsymbol{r},\boldsymbol{\beta}) = \sum_{d}\sum_{n,n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) - \sum_{l}\beta_l\left(t_l - \Phi_l(r_l)\right) = \left(\sum_{d}\sum_{n,n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) - \sum_{l}\beta_l t_l\right) + \sum_{l}\beta_l\Phi_l(r_l) \tag{4.2}$$

The dual function, i.e., the objective function of the dual problem, is defined as

$$V(\boldsymbol{\beta}) = \sup_{\boldsymbol{x},\boldsymbol{s},\boldsymbol{t},\boldsymbol{r}}\left\{ L(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t},\boldsymbol{r},\boldsymbol{\beta}) \;\middle|\; \begin{array}{c} \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)},\;\; \boldsymbol{x}^{(d)} \succcurlyeq 0,\;\; \boldsymbol{s}^{(d)} \succcurlyeq_d 0 \;\; d = 1,\dots,D \\ t_l = \sum_{d} x_l^{(d)},\;\; l = 1,\dots,L \\ \boldsymbol{F}\boldsymbol{r} \preccurlyeq \boldsymbol{g},\;\; \boldsymbol{r} \succcurlyeq 0 \end{array} \right\} \tag{4.3}$$

where $\sup_x\{f(x,y)|\text{constraints}\}$ denotes the maximum of the function $f(x,y)$ when optimizing respect to x.

We can see that the dual function can be evaluated separately in the network flow variables $\boldsymbol{x},\boldsymbol{s},\boldsymbol{t}$ and the communications variables $\boldsymbol{r}$: $V(\boldsymbol{\beta}) = V_{\text{net}}(\boldsymbol{\beta}) + V_{\text{comm}}(\boldsymbol{\beta})$, where

$$V_{\text{net}}(\boldsymbol{\beta}) = \sup_{\boldsymbol{x},\boldsymbol{s},\boldsymbol{t}}\left\{ \sum_{d}\sum_{n,n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) - \sum_{l}\beta_l t_l \;\middle|\; \begin{array}{c} \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)},\;\; \boldsymbol{x}^{(d)} \succcurlyeq 0,\;\; \boldsymbol{s}^{(d)} \succcurlyeq_d 0 \;\; d = 1,\dots,D \\ t_l = \sum_{d} x_l^{(d)},\;\; l = 1,\dots,L \end{array} \right\} \tag{4.4}$$

$$V_{\text{comm}}(\boldsymbol{\beta}) = \sup_{\boldsymbol{r}}\left\{ \sum_{l}\beta_l\Phi_l(r_l) \,|\boldsymbol{F}\boldsymbol{r} \preccurlyeq \boldsymbol{g},\;\; \boldsymbol{r} \succcurlyeq 0 \right\} \tag{4.5}$$

The Lagrange dual problem associated with the primal problem (2.4) is given by

$$\begin{array}{ll} \text{minimize} & V(\boldsymbol{\beta}) = V_{\text{net}}(\boldsymbol{\beta}) + V_{\text{comm}}(\boldsymbol{\beta}) \\ \text{subject to} & \boldsymbol{\beta} \succcurlyeq 0 \end{array} \tag{4.6}$$

Since the dual function is always convex [7], this is a convex optimization problem. We assume that Slater's condition for constraint qualification [7, p. 226] is satisfied for the SRRA problem, i.e., there exists a feasible solution such that the capacity constraints (the only nonlinear constraints) hold with strict inequality $t_l < \Phi_l(r_l),\ l = 1, \dots, L$ (this is almost always true in practice [4]). With this assumption, we conclude that strong duality holds, i.e., the optimal values of the dual problem (4.6) and the primal problem (2.4) are equal. This allows us to solve the primal via the dual.

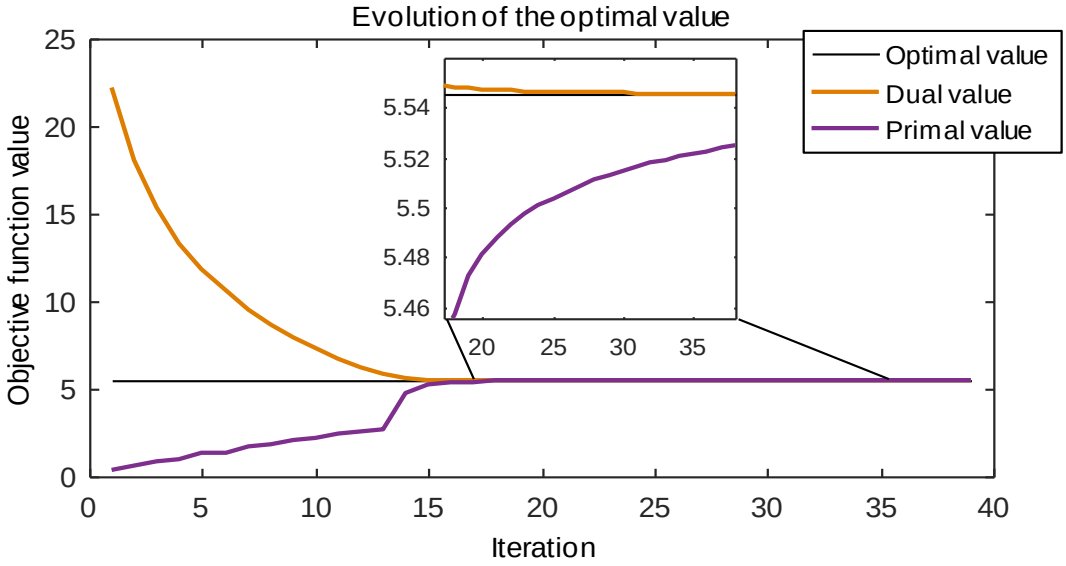

Figure 4.1 Initial decomposition: evolution of the optimal value. If we zoom in in the final iterations, we can see that it continues converging to the solution

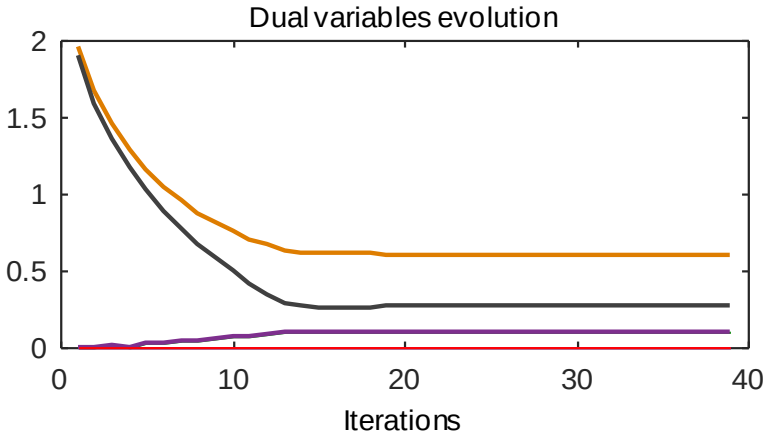

Figure 4.2 Initial decomposition dual variables evolution

4.2 Decomposition of the network flow subproblem

In this section, we explain the decomposition of the Network Flow Subproblem into D single-commodity flow problems. Initial problem (4.4):

$$\begin{aligned} \text{maximize} \quad & \sum_{d} \sum_{n,n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) - \sum_{l} \beta_l t_l \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)},\ \ \boldsymbol{x}^{(d)} \succcurlyeq 0,\ \ \boldsymbol{s}^{(d)} \succcurlyeq_d 0, \quad d = 1, \dots, D \\ & t_l = \sum_{d} x_l^{(d)}\ , l = 1, \dots, L \end{aligned} \tag{4.7}$$

The only restriction that is not directly separable by destinations is the last one, but with a very simple substitution, we can directly decompose the problem into D subproblems with the only link of parameter β_l,

which is the relation with the resources subproblem. Since in each iteration of the main problem (see the Appendix *Dual decomposition* for more information on the meaning of *iteration*) this parameter is fixed, we can separate these problems as follows:

$$\sum_d \sum_{n,n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) - \sum_l \beta_l t_l = \left\{t_l = \sum_d x_l^{(d)}\right\}$$
$$= \sum_d \sum_{n,n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) - \sum_l \beta_l \sum_d x_l^{(d)} = \sum_d \left(\sum_{n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) - \sum_l \beta_l x_l^{(d)}\right) \quad (4.8)$$

So eventually we have D (for all d) completely independent problems like:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{s}}{\text{minimize}} \quad & \sum_{n\neq d} U_n^{(d)}\left(s_n^{(d)}\right) - \sum_l \beta_l x_l^{(d)} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)}, \;\; \boldsymbol{x}^{(d)} \succcurlyeq 0, \;\; \boldsymbol{s}^{(d)} \succcurlyeq_d 0 \end{aligned} \quad (4.9)$$

The practical interpretation of this decomposition is the following: we still need the information of all the resources in every subproblem, but we can separate the different D destinations. We can imagine that each destination is an MNO that wants its flow to go from one specific point to another, and every link has an associated *price* β_l, meaning that using that link costs β_l per amount of traffic unit used. Each operator has to decide according to its utility function (which can be different for every operator) if it is worthwhile to send more or less traffic through a certain link, seeing the penalization in its objective function the price supposes.

When all the operators have decided the amount of traffic they want to send, the master problem (which is controlled by the InP) updates the prices β_l (if a link was highly demanded, its price is lowered, and the other way around, as in supply and demand), depending on the information arriving from the communications subproblem, until the optimal is obtained. Note that in an iteration the total traffic $\boldsymbol{t}$ is *not* fixed, there is no limit, there is just a cost or price for using it, so the different operators do not depend on each other when fixing its $\boldsymbol{x}^{(d)}$. The choice of the other operators affects because when the main problem calculates the β_l for the next iteration, its value (price) will be greater if the other users also wanted to use that link.

During the main problem iterations, the results may not be feasible, because the sum of flows may be higher than the maximum. But it is easy to reach a feasible –not optimal– solution from there, by just multiplying the value of each link by $t/\sum x$. We may want to do that if we have a limit of iterations or time, and when reaching this limit the result has not reached its optimum value yet. If all the links are under-utilized, this operation will actually increase the flow [11].

4.3 Decomposition of the resource allocation subproblem

We can also decompose the communications subproblem (resource allocation) into N subproblems. From (4.5):

$$\begin{aligned} \text{minimize} \quad & -\sum_l \beta_l \Phi_l(r_l) \\ \text{subject to} \quad & \boldsymbol{F}\boldsymbol{r} \preccurlyeq \boldsymbol{g}, \quad \boldsymbol{r} \succcurlyeq 0 \end{aligned} \quad (4.10)$$

where for the moment we consider $r_l \equiv p_l \rightarrow \Phi_l(p_l) = \log(1 + k_l p_l)$, this is, we only consider the power resources. Then, the only related links are the ones starting from the same node, as they share the restriction of the maximum power per node ($\boldsymbol{F}\boldsymbol{p} \preccurlyeq \boldsymbol{g}$). This way, we can separate the problem into N subproblems like:

$$\begin{aligned} &\text{minimize} && \sum_{l\in\mathfrak{O}(n)} -\beta_l \log(1+k_l p_l)\,, \\ &\text{subject to} && \sum_{l\in\mathfrak{O}(n)} p_l \le g_n,\ \ p_l \ge 0\ \forall l \in \mathfrak{O}(n) \end{aligned} \tag{4.11}$$

where $g_n \equiv P_{max_n}$

The problem (4.11) has exactly the same form as (4.10), but only uses information of the links related to each node, so each node can optimize its output power without knowing anything about the others (we can implement a distributed solution). This problem can be solved using the classical waterfilling algorithm [7, p. 245], which further decomposes the problem link by link and relates them through a dual variable.

We can also optimize the bandwidth. To that end, in the rest of the section, we will consider $\boldsymbol{p}$ is **fixed**, and it includes the power, the channel gain and the noise spectral density:

$$\Phi_l(w_l) = w_l \cdot \log\left(1+\frac{p_l}{w_l}\right) \tag{4.12}$$

Now the bandwidth has to be taken into account, so the restriction is no longer local node by node, but is more general. To simplify the study, we consider the general restriction of $\sum_l w_l \le W_{total}$.This way we can ensure orthogonality between links. It is the basic case presented at the beginning of the section 3.1.

In this case, our strategy is to modify the waterfilling algorithm so that we can solve also this problem in a similar way. Then, there will be L subproblems, one for each link, related only by a single dual variable, which is very easy to update. The development of the solution is inspired on the one in [7, p. 245], with the pertinent changes. We will only consider the bandwidth resources, as if the power ones were fixed (this could happen for example if the nodes only had one output link, and then all the power of the node went to that link), or if it was difficult to dynamically modify the output power of a node.

The problem formulation (with fixed $\boldsymbol{p}$) is as follows:

$$\begin{aligned} &\text{minimize} && \sum_{l} -\beta_l w_l \cdot \log\left(1+\frac{p_l}{w_l}\right) \\ &\text{subject to} && \boldsymbol{w} \succcurlyeq 0,\ \ \mathbf{1}^T\boldsymbol{w} = W_{total} \end{aligned} \tag{4.13}$$

Lagrangian: $$L(\nu, \boldsymbol{w}) = \sum_l -\beta_l w_l \log\left(1+\frac{p_l}{w_l}\right) + \nu\left(\sum_l w_l - W_{total}\right) - \boldsymbol{\lambda}^T\boldsymbol{w} \tag{4.14}$$

We want to obtain the Karush-Kuhn-Tucker (KKT) conditions [7, p. 243]:

$$\begin{gathered} \boldsymbol{w} \succcurlyeq 0,\ \ \mathbf{1}^T\boldsymbol{w} = W_{total}, \boldsymbol{\lambda} \succcurlyeq 0.\ \ \lambda_l w_l = 0,\ \ l = 1, \dots, L \\ \frac{\partial L}{\partial w_l} = -\beta_l \log\left(1+\frac{p_l}{w_l}\right) + \beta_l \frac{p_l}{p_l+w_l}\cdot\frac{1}{\ln 2} - \lambda_l + \nu = 0, \quad l = 1, \dots L \end{gathered} \tag{4.15}$$

If we isolate λ in the last equation we can substitute and obtain:

$$\begin{gathered} \boldsymbol{w} \succcurlyeq 0,\ \ \mathbf{1}^T\boldsymbol{w} = W_{total},\ \ \left(-\beta_l \log\left(1+\frac{p_l}{w_l}\right) + \beta_l \frac{p_l}{p_l+w_l}\cdot\frac{1}{\ln 2} + \nu\right) w_l = 0, \qquad \forall l \\ -\beta_l \log\left(1+\frac{p_l}{w_l}\right) + \beta_l \frac{p_l}{p_l+w_l}\cdot\frac{1}{\ln 2} + \nu \ge 0, \quad l = 1, \dots L \end{gathered} \tag{4.16}$$

Immediately we can see that w_l cannot be zero, because then the $\log\left(1+\frac{p_l}{w_l}\right)$ would be ∞, so we can arrange the third equation in (4.16) to obtain:

$$-\beta_l \log\left(1+\frac{p_l}{w_l}\right)+\beta_l \frac{p_l}{p_l+w_l}\cdot\frac{1}{\ln 2}+\nu=0 \rightarrow \nu=\beta_l \log\left(1+\frac{p_l}{w_l}\right)-\beta_l \frac{p_l}{p_l+w_l}\cdot\frac{1}{\ln 2},\quad \forall l \tag{4.17}$$

Clearly, now the last inequation is completely equivalent to this condition (in the inequation, the equality is always met). To simplify notation, we define:

$$f(w_l) \triangleq \frac{1/\beta_l}{\log\left(1+\frac{p_l}{w_l}\right)-\frac{p_l}{p_l+w_l}\cdot\frac{1}{\ln 2}} \tag{4.18}$$

Now, following the idea of the waterfilling algorithm, we would increase the value of $1/\nu$, which would be the water level, and would calculate all the w_l from

$$\frac{1}{\nu}=f(w_l)\quad l=1,\dots L \tag{4.19}$$

until the sum of all the bw_l is equal to BW_{total}. This equation has always a solution, for $\nu>0$.

Algorithm for calculating the bandwidths

1. Set initial dual variable (water level) $\nu=1$, step $\alpha=0.1$, and $W_{\text{used}}=0$
2. Repeat until $|W_{\text{used}}-W_{\text{total}}|<$ tolerance
   a. For all links l
      i. Find w_l solving equation $1/\nu=f(w_l)$ (with Matlab, using `fzero`)
   b. Update total used bandwidth $W_{\text{used}}=\sum_l w_l$
   c. Update water level $\nu=\max(0,\nu+\alpha\cdot(W_{\text{total}}-W_{\text{used}})/L)$

Table 4.1 Algorithm for calculating the bandwidths

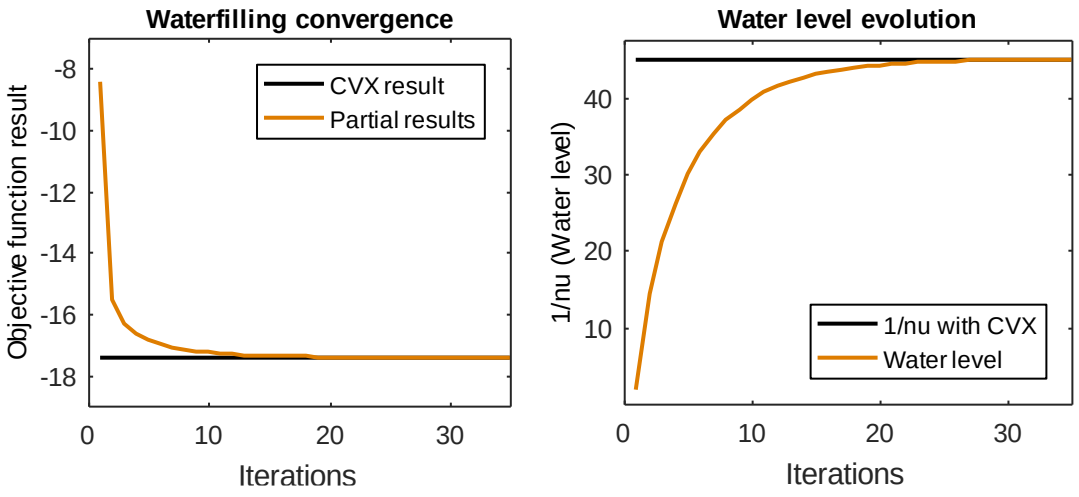

Figure 4.3 Waterfilling results. CVX time: 1.4188s, Waterfilling time: 0.26468s

We work with "increasing" $1/\nu$ instead of "decreasing" ν because then we can give the interpretation of "increasing" the water level. The visual interpretation is not as clear as in the original case, but we will give an example with $L=2$. We have to imagine different "containers" with the shape of $\partial f(p_l,w_l)/\partial w_l$, and we fill them as we increase the water level. Some of them will be wider and will contain more water. We will fill them until we do not have more water (W), which means that the total "area" being represented by water is equal to W_{total} (that is why we work with the derivate, because the actual used value is the area, which is its integral):

$$w_l=f^{-1}\left(\frac{1}{\nu}\right)=\int\left(f^{-1}\left(\frac{1}{\nu}\right)\right)' \tag{4.20}$$

Then, we want to fill $\left(\left(f^{-1}\left(\frac{1}{\nu}\right)\right)'\right)^{-1}$, which has the shape in Figure 4.4.

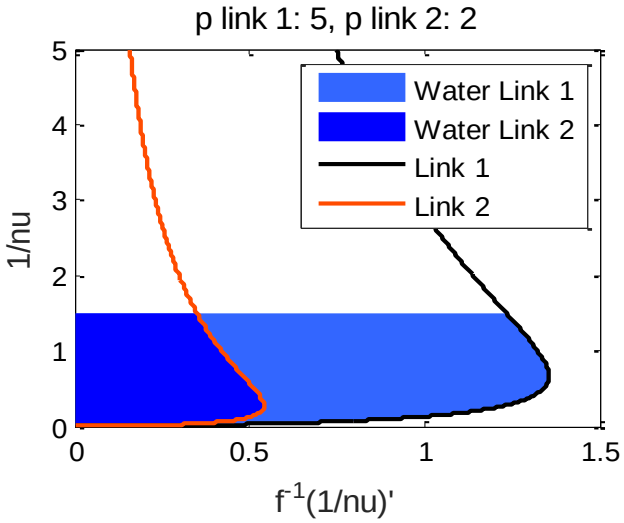

Figure 4.4 Waterfilling solution for the bandwidth assignment in a maximum sum rate optimization problem

4.4 Decomposition for almost isolated subnetworks

In this section, we explain the decomposition of the network in separated subnetworks, which are united by a few links that must be jointly optimized. We present a simple case of only two subnetworks first, in the case of interference avoidance, and then a more complex case with more than two subnetworks for the interference management case. The two approaches can be used in the two cases; we just present the two different cases in order to show how to work with them.

4.4.1 Decomposition of the resource allocation subproblem in the interference avoidance case

In the situation where the nodes are not interfering with all the rest of nodes, but only with some of them, which is the case proposed in 3.1, the decomposition analysis turns out to be much more difficult, as there are restrictions for each node (instead of one general restriction alone), and all of them are related as they use the same variables. However, an interesting analysis can be done considering a special structure of the network.

If we consider separated subnetworks only connected by a few links, then we can optimize each of them separately and, after that, optimize the whole problem from the partial results. The subproblems are not linked by a *restriction*, but by a variable (the bandwidth of the link *a* in Figure 4.5), which appears in a different restriction of each subproblem (in node 2, and in node 1). To solve this problem, we apply a *primal decomposition*.

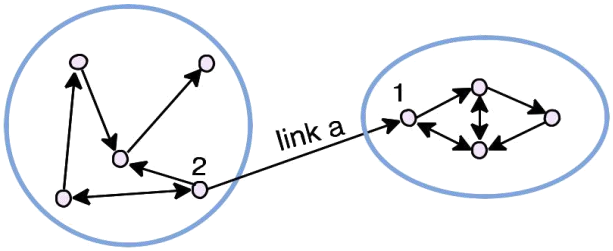

Figure 4.5 Two subnetworks example

Each subnetworks problem would have the form (particularizing (4.5)):

$$\begin{aligned} \text{maximize} \quad & \sum_{l \in L(\text{subnetwork})} \beta_l \Phi_l(p_l, w_l) \\ \text{subject to} \quad & \boldsymbol{F} \cdot \boldsymbol{p} \preccurlyeq \mathbf{g} \\ & \sum_{l \in \mathfrak{O}(n), \mathfrak{I}(n)} w_l \leq \frac{2}{3} W_{total} \ \forall n \in V(\text{subnetwork}) \\ & p_l \geq 0, \ \ w_l \geq 0 \quad \forall l \end{aligned} \tag{4.21}$$

If there is only one link (as in the figure) between the two networks, and that link has only one associated variable (in this case, the bandwidth of the link, as the power only affects in the transmitter), then a simple bisection method can be used, if not, we can use a gradient method.

With the bisection algorithm, we would add a new restriction to relate both subproblems:

$$\nu_i: \ w_{a_i} = t \tag{4.22}$$

where ν_i is the dual variable associated to the restriction. This restriction forces the variables w_{a_i} of the two subnetworks (for $i = 1$ and $i = 2$) to be equal to the previously fixed value t, which is the value that will be updated each iteration.

We consider with loss of generality that the link a belongs to the subnetwork of the transmitter. Here we exploit that the dual variable is in fact the derivate of the objective function as a function of the changes in a restriction (idea further developed in 0. Sensitivity analysis). Then, the bisection algorithm is described in Table 4.2

Bisection algorithm

1. Set initial limits: $w_{min} = 0, w_{max} = W_{total}$,
2. Set initial point $t = (w_{min} + w_{max})/2$
3. Repeat until convergence
    a. Solve subproblems 1 and 2
    b. Calculate total value of the derivate: $\nu = \nu_1 + \nu_2$
    c. Update limits. If $\nu > 0, w_{max} = t$, else, $w_{min} = t$
    d. Update new point $t = (w_{min} + w_{max})/2$

Table 4.2 Bisection algorithm

In a way, the intuitive justification of this procedure is that the subnetworks are agreeing upon the value that they have to let the shared resource take. The following results are obtained from a simulation of the network in Figure 4.5, considering as a reference the distance from node 2 to 1 100 m, with the parameter values defined in 0, and the β_l values randomly generated uniformly between 0 and 1.

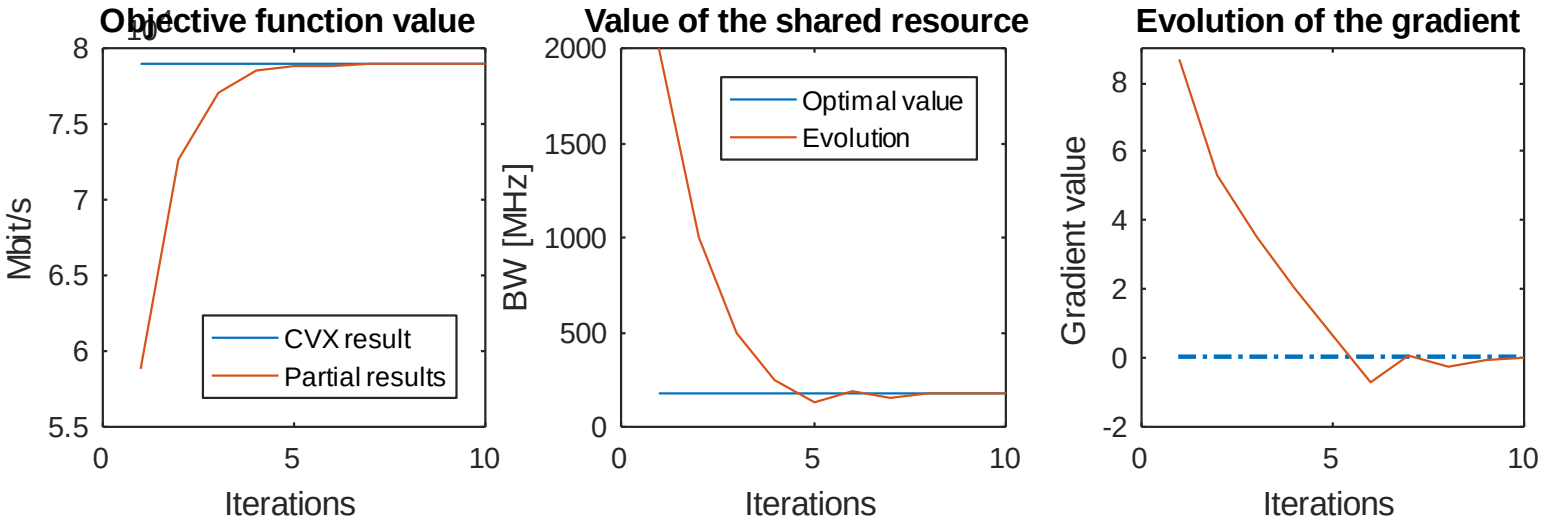

Figure 4.6 Bisection results

4.4.2 SISO case, interference management, communications subproblem

In this section, we introduce a general way of working with several subnetworks, with internal parameters each one, that share general (public) variables. This can be applied in many different problems and fields, and is very well explained in [11], from where we have borrowed the model. We can use this approach when considering several subnetworks related through a few links. Each subnetwork optimizes its own variables with the condition that the shared variables must have the same value in all the subnetworks.

With a fixed $\boldsymbol{\beta}$, the subproblem has the form (from (3.17)):

$$\begin{aligned} &\text{maximize} && \sum_l \beta_l \psi_l(Q) \\ &\text{subject to} && \sum_{l \in \mathfrak{O}(n)} e^{Q_l} \preccurlyeq P_{tot}^{(n)}, \quad n = 1, \dots, N \end{aligned} \tag{4.23}$$

We will suppose K networks, and will denote the set of links in the subnetwork k $L(k)$, and the set of nodes in k $N(k)$. We will consider that the link starting in the subnetwork A and ending in the subnetwork B interferes the links in B and is interfered also by the links in B (as the receiver is in B), so this link will be included in $L(B)$, but its resources will also affect A as the starting node is in $N(A)$. Q_k denotes the resources associated to $L(k)$. Then, the problem can be written as:

$$\begin{aligned} &\text{maximize} && \sum_k \sum_{l \in L(k)} \beta_l \psi_l(Q_k) \\ &\text{subject to} && \sum_{l \in \mathfrak{O}(n)} e^{Q_l} \preccurlyeq P_{tot}^{(n)}, \quad \forall n \in N(k), \quad \forall k \end{aligned} \tag{4.24}$$

In order to make it easy to explain and understand, we will consider a specific network example (Figure 4.7). For each k, we denote x_k the variables that only affect the subnetwork k, and y_k the variables that are shared between subnetworks. In order to relate these variables, define $\boldsymbol{y}$ as the vector containing all the different y_k , and $\boldsymbol{z}$ the vector of public variables (the vector containing the shared value). The values in $\boldsymbol{z}$ have to be related to the values in $\boldsymbol{y}$. We call the links connecting the different subsystems *nets*.

In our example, $y_{A_1} \equiv z_1 \equiv y_{B_1}$. To define this relationship between the public variables, we use a matrix $\boldsymbol{E}$, with as many rows as the length of $\boldsymbol{y}$, and as many columns as the number of shared variables (length of $\boldsymbol{z}$), with a "1" if the $\boldsymbol{y}$ and the $\boldsymbol{z}$ are equivalent, and "0" if not. In our

$$\begin{aligned} &y = [y_{A_1}, y_{A_2}, y_{A_3}, y_{B_1}, y_{B_2}, y_{B_3}, y_{C_1}, y_{C_2}]' \\ &z = [z_1, z_2, z_3, z_4]' \\ &E = \begin{pmatrix} E_A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ E_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ E_C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{pmatrix} \end{aligned} \tag{4.25}$$

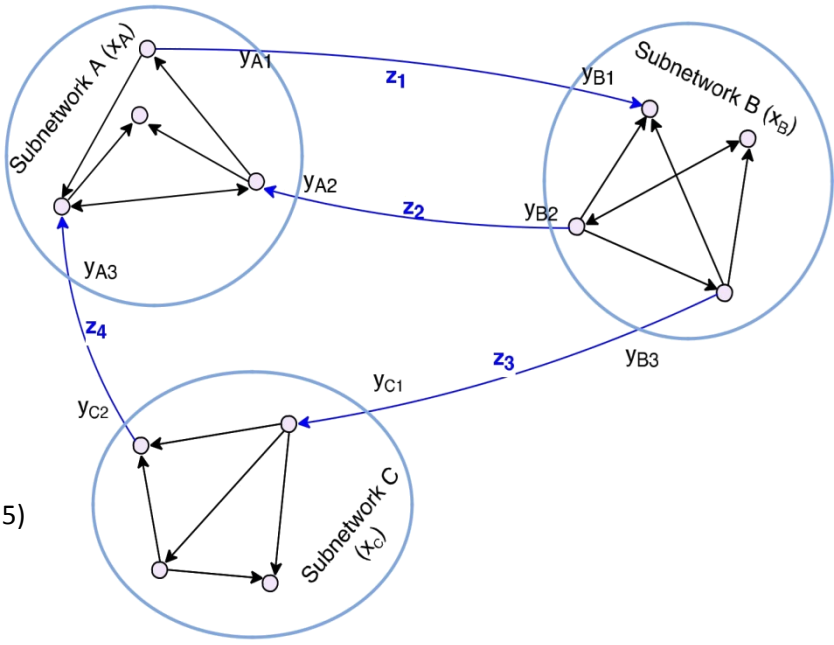

Figure 4.7 Multiple networks example

example:

The final problem would be:

$$\begin{aligned} &\text{maximize} && \sum_{k}\sum_{l\in L(k)} \beta_l \psi_l(x_k, y_k) \\ &\text{subject to} && \sum_{l\in\mathfrak{O}(n)} e^{Q_l} \preccurlyeq P_{tot}^{(n)}, \quad \forall n \in N(k), \quad \forall k \\ & && \boldsymbol{y} = \boldsymbol{E}\boldsymbol{z} \end{aligned} \tag{4.26}$$

This problem can be solved via primal or dual decomposition. In the primal decomposition, we would fix a value for $\boldsymbol{z}$ (and thus, for $\boldsymbol{y}$), then optimize each problem separately, and after that we would update the value of $\boldsymbol{z}$ using the direction of the gradient of the objective function, which would be the sum of all the partial objective functions' gradients. As we will see in the chapter 5, the gradient is given by the value of the dual variable associated with the problem, which is calculated along with the primal solution, so there is no need to calculate any gradient. All the iterations would be feasible.

This algorithm is decentralized: at each step, the actions taken involve only the subsystems, which act independently of each other, or the nets, which act independently of each other. The only communication is between subsystems and the nets they are adjacent to.

Using the *dual decomposition*, each y_k would take different values, and we would update them depending on the distance they have among them, until all of them reach the same value (the value of $\boldsymbol{z}$). As explained in the Appendix Dual decomposition, this is done using the dual variable as a price. The values of the dual variable can be updated using, for example, the subgradient algorithm.

It is important to notice that the iterations are *not* feasible, as the values for the different $\boldsymbol{y}$ are different (and different to $\boldsymbol{z}$). However, it is easy to reach a feasible –not optimal– solution from an iteration, just doing the arithmetical mean of the different values and assigning them to the local variables:

$$\hat{\boldsymbol{z}} = (\boldsymbol{E}^T\boldsymbol{E})^{-1}\boldsymbol{E}^T\boldsymbol{y} \rightarrow y_i = \boldsymbol{E}_i\boldsymbol{z} \tag{4.27}$$

For the previously explained problem, the simulation results are the ones in Figure 4.9, and for the interference avoidance problem (stated partially in 4.4.1 in a simpler case), the ones in Figure 4.8.

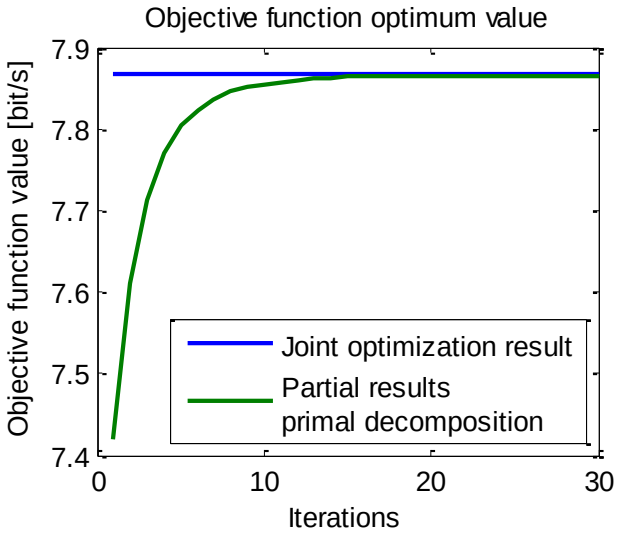

Figure 4.9 Primal decomposition interference management

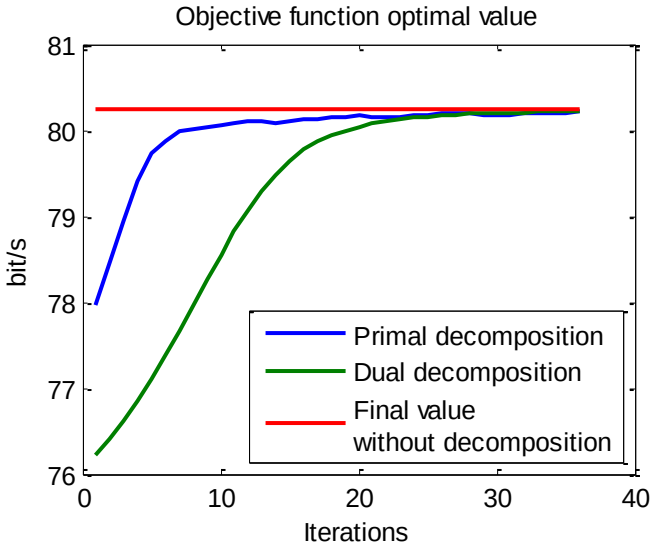

Figure 4.8 Primal and dual decomposition interference avoidance

5 CONSTRAINTS ANALYSIS AND INTERPRETATION

This chapter is intended to explore and exploit the information conveyed by the dual variables (or Lagrange multipliers) when solving convex optimization problems, associated to the infeasibility of the problem or the sensibility of the objective function. We will classify this study into two different parts: first, we will focus on the improvement or deterioration a change on the constraints would entail on the objective function, and then we will focus on the feasibility of the primal problem, and whether or not the changes of some variables can make the problem (in)feasible.

5.1 Sensitivity analysis

In this section, we explore how to control the solution to our problem if we changed the value of the restrictions.

When we have strong duality, the dual variables can be easily shown to be the derivate of the objective function with respect to the constrains values. Moreover, when the objective function is convex, if we define a restriction as $f(x) \leq u$, the optimal value of the objective function as a function of u, $p^*(u)$, will always be greater than the linear approximation of $p^*(u)$ in a point using the derivate $(p^*(0) - \lambda^* u)$ [7, pp. 249 - 252]. This gives us the minimum increase the optimal value will suffer if we tighten the constraint ($u < 0$), and the maximum decrease we will be able to get if we loosen it ($u > 0$).

In our problem, we can apply this idea to several of the restrictions we have defined. We will explain a couple of examples and will give a practical interpretation of the mathematics presented. The idea of this study is to be able to *compare* different values for the amount of resources, this is, to know which is the more limiting node, or the less necessary link, etc., but also can be useful to give an approximated value of how much worse or better the objective function would be.

5.1.1 Resource limits

Given this constraint:

$$\boldsymbol{Fr} \preccurlyeq \boldsymbol{g} \tag{5.1}$$

the dual variable associated has a very clear interpretation: "the higher the dual variable, the more the objective function will vary if we vary the g", as we can write $g_2 = g + \Delta g$, and consider the Δg as the u in the formulation above. We have to note that we can be certain that the objective function will decrease *at least* linearly with Δg when it is negative, so we have a lower bound on its decrease, but we cannot be certain that it will improve a lot if the dual variable is high, we just know its maximum improvement, which is linearly.

To better understand the situation, we can think of the specific objective function of minimizing $\sum_l p_l$, i.e., the total consumed power. We will also consider that the resources are only power resources, so $\boldsymbol{g}$ is the vector of $P_{\max_n}$ associated with each node. The Lagrange multiplier will be a vector of length N. If the nth element of that vector is λ_n, that means that by increasing $P_{\max_n}$ one unity, the objective function can decrease up to λ_n. This example allows us to work with units, being more intuitive: the units of λ_n are Watt/Watt, meaning that if we allow a certain node to spend 1 more Watt, the *total* used power can be decreased by λ_n Watts.

If λ_n is 0, it simply means that that base station is not using its maximum power, so allowing it to use more is simply useless.

5.1.2 Flow conservation law

Assume now the constraint:

$$\left.\begin{matrix} \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)} \\ \boldsymbol{s}^{(d)} \geqslant \boldsymbol{s}_{min}^{(d)} \end{matrix}\right\} \rightarrow \boldsymbol{A}\boldsymbol{x}^{(d)} - \boldsymbol{s}_{min}^{(d)} \geqslant 0 \tag{5.2}$$

This case appear when two operators ask for more traffic, and the InP has to decide which one is allowed to transmit more. Imagine that we have a minimum flow $\boldsymbol{s}_{min}$ we have to guarantee. If this minimum flow increases, the condition in (5.2) gets harder to accomplish, and the dual variable v_i associated to each flow tells us which flow cannot be increased without worsen too much the objective function, which would be the ones with higher dual variables. It does not assure, however, that the others will not worsen a lot the objective function, it is just a lower bound.

It is interesting to notice that the two expressions (equation and inequation) at the left will give the same dual variables, as they are actually expressing the same restriction.

As seen in Figure 5.1, it is a local derivative: the variation is local. If we move little values, the prediction done with the dual variables will be more reliable.

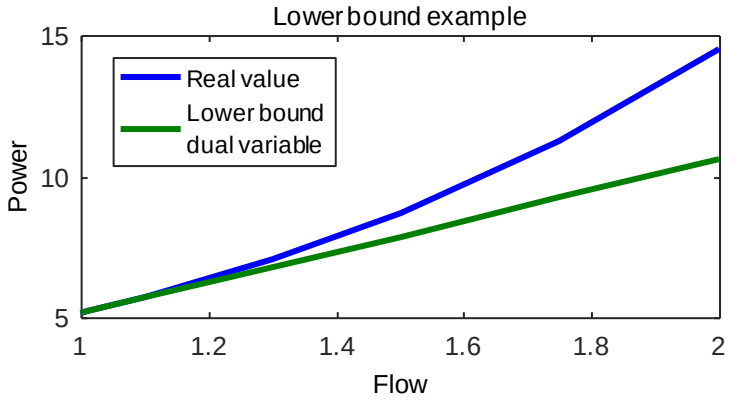

Figure 5.1 Objective function depending on the flow demand

5.2 Infeasibility analysis

The main purpose of this section is to analyze when the problem turns out to be infeasible, and what to do when it occurs, in terms of what are the best changes to do on the constraints. There are two different options to apply.

5.2.1 Objective with a restriction

The first one is based on the idea we just saw, that the dual variables are the derivate of the objective function. To use this, we set a maximum for the optimal value. If, after optimizing (minimizing) the problem, the optimal value is greater than the maximum we fixed, then we will consider the problem infeasible (as explained in [12]). The dual variables will give us information on what are the restrictions that are impeding the optimal value to be inferior to the fixed maximum.

The way forward in this case is to modify the restriction with the higher dual variable (loosening it). As the objective function is convex, we can be certain of the minimum quantity we will have to loosen the conditions, but not of the maximum. Hence, the algorithm proposed in [12] has to be iterative (generalizing for any objective function and interpretation) as the one in Table 5.1.

Algorithm for reaching feasibility

1. Set initial parameters (including restrictions and p_{max})
2. Solve optimization problem and obtain optimal value p*
3. while $p^* > p_{max}$ do
    a. Choose restriction with the highest dual variable.
    b. Calculate minimum change (loosening) in that restriction.
    c. Solve optimization problem and obtain optimal value p* with the new restriction

Table 5.1 Algorithm for reaching feasibility. The interpretations given to this algorithm are the same as in the section 5.1.

5.2.2 Farkas lemma

For conic convex programs where the objective is *linear* and the constraints are convex, there exists the following lemma [16]:

Lemma (First Farkas-type lemma): *Consider a conic convex program. The primal is strongly infeasible if and only if there exists a dual improving direction.*

What this lemma tells us is that when the primal problem is infeasible, the dual is unbounded, and the solver returns as dual variables the directions that can make the dual unbounded.

We can only apply this lemma when we have linear objective functions, such as minimizing the total power. The use of this theorem is by no means direct; we need the dual function in order to interpret the situation. We will follow the same intuition as in [17], where they use the restrictions of the dual problem, putting those restrictions in function of the improving direction. Then, the restrictions have to be made impossible to accomplish, which means that it is impossible to obtain an improving direction.

We will use this intuition in the network subproblem, when the objective function is to minimize the used power and the minimum flow is fixed. The problem is the one stated in (4.9), with the simplification that the flow is fixed. The restriction we want to analyze is $\boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)}$, and its interpretations will be explained later. To obtain the dual problem of the network subproblem, we just need to notice that it is an LP in the standard form, so we can apply the procedure explained in [7, p. 224]:

$$\begin{array}{lll} & \text{Primal problem} & \\ \text{minimize} & \sum_d \sum_l \beta_l x_l^{(d)} = \sum_d \boldsymbol{\beta}^T \boldsymbol{x}^{(d)} & \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)} \ \forall d & \\ & \boldsymbol{x}^{(d)} \geqslant 0 \ \forall d & \end{array} \qquad \begin{array}{ll} \text{Dual problem} & \\ \text{maximize} & -\sum_d \boldsymbol{s}^{(d)^T} \boldsymbol{\nu}^{(d)} \\ \text{subject to} & \boldsymbol{A}^T \boldsymbol{\nu}^{(d)} + \boldsymbol{\beta} \leqslant 0 \quad \forall d \end{array} \tag{5.3}$$

where $\boldsymbol{\nu}$ is the dual variable associated to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{s}$. If the primal problem is infeasible, and the dual is unbounded, the solver will return D vectors $\boldsymbol{\nu}$ containing the dual improving directions. Our objective is to modify the problem parameters so that the solver cannot give us these vectors $\boldsymbol{\nu}$ containing the dual improving directions (if the solver cannot find these values, the dual will not be unbounded, and for the Farka's lemma the primal will not be infeasible). We have to difficult the existence of a vector $\boldsymbol{\nu}$ that accomplishes the restriction of the dual problem and at the same time makes its objective function tend to infinite. The parameters we can modify are $\boldsymbol{A}$ and $\boldsymbol{s}$, as $\boldsymbol{\beta}$ is given and $\boldsymbol{x}$ and $\boldsymbol{\nu}$ are variables. We can see that $\boldsymbol{s}$ has not influence in the inequation system, so all we can modify is $\boldsymbol{A}$**.** Modifying $\boldsymbol{A}$ implies adding a link to the network, so we want to know which is the link we have to add in order to make the primal problem feasible.

The creation of a new link creates a new restriction in the dual problem: for each d, we have $\nu_{\text{start}} - \nu_{\text{end}} + \beta_{\text{link}} \leq 0$. Our strategy is to create a link so that this new restriction is the one that makes the dual improving

direction impossible to exist. We will make this last inequation difficult to accomplish by taking as the starting node the one with the higher v and the ending node of the link the one with the lower v. For the current $\boldsymbol{\nu}$, the new restriction is not met: $v_{\text{higher}} - v_{\text{lower}} + \beta_{\text{link}} \nleq 0$. This way, the current $\boldsymbol{\nu}$ will not be valid: we are forcing the solver to find a completely different dual improving direction (a similar direction will have the same problem).

This happens for all destinations, so if we only want to add a link, we have to see which the destination that is making the problem infeasible is. There may be more than one, but in order to choose one, the best option is to choose the one that is maximizing $-\sum_d {s^{(d)}}^T \nu^{(d)}$ the most. Certainly, that destination will be making the problem infeasible. Therefore, a suitable algorithm to connect all nodes to their destinations would be the following:

Algorithm to connect the network (from an existing and unconnected network)

1. Solve network sub-problem.
2. while (status == 'Infeasible')
    a. Choose limiting destination d with the highest ${\boldsymbol{s}^{(d)}}^T \boldsymbol{\nu}^{(d)}$.
    b. Choose starting and final node of the new link with the highest and lowest dual variable associated to the condition $\boldsymbol{A}\boldsymbol{x}^{(d)} = \boldsymbol{s}^{(d)}$.
    c. Update $\boldsymbol{A}$ with the new link.
    d. Solve network sub-problem.

Table 5.2 Algorithm to connect the the nodes to their destinations. It does not have vision of more than one added link, but it is an interesting way of seeing how the Farkas Lemma can be applied to our problem or to a similar one.

6 ECONOMIC ANALYSIS

Despite being a rather theoretical study, this project has a direct practical implementation, which can bring economic benefits. In this section, a brief and conceptual analysis of these benefits is explained.

Using the backhauling network example previously given, we can decompose the costs of deploying a network like this in Capital Expenses (CapEx) and Operating Expenses (OpEx):

CapEx	OpEx
Routers	Frequency band license
Servers	Energy
Installation	Node location rental
Software	Maintenance

Table 6.1 Costs of deploying a network

Comparing an optimized network with a non-optimized one, from the previous expenses, and if we consider that the control traffic uses the same network as the data traffic, only the Software and the Servers suppose a higher cost, and they are also necessary in a non-optimization scenario.

We are going to consider that the spent energy is the same in both cases, and that we are optimizing the maximum traffic the network can afford. The idea is the following: if the income is proportional to the traffic we can provide, and the costs are only slightly superior to the non-optimizing case, how much benefits can we earn by optimizing the network? The traffic has to be of the same "quality" in terms of reliability – that is why the objective function is not the total traffic, but a utility function of it, which also maximizes the traffic but at the same time looks for it to be equitable between operators.

Using the utility function of the traffic as the object function, and taking as a reference the network defined in Figure 2.1, the total amount of traffic the network can accept is:

Case	Total traffic
Without optimizing resources (only the routing), equally distributing the resources among the links	44.93 Gbit/s
Interference management case with the Shannon method (for the other optimization cases, the interpretation would be the same, as we can see in Figure 3.4 and Figure 3.5	105.21 Gbit/s

Table 6.2 Traffic comparison between optimizing and non-optimizing resources

This simple example shows that we could more than double the income by just optimizing the bandwidths and the power, only increasing the (CapEx) costs of Software and Servers, and having to devote a (minor) part of that traffic to control traffic. Being this a very basic analysis, it includes the idea of all the concepts explained in this thesis, and gives a clear view of the economical utility of the explained analysis.

This approach is only valid for a high traffic demand; if there is not such a traffic quantity the approach should be different, but following the same idea. We would minimize the spent energy, for example: given the same amount of traffic, and instead of trying to maximize the income, we would fix the income (which depends on the traffic), and would minimize the expenses. Obviously, the two ideas (minimize the expenses and maximize the income) can also be combined.

7 CONCLUSIONS AND FUTURE DEVELOPMENT

The main objective of this project was to optimize the efficiency of a network wireless backhaul, from a virtualization approach, and giving tools to analyze the obtained results. This had to be done both from a theoretical point of view, giving answers to the formulated problems using mathematical tools such as convex optimization or graph theory, and also from a practical point of view, with simulations supporting the theoretical results.

We started introducing the problem we were facing, detailing the considered scenarios, and the mathematical model we worked with throughout the project. Then, specific approaches to the problem were formulated, and a solution for each one was presented: first, the interference avoidance case was solved using graph coloring, and introducing it into the convex optimization model; and then, the interference management case was considered both for SISO and MIMO models, providing a solution based on convex approximations of the channel capacity formulas. The simulations showed great improvements compared to the non-optimizing case, and the decision on which approach is the most appropriate one depends on the considerations that can be done on the network.

After that, decomposition methods for improving the flexibility of the network optimization were introduced, together with intuitive explanations on how to interpret the different decompositions. Finally, an analysis of the problem constraints, based on the dual variables intrinsically present in our convex optimization problem, was explained in order to give more tools to better manage the network.

We consider the project has widely fulfilled its initial goals, because not only theoretical answers have been given to the stated problem, but also a wide variety of cases has been taken into account, reaching a solution for each one, and permitting this way a greater flexibility to the network administrator. Moreover, tools for interpreting the results in terms of limiting restrictions and infeasibility have also been provided and, in addition, these theoretical answers have been supported by practical simulations based on realistic networks, its results permitting us to perform a very positive economic analysis.

For all the above-mentioned, we are more than satisfied with the outcome of the project, and finalize it with a positive and encouraging view on future developments on the subject matter. For example, future research may be done in adapting the results provided in this project to a more specific network, with detailed requirements, both from a theoretical and practical point of view.

From a theoretical point of view, because our model describes the general behavior of a backhaul network, and does not take into account detailed aspects such as packet retransmissions, specific protocols or the inexact knowledge of some parameters. In our project, we tried to be as general as possible in all our solutions, and used a specific network example just as a guide, to be able to analyze the results. However, when facing a specific network, there is no need to provide general solutions, and a detailed analysis can provide solutions that are more useful.

And from a practical point of view, because the true limitations of the SDN approaches can only be evaluated when implementing a real, physical network, and the knowledge obtained from this practical implementation can, at the same time, give feedback to the theoretical analysis.

APPENDICES

Appendix I. System Level Simulator

There is a very important difference between formulating a problem and solving it. In traditional optimization, the key point of the problem was to solve it in an efficient way, choose the initial points and the steps, etc.; formulating it was just putting the problem in a mathematical way. On the contrary, in convex optimization, the *magic* of the problem remains in formulating the engineering problem in a convex way (as it will not be convex in the majority of cases); the resolution of the problem is just applying the current techniques to solve convex problems.

CVX and fmincon

In order to simulate the previous problems we do not program the algorithms entirely, we just put them in a convenient way so that then a *solver* solves it. These solvers are specially designed to work much more efficiently than any code we could write, and depending on the problem some of them may be more appropriate than the others (some accept integers, some are fastest with the logarithms, etc.). Some examples of convex solvers are Mosek[4], SDPT3[5] or SeDuMi[6].

CVX [18] is a tool that facilitates the specification and definition of convex problems in Matlab, and then it calls a solver that solves the problem. Following the example on its website, for a problem like:

$$\begin{array}{ll} \text{maximize} & \|Ax-b\|_2 \\ \text{subject to} & Cx = d \\ & \|x\|_\infty \leq e \end{array} \tag{0.1}$$

The following code segment generates and solves a random instance of this model:

```
m = 20; n = 10; p = 4;
A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1); e = rand;
cvx_begin
    variable x(n)
    minimize( norm( A * x - b, 2 ) )
    subject to
        C * x == d
        norm( x, Inf ) <= e
cvx_end
```

A very detailed explanation of CVX can be found online. One peculiarity of CVX is that it does not accept all convex functions, but only the ones it can transform into a standard and well-defined convex problem (they call it the DCP ruleset). Thus, it is important to know some tricks in order to be able to write our problems. The main one is the following one:

$$\begin{aligned} \boldsymbol{w}\cdot\log\left(1+\frac{\boldsymbol{p}}{\boldsymbol{w}}\right) = \boldsymbol{w}\cdot\log\left(\frac{\boldsymbol{w}+\boldsymbol{p}}{\boldsymbol{w}}\right) = -\boldsymbol{w}\cdot\log\left(\frac{\boldsymbol{w}}{\boldsymbol{w}+\boldsymbol{p}}\right) = \left\{\text{rel\_entr}(x,y) = x\cdot\log\left(\frac{x}{y}\right)\right\} \\ = -\text{rel\_entr}(\boldsymbol{w},\boldsymbol{w}+\boldsymbol{p}) \end{aligned} \tag{0.2}$$

[4] https://www.mosek.com/
[5] http://www.math.cmu.edu/~reha/sdpt3.html
[6] http://sedumi.ie.lehigh.edu/

As the relative entropy (*rel_entr*) is accepted as a standard convex problem and it is known how to deal with it. However, the logarithm-based formulas do not have a very efficient solving, so as far as it is possible, it is convenient to substitute it for equivalent formulations. For example, when we have a sum of logarithms in the objective function:

$$\operatorname{argmax}\left(\sum_i \log(x_i)\right) = \operatorname{argmax}\left(\log\left(\prod_i x_i\right)\right) = \operatorname{argmax}\left(\prod_i x_i\right) \tag{0.3}$$

As the logarithm is a monotonous increasing function. This does not work for the restrictions, so we have to stay with the logarithm.

Matlab has its own nonlinear programming software, called *fmincon*, which has the same functionality as the CVX, which is accepting Matlab instructions and sending them in a proper way to the solver. Fmincon is included in the *Matlab Optimization Toolbox.* There are two main differences between these two.

- Writing the problem: fmincon is much more flexible and accepts different formulations, while CVX is stricter in its formulation. However, CVX is much more intuitive, tidy and easy to modify, it just has the problem that the formulation has to be rethought and *sometimes a problem cannot be formulated.*
- Computing time: as CVX has to transform the model into one of the predetermined models, for small problems fmincon is faster, but for large problems CVX outperforms fmincon. In our simulations, we used mainly CVX, although some of them were done using both systems in order to compare and corroborate results. Different computations have been done comparing these two.

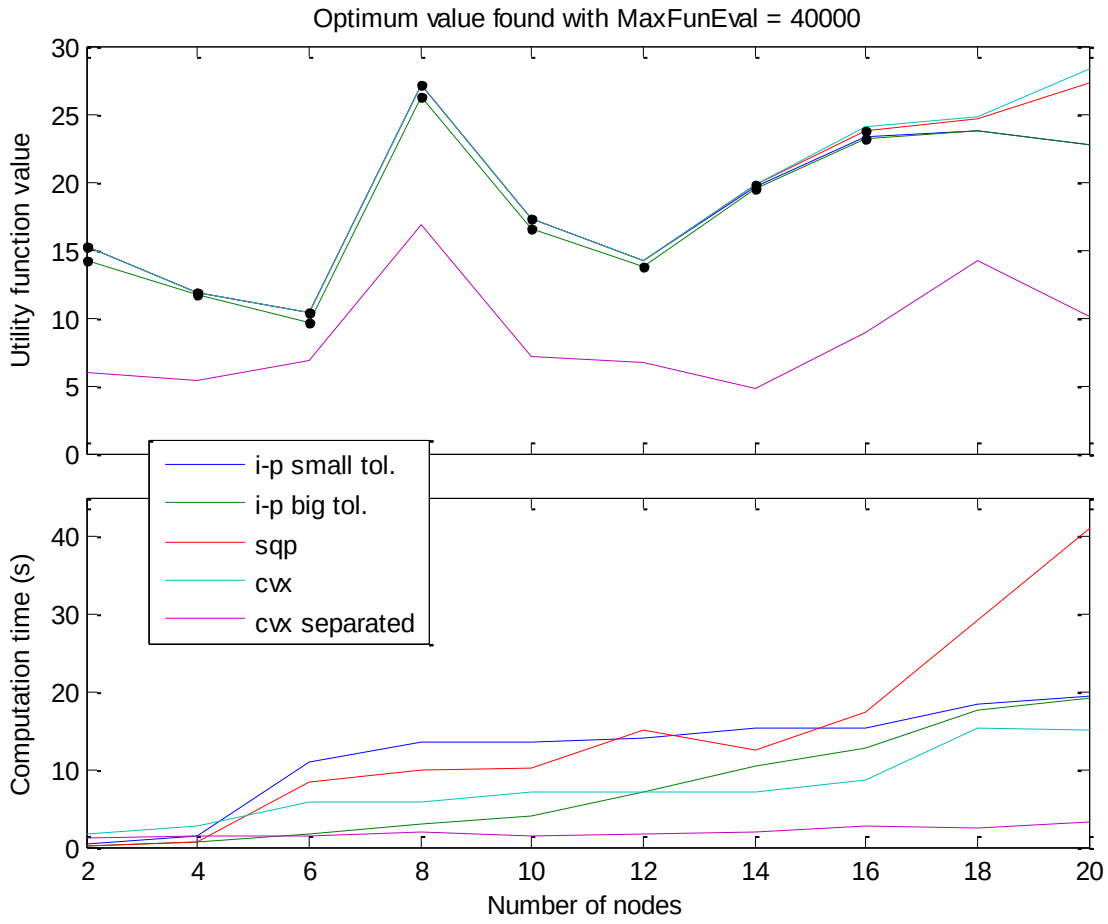

Figure 0.1 CVX vs fmincon

Several parameters can be modified when simulating with fmincon and CVX: the tolerance, the number of iterations, the method used (in the fmincon case), etc. In our simulations, we fixed the maximum number of iterations in the fmincon case to 40000, and did 5 different simulations for the same networks:

- CVX
  - The default CVX mode.
  - Separated: optimizing separately resources and routing (just to compare not only the results, but also the computation time, which is much lower)
- Fmincon
  - With the interior-point method (a method for solving convex optimization problems)
    - Low tolerance
    - Big tolerance
  - With the sqp method

The black points in the graph in Figure 0.1 indicate that the tolerance has been reached, and the absence of black points, that the maximum number of iterations has been reached.

Many interpretations can be taken from the previous simulation, but the most important one is that the CVX optimization is faster for large networks, and always gives the same or better optimal values than the fmincon optimization. Together with the neatness of the code, that is why in our simulations we used CVX.

Simulation program

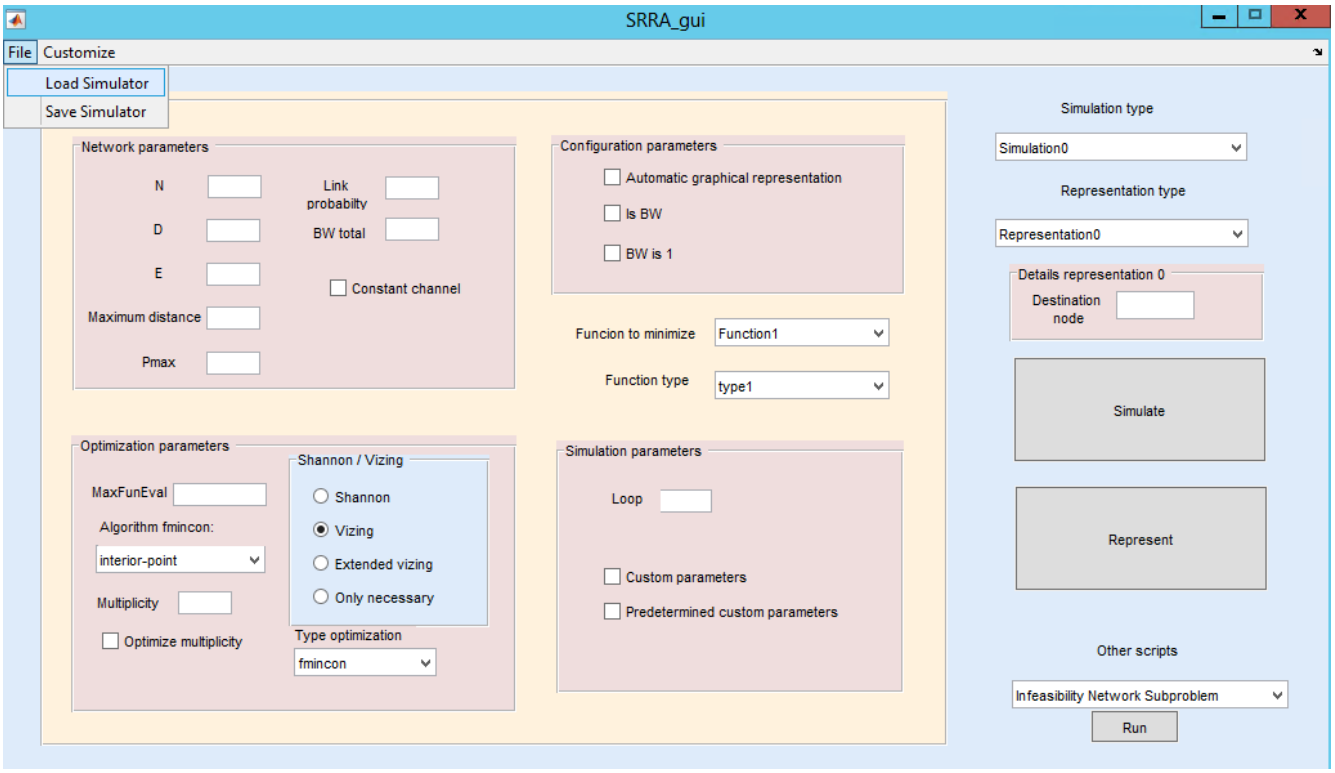

Figure 0.2 Simulation program

For all our simulations, we created a Matlab program where we united all the different parts of the simulations:

- Parameters creation and calculation (manual and random initialization).
- Problem solving.
- Graphical representation.
- GUI enabling the user to modify the parameters, to choose the type of simulation, to save the results...

The complex channels h have been simulated using the following formula:

$$h = \frac{1}{\sqrt{L}} \cdot f_f \tag{0.4}$$

where L represents the open space path losses explained in 0, and f_f stands for *fast fading*, and is modelled by a complex normal distribution with unit power ($f_f \sim CN(0,1)$) , which is simulated with Matlab using:

```
ff = sqrt(1/2)*randn(1)+i*sqrt(1/2)*randn(1)
```

When simulating the channels between transmitters and receivers from different links (for example in the interference management SISO case), the same formula is used, but the channels are multiplied by a constant $K_{lj} < 1$, which takes into account the antenna pattern, its directivity. In our simulations, we used $K_{lj} = 0.01$ for all pairs $(l, j), l \neq j$, except when the transmitter and the receiver are at the same node, where $K_{lj} = 0$, to avoid dealing with the distance between transmitter and receiver being zero.

This is not the most realistic case to simulate these networks, as the ideal would be to consider a real antenna pattern, and calculating the angle between links, and the correspondent antenna gain, etc. This is why the comparisons with the interference management case are not totally fair, because they depend considerably on the value of the constant K_{lj}.

When random nodes are generated, we create a 250 m x 250 m area, and each coordinate *x* and *y* of the node is randomly chosen from a continuous uniform distribution $\text{unif}(0,250)$.

Simulation results

For each simulation, the program saves information of all the simulation parameters (nodes, links, maximum powers and bandwidths…), and also the results, properly classified in a struct.

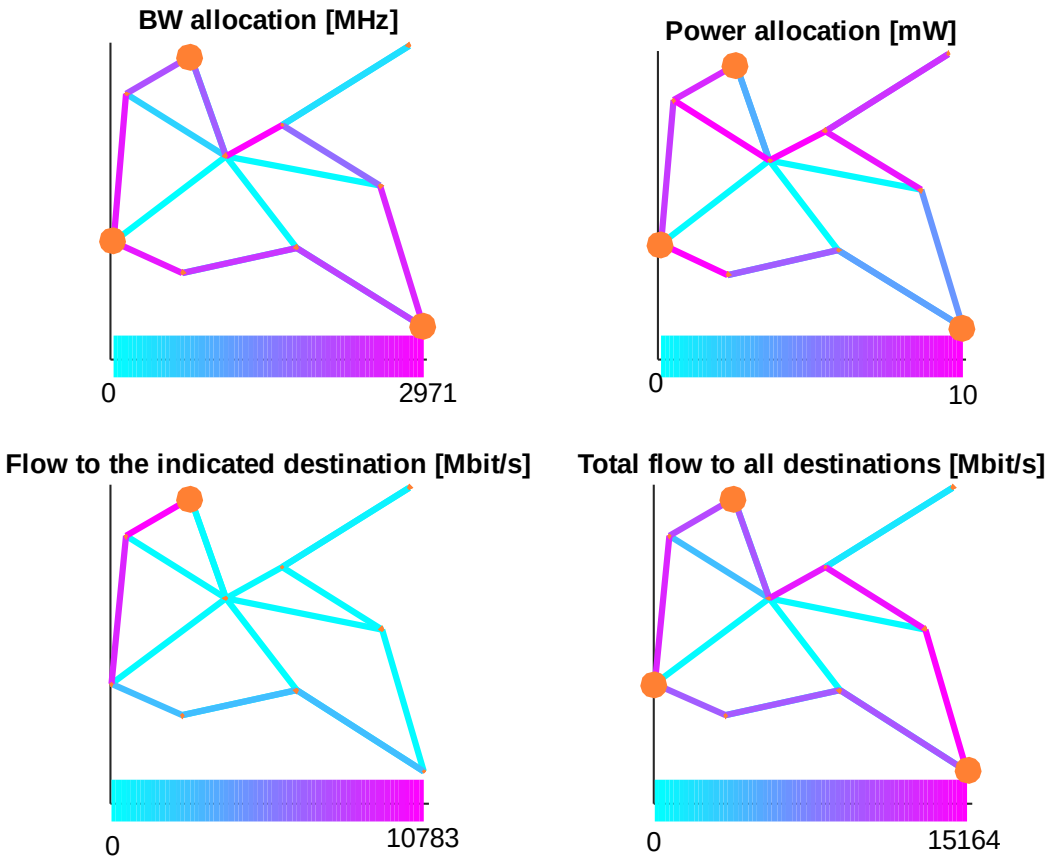

Figure 0.3 Example of simulation results. Results obtained for an interference management case using the Shannon bound, for the network defined in Figure 2.1

The meaning of the previous representation is the following: among the variables given by the solver, some of them are associated to the links of the network ($x, t, r, p, w, time$, etc.). These values are represented on the network topology, so that each link is pained depending on the value of the variable it has associated. The orange dots are the nodes, and the big orange dots are destination nodes.

The units of the solution are "Mega-", because working with values of "Giga-"(Hertz, for example) made the solvers work a slower and made them more prone to fail. Everything has been scaled adequately to provide stability to the system.

Appendix II. Mathematical background

The objective of this appendix is to give a very basic and simplified overview of the mathematical background necessary for this project. It has to be seen as a rather intuitive approach to the explanations, and oriented to the unexperienced (in the world of convex optimization) reader. By no means is the objective of this appendix to give a complete comprehension of the topics treated in the thesis.

Convex optimization

In this section, we present a brief introduction to convex optimization, based on the explanations in [7] and [11], and we also state the main ideas we learned when doing this thesis, so that the reader can quickly understand the mindset behind the thesis.

Convex optimization is a special class of mathematical optimization problems for which very efficient methods of solving them exist. As the name explains, it requires the problem to be convex, and thus it is restricted to certain conditions, but with the proper reformulation, many problems can be expressed as convex problems.

In general, and using the notation in [7], an optimization problem has the form:

$$\begin{array}{lll} \text{minimize} & f_0(\boldsymbol{x}) & \\ \text{subject to} & f_i(\boldsymbol{x}) \leq b_i, & i = 1, \dots, m \\ & h_i(\boldsymbol{x}) = 0, & i = 1, \dots, p \end{array} \tag{0.5}$$

where $\boldsymbol{x} = (x_1, \dots, x_n)$ is the *optimization variable* of the problem, the function $f_0: \mathfrak{R}^n \to \mathfrak{R}$ is the *objective function*, the functions $f_i: \mathfrak{R}^n \to \mathfrak{R}, i = 1, \dots, m$ are the *equality constraint functions*, the constraints $b_1, \dots, b_m$ are the limits for the constraints, and finally the functions $h_i: \mathfrak{R}^n \to \mathfrak{R}$ are the *equality constraint functions*. A vector x^* is called *optimal* of the problem (0.5) if it has the smallest objective value among all vectors that satisfy the constraints.

There are many classes of optimization problems, and a *solution method* for a class of optimization problems is an algorithm that computes a solution of the problem (to some given accuracy), given a particular problem from the class. The effectiveness of these algorithms varies depending on the form of the objective and constraint functions, the number of variables and the structure of the constraints. Thus, the general optimization problem (0.5) is very hard to solve (meaning by hard that it needs a very long computation time, or otherwise, it can exist the possibility of not finding the solution) and we need special cases with effective algorithms.

In a convex optimization problem, the equality constraint functions are affine ($h_i(x) = a_i^T x - b_i$), and the functions $f_0, \dots, f_m$ are convex, *i.e.*, they satisfy

$$f_i(\alpha \boldsymbol{x} + \beta \boldsymbol{y}) \leq \alpha f_i(\boldsymbol{x}) + \beta f_i(\boldsymbol{y}) \tag{0.6}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathfrak{R}^n$ and all $\alpha, \beta \in \mathfrak{R}$ with $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$. For more details on the definition of a convex function, see [7] , chapter 3.

Although there is no general analytical formula for the solution of convex optimization problems, there **are very effective methods** for solving them (like the interior-point methods), and this is why this thesis focus on the use of convex optimization instead of general optimization. The idea behind it is that convex functions only have one local minimum, which is obviously the global minimum (the same happens with quasi-convex functions, which can also be solved through a sequence of convex problems, see [7], chapter 3).

Below a schematic approach on how to work with convex optimization, based on the learning we obtained from the current project, is presented.

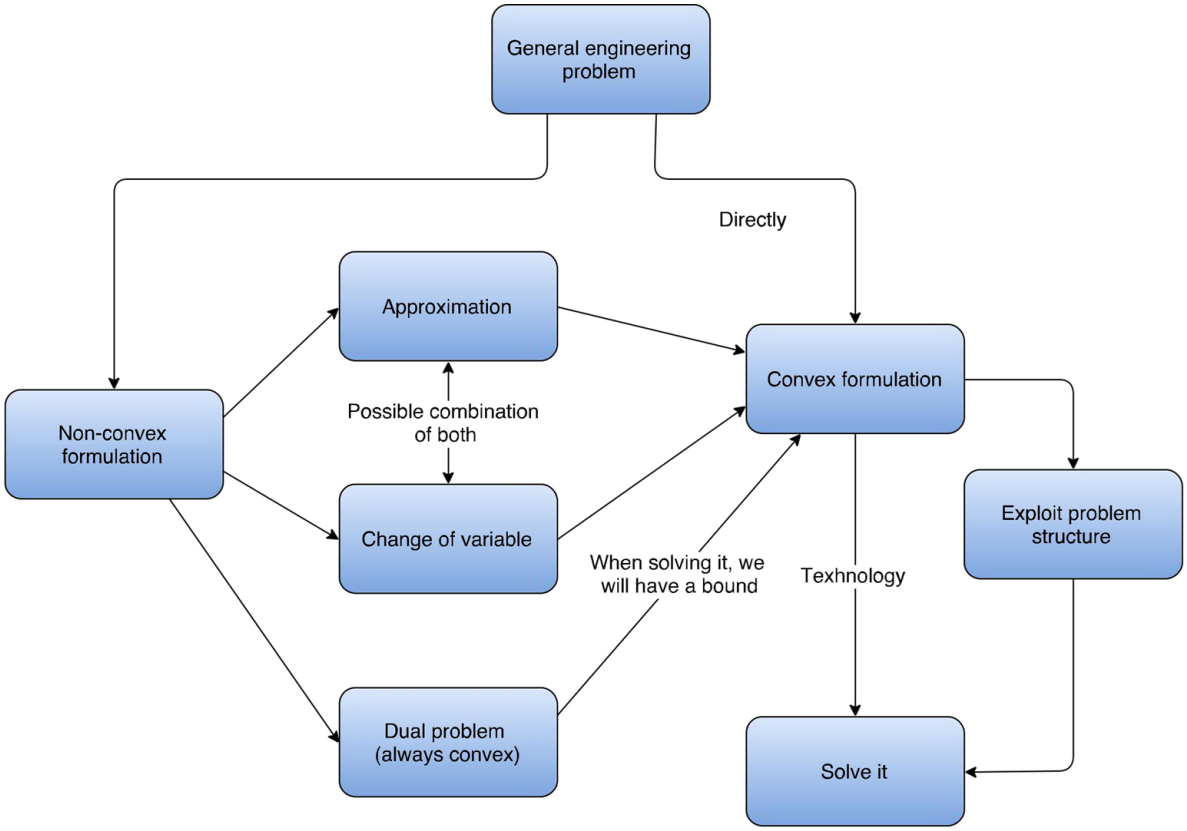

Figure 0.4 Schematic approach to convex optimization

Some ideas must be commented from the above diagram:

- The direct convex formulation from a general problem is, in general, very difficult.
- In convex optimization, going from the general engineering problem to the convex formulation following any of the paths is an art. The difficulty of the problem remains there. **Sometimes it will be just impossible to reach a convex formulation**, and that is something we have to cope with. There is no point in trying to formulate a highly non-convex problem in a convex way. The changes of variable, for example, only work in very few cases. On the contrary, in some other (non-convex) optimization problems classes, formulating the problem is straightforward, and the art remains in its solving.
- Following the last idea, in a convex optimization problem, once it has been formulated in a convex form, solving it is just technology; it is just applying the existing methods. However, if we control these methods, we can exploit the structure of our problem, such as its sparsity or its natural decomposition in smaller problems, so that the solving becomes even more efficient or useful.
- Doing approximations can be useful for several reasons:
    - If we do not want the exact optimal point, it can give a pretty good result.
    - It can be used to found the starting point of a subsequent non-convex optimization.
    - It can be used to give bounds to the optimal value (relaxing the constraints into convex ones, or using the dual function –explained below–).

Duality

The *Lagrangian* of the problem (0.5) is defined as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x}) \tag{0.7}$$

where λ_i is the *Lagrange multiplier* (or dual variable) associated with the *i*th inequality constraint, and ν_i is the Lagrange multiplier associated with the *i*th equality constraint. The *Lagrange dual function* $g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ is the minimum value of the Lagrangian over $\boldsymbol{x}$, and it is always convex.

When we restrain $\boldsymbol{\lambda}$ to be non-negative, the dual function always represents a lower bound on the optimal value of the problem (0.5), and that is very easy to prove, as for any feasible point $\tilde{\boldsymbol{x}}$, $h_i(\boldsymbol{x}) = 0$ and $f_i(\boldsymbol{x}) \leq 0$, so $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f_0(\tilde{\boldsymbol{x}})$. The dual function can be understood as a soft version of the problem, as we are substituting hard and mandatory constraints for a linear approximation, which penalizes the infeasible values: the smaller the value of $f_i(\boldsymbol{x})$, the better the value of the Lagrangian (remember the initial condition was that $f_i(\boldsymbol{x}) \leq 0$).

The *Lagrange dual problem* comes up when we want to obtain the best lower bound:

$$\begin{aligned} \text{minimize} \quad & g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \succcurlyeq 0 \end{aligned} \tag{0.8}$$

The initial problem (0.5) is called the *primal problem*. The dual problem is always convex, so it may be easier to solve than the primal one. We define the *duality gap* as the difference between the optimal value of the primal problem (*p**) and the optimal value of the dual problem (*d**), where $d^* \leq p^*$. If the duality gap is 0, then we say *strong duality* holds, and that is something we want to accomplish, because it would mean that we can solve the primal problem by solving the dual, which may be easier to solve. There are different conditions that ensure the strong duality (such as Slater's conditions [7, p. 226]), and in our problems they hold, as explained below.

Informally, the Slater's conditions say that for a convex problem, the feasible region must have an interior point to have zero duality gap. This means that a solution must exist so that all the inequalities in the problem can be met without equality for that solution. In our problem the inequalities are:

$$s \geq 0, \qquad x \geq 0, \qquad t \leq \Phi(r), \qquad r \geq 0, \qquad Fr \leq g \tag{0.9}$$

The last one is not a problem, as g will always be strictly positive. We just need to use a r such that $Fr = g/2$, for example. Using the previous condition, we can also accomplish $r > 0$. The first two will be easily accomplished without equality as long as $t > 0$, so the only problem can come from $t \leq \Phi(r)$. To reach $t < \Phi(r)$ we just need $\Phi(r)$ to be strictly positive, and using $= \Phi(r)/2$, for example, will be sufficient. In our formulations, $\Phi(r)$ is the Shannon capacity formula, which simplified is $\log(1 + \mathrm{SNR})$. As long as the SNR is greater than zero, $\Phi(r)$ will also be greater than zero, and that will always be the case when $r > 0$. The only problem can come when approximating $\log(1 + \mathrm{SNR}) \cong \log(\mathrm{SNR})$ for high SNRs, then if the SNR is low, the problem may be infeasible, but in any case, if that happens, the model is wrong in the first place anyway.

There is an important group of conditions to check the strong duality:

$$\begin{aligned} f_i(x) &\leq 0, \quad i = 1, \dots, m \\ h_i(x) &= 0, \quad i = 1, \dots, p \\ \lambda_i &\geq 0, \quad i = 1, \dots, m \\ \lambda_i f_i(x) &= 0, \quad i = 1, \dots, m \\ \nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{p} \nu_i \nabla h_i(x) &= 0, \end{aligned} \tag{0.10}$$

These are called the Karush-Kuhn-Tucker (KKT) conditions. For any optimization problem with differentiable objective and constraint functions for which strong duality obtains, any pair of primal and dual optimal points must satisfy them. When the primal problem is convex, the KKT conditions are also sufficient for the points to be primal and dual optimal [7, pp. 243-244].

Decomposition

The decomposition of a convex problem consists in dividing that problem into several subproblems. That division includes the objective function as well as the constraints, leaving more but simpler problems to resolve, and to reach a global minimum, these subproblems need to be related through certain variables and interchange some information. There are several advantages in decomposing the problem into subproblems, and depending on the problem, they can be more or less useful. Among them, we can find:

- Distributed systems.
  - Subproblems can be solved in different servers or processors and only sharing the dual variables the primal problem can be solved: lower computational burden.
  - The solver does not need to know all the parameters of the network. There is encapsulation of information. For example, each MNO can optimize its own traffic with its own considerations.
- Faster solving, if the complexity of the problem scales more than linearly with the number of variables.
- Separation of the problem or system in subproblems or subsystems, with lots of internal variables and a few public (shared) variables.

There are two main ways of doing this division, and each one can be interpreted in a different way. For a complete explanation, see [11], *this is just a summary of its explanations*. In both cases, we will use the following problem:

$$\text{minimize} \quad f_1(\boldsymbol{x}_1, \boldsymbol{y}) + f_2(\boldsymbol{x}_2, \boldsymbol{y}) \tag{0.11}$$

We can appreciate that the variable $\boldsymbol{y}$ prevents us from separating this problem in two completely separated subproblems, that is why it is called the *complicating variable*.

The explanations below present a very simple way of dealing with the problem, because in these examples there are just two subproblems and there are no constraints, but the idea is the same when generalizing the concept to more subproblems (as it is done in the section 4.4.2), and more important, the conceptual interpretations of the variables are the same.

Primal decomposition

We can define two subproblems, number 1 and number 2, and a master problem, like this:

Problem	Optimal value	Definition	
1	$\phi_1(\boldsymbol{y})$	minimize_{x1}	$f_1(\boldsymbol{x}_1, \boldsymbol{y})$
2	$\phi_2(\boldsymbol{y})$	minimize_{x2}	$f_2(\boldsymbol{x}_2, \boldsymbol{y})$
Master	-	minimize_{y}	$\phi_1(\boldsymbol{y}) + \phi_2(\boldsymbol{y})$

(0.12)

A decomposition method solves the problem (0.5) by solving the master problem, using an iterative method such as the subgradient method. We have two subproblems, with private variables or local variables $\boldsymbol{x}_1$ and

$\boldsymbol{x}_2$, respectively. We also have the complicating variable $\boldsymbol{y}$, which appears in both subproblems. At each step of the master algorithm the complicating variable is fixed, which allows the two subproblems to be solved independently. From the two local solutions, we construct a subgradient for the master problem, and using this, we update the complicating variable. Then we repeat the process.

Dual decomposition

We can apply decomposition to the problem (0.5) after introducing some new variables, and working with the dual problem. We first express the problem as:

$$\begin{aligned} \text{minimize} \quad & f_1(\boldsymbol{x}_1, \boldsymbol{y}_1) + f_2(\boldsymbol{x}_2, \boldsymbol{y}_2) \\ \text{subject to} \quad & \boldsymbol{y}_1 = \boldsymbol{y}_2 \end{aligned} \tag{0.13}$$

by introducing a new variable and equality constraint. We have introduced a local version of the complicating variable $\boldsymbol{y}$, along with a *consistency constraint* that requires the two local versions to be equal. Note that the objective is now separable, with the variable partition $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $(\boldsymbol{x}_2, \boldsymbol{y}_2)$.

Now we form the dual problem. The Lagrangian is:

$$L(\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{x}_2, \boldsymbol{y}_2, \boldsymbol{v}) = f_1(\boldsymbol{x}_1, \boldsymbol{y}_1) + f_2(\boldsymbol{x}_2, \boldsymbol{y}_2) + \boldsymbol{v}^T \boldsymbol{y}_1 - \boldsymbol{v}^T \boldsymbol{y}_2 \tag{0.14}$$

which is separable. The dual function is

$$g(v) = \inf_{x_1, y_1} (f_1(\boldsymbol{x}_1, \boldsymbol{y}_1) + \boldsymbol{v}^T \boldsymbol{y}_1) + \inf_{x_2, y_2} (f_2(\boldsymbol{x}_2, \boldsymbol{y}_2) - \boldsymbol{v}^T \boldsymbol{y}_2) = g_1(\boldsymbol{v}) + g_2(\boldsymbol{v})$$

Note that g_1 and g_2 can be evaluated completely independently, e.g., in parallel, from a fixed $\boldsymbol{v}$. Now the idea is the same as in the primal decomposition. Using an iterative algorithm, the $\boldsymbol{v}$ is updated through a subgradient algorithm, and the two subproblems are recalculated. Generally, the iterates are not feasible for the original problem, i.e., we have $\boldsymbol{y}_1 - \boldsymbol{y}_2 \neq 0$.

Dual decomposition has an interesting economic interpretation. We imagine two separate economic units, each with its own private variables and cost function, but also with some coupled variables. We can think of $y_1$as the amounts of some resources consumed by the first unit, and $\boldsymbol{y}_2$ as the amounts of some resources generated by the second unit. Then, the consistency condition $\boldsymbol{y}_1 = \boldsymbol{y}_2$ means that supply is equal to demand. In primal decomposition, the master algorithm simply fixes the amount of resources to be transferred from one unit to the other, and updates these fixed transfer amounts until the total cost is minimized. In dual decomposition, we interpret v as a set of prices for the resources. The master algorithm sets the prices, not the actual amount of the transfer from one unit to the other. Then, each unit independently operates in such a way that its cost, including the cost of the resource transfer (or profit generated from it), is minimized. The dual decomposition master algorithm adjusts the prices in order to bring the supply into consistency with the demand. If the demand is higher than the supply, the price is increased, and if not, the price is lowered (but not below zero).

The dual decomposition process is the following: we start fixing a value for the dual variable $\boldsymbol{v}$. Then, the two subproblems are solved using that fixed $\boldsymbol{v}$, for them, its value is like a constant. Then, the value of $\boldsymbol{v}$ is updated using the subgradient method, which uses the subgradient of the objective functions of the two subproblems. Once the value of the dual variable has been updated, the same process is done, until convergence of the solution. Each one of these processes is called an *iteration* of the main problem. When solving the subproblems, there can also be many inner iterations.

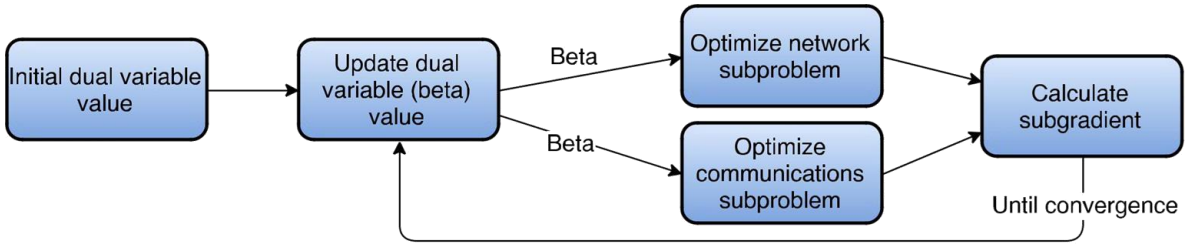

Figure 0.5 Block diagram of each iteration

Graph theory

In this section, a very simple introduction to graph theory is presented. Its purpose is just to give the sufficient graph theory background so that the reader can fully understand the thesis. No further explanations will be attempted. Some definitions have been inspired by those in [19].

First, some important concepts will be defined:

- A graph G(V,E) is a set V of vertices and a set E of edges, where the edges define a relation between the vertices, and this relation is different depending on the type of graph. Some of them (which are not exclusive) are:
  - Undirected graph: each edge is an unordered pair of vertices.
  - Multigraph: Each pair of vertices can be related by more than one edge.
  - Digraph: The starting and ending point of an edge can be the same (that edge is called a *loop*)
  - Weighted graph: each edge has an associated value, which can define its importance.
  - Simple graph: undirected and unweighted graph with no loops and no multiple edges
  - Connected graph: there is a path from any point to any other point in the graph.
- Degree of a vertex, $\deg(v)$: number of edges incident to it.
- Degree of a graph (or maximum degree of a graph), $\Delta(G)$: maximum of the degrees of V(G).
- Incident edges: two edges are called incident if they share a vertex.
- Adjacent or neighbor vertices: if two vertices are endpoints of an edge, we say they are adjacent.

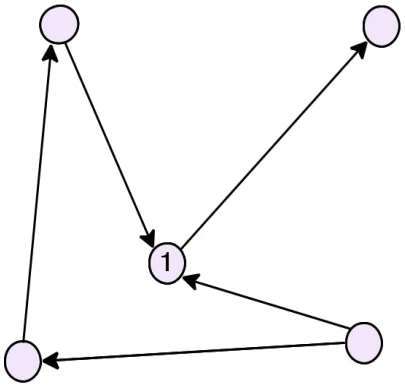

Figure 0.6 Example of a graph G. Type: connected and directed multigraph. Degree of the central node:"deg" (1)=3. Degree of the graph: Δ(G)=3

Graph theory can be used in lots of contexts, so there are many problems that can be formulated through it. One of them is the **graph coloring**, which has two variations: the edge and the vertex coloring. The vertex coloring consists in assigning a color to each vertex so that two adjacent vertices do not have the same color. The problem (called *minimum vertex* coloring) consists in finding the minimum number of colors needed (and

given that number of colors, distributing them among the vertices) in order to accomplish that. Similarly, the edge coloring consists in assigning a color to each edge so that two incident edges do not have the same color.

Related to this problem, some definitions can be stated:

- Chromatic number $\chi(G)$: the chromatic number of a graph G is the smallest number of colors needed to color the vertices of G so that no two adjacent vertices share the same color
- Chromatic index $\chi'(G)$: The chromatic index, sometimes also called the edge chromatic number, of a graph G is fewest number of colors necessary to color each edge of G such that no two edges incident on the same vertex have the same color.

Determining the chromatic number and the chromatic index of a graph is an NP-complete problem, which means that it is very hard to solve (it takes a lot of computing time). Note: the problem is very hard to solve, but very efficient to check, as we just need to go node by node and check that all the neighbors have different colors, and finally count the number of colors. This algorithm will be at most quadratic in the number of vertices ([20]).

That is why in general upper and lower bounds on the chromatic number and index are used. The used bounds are explained in the pertinent section.

Appendix III. Virtualization and SDN

A Software Defined Network is a network where the control plane has been abstracted from the data plane, permitting a dynamic, centralized, programmable and hardware-independent management of the network. This way, the administrator does not need to manage it in a low level, and can have a centralized and general view of its state.

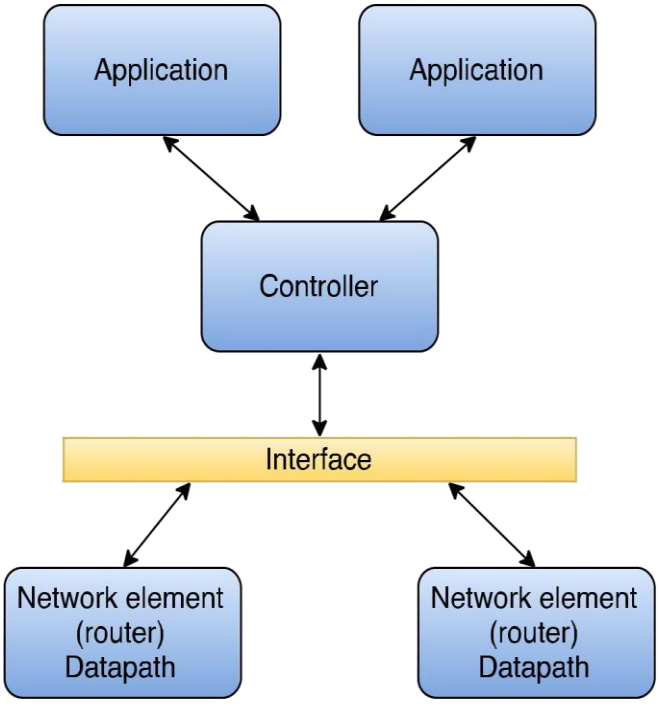

Figure 0.7 SDN structure

The fact of being able to control the parameters of a network via software from a centralized server provides a much more scalable, dynamic and adaptable scenario, and the fact of abstracting this control from the underlying infrastructure makes it simpler to optimize it, as it does not care about the used protocol or technology.

The key point about SDN is that neither the applications nor the controller know about the underlying infrastructure (at least they do not need to know about it); the interface will be in charge of translating the controller instructions to the routers.

By doing this separation between control logic and the physical routers that forward traffic, network operators can write high-level control programs or applications that specify the behavior of the whole network.

The network virtualization is a very close concept, which expresses the creation of an abstract network from a physical one. This can be done in different ways, for example we can have several networks working with different technologies (wired and wireless, for example), and we can create a new one joining them. Inside this new network there will be routers using different protocols, but from our point of view, we just have nodes and connections between them. It is not important how they connect, as long as we know that the connection is guaranteed and its limitations known.

Another example can be starting from a single physical network and dividing it creating several smaller networks. For example, we can imagine that from a network composed by five nodes, we as users are only allowed to use three of them (and the links between them). We do not want to know whether there are more nodes than these three, so our network is a virtual one composed by these three nodes.

From the previous examples, it can be deduced that the virtualization of the networks allows the creation and use of much more dynamic and adaptable networks than the physical ones. For example, if for any reason we need more traffic, we can transform our previous three-node network to a four-node one by just changing an instruction in a controller: this is where the interplay between the SDN and the network virtualization arises. The SDN is the tool we use to manage virtual networks, and having a general view of all the resources and parameters permits make an optimal use of them.

In summary, the concepts of SDN and network virtualization are the ones we based our work on, as the tools we develop fit perfectly in the idea of a centralized controller that manages the network parameters.

Appendix IV. Work plan and incidences

There have been no major incidences in the development of the project. The evolution of the project has revealed that some parts were not as important as we thought they would be, or did not suppose so much work as we predicted, and the other way around.

We would like to highlight that an inflection point in our project was a meeting we had with people from i2CAT Foundation, a research center associated to the UPC, which “promotes mission-oriented R+D+I activities on advanced Internet architectures, applications and services”[7]. They have practical experience in developing backhaul networks, and in implementing software defined networks. Their feedback on our approach corroborated some of our assumptions and developed work, and slightly changed others. It was also helpful to specify the details and consolidate some definitions and scenarios explained in the project, for them to be as realistic as they could possibly be. They showed a lot of interest in our project, because the possibility of determining techniques to optimize the Backhaul can be very useful for they work.

The main change in our work plan, which emerged from this meeting and from our own experiences throughout the project, was that the implementation of our system in OpenFlow was discarded. The reason is that the people from i2CAT considered it was not an important point, that the communication protocols between the nodes and the server can be easily made, it is just a language that the two parts understand. Moreover, the OpenFlow protocol cannot deal with the resources information (which, again, it is not a problem as a custom protocol can be always designed). The important part is not the protocol itself, but assuring a proper communication in terms of capacity and congestion, and that is contemplated in other sections. Instead of doing the OpenFlow study, the theoretical content of our work was increased:

[7] http://www.i2cat.net/en/presentation

- Firstly, the flow admission procedures study has received more importance than it was assigned in the first time, and that is why we decided to create a whole work package for it.
- The decomposition of the problem in subproblems has been found to be also a very powerful tool to exploit, especially in a virtualization context, so we also created a Work Package for it.

The updated work packages, tasks and milestones are the following:

Project: Project Definition	**WP ref: 1**	
Major constituent: Planning	Sheet 1 of 7	
Short description: Definition of the project and the system we are going to deal with. The kind of networks that will be studied and the different scenarios.	Planned start date: 19/02/2016 Planned end date: 27/06/2016	
Internal task T1: WBS/ Planning Internal task T2: Write Final Report Internal task T3: Define Scenario 1 Internal task T4: Define Scenario 2	Deliverables: FinalReport.doc Scenarios.doc ProjectProposalWorkplan.doc	Dates: 27/06/2016 25/03/2016 01/03/2016

Project: Literature Review	**WP ref:2**	
Major constituent: Documentation	Sheet 2 of 7	
Short description: Read documentation about the topic and learn about graph theory and convex optimization	Planned start date: 01/01/2016 Planned end date: 11/04/2016	
Internal task T1: Convex Optimization Tools Internal task T2: Simultaneous Routing and Resource Allocation Internal task T3: Survey on Network Function Virtualization Internal task T4: Other Mathematical Tools	Deliverables: -	Dates: -

Project: Joint Routing and Resource allocation in wireless interference networks	**WP ref: 3**	
Major constituent: Theoretical results	Sheet 3 of 7	
Short description: Provide theoretical results and solutions to the proposed problems	Planned start date: 15/02/2016 Planned end date: 22/04/2016	
Internal task T1: SISO - Interference avoidance based on Graph Colouring Internal task T2: SISO - Interference management Internal task T3: MIMO - Interference management	Deliverables: FlowAdmission.doc InterferenceAvoidance.doc InterferenceManagement.doc	Dates: 21/04/2016 25/03/2016 10/04/2016

Project: Flow Admission Procedures	**WP ref: 6**	
Major constituent: Theoretical results	Sheet 4 of 7	
Short description: Several tools are analyzed in order to better understand the results provided by the problem, interpreting what are the most limiting restrictions and what can be done in order to improve the functioning of the network.	Planned start date: 21/03/2016 Planned end date: 13/05/2016	
Internal task T1: Sensitivity Analysis Internal task T2: Infeasibility Analysis	Deliverables: Sensitivity Analysis.doc Infeasibility Analysis.doc	Dates: 30/04/2016 13/05/2016

Project: Decomposition Methods		WP ref: 6
Major constituent: Theoretical results		Sheet 5 of 7
Short description: Decomposing the problem in several subproblems and solving them separately can be useful to give alternatives to the solving process. Several interpretations of the decomposition and their mathematical study are done in this section.		Planned start date: 7/03/2016 Planned end date: 22/04/2016
Internal task T1: Network Subproblem Decomposition Internal task T2: Resource allocation decomposition Internal task T3: Subnetworks decomposition	Deliverables: NetworkDecomposition.doc ResourceDecomposition.doc SubnetworkDecomposition.doc	Dates: 08/04/2016 22/04/2016 15/04/2016

Project: System Level Simulator	WP ref: 6	
Major constituent: Software	Sheet 6 of 7	
Short description: Simulation of the different proposed networks along with the given solution, in order to evaluate the proper functioning of the results, and to compare the different networks and solutions.	Planned start date: 19/02/2016 Planned end date: 28/05/2016 Start event: T1 End event: T6	
Internal task T1: SISO case - Interference avoidance based on Graph Coloring Internal task T2: SISO case - Interference management Internal task T3: MIMO case - Interference management Internal task T4: Decomposition simulations Internal task T5: Evaluation of the Key Performance Indicators	Deliverables: Simulator.zip	Dates: 28/05/2016

Project: Economic Analysis	WP ref: 7	
Major constituent: Economics	Sheet 7 of 7	
Short description: Economic analysis of the project itself and also of the possible implementation of the designed network.	Planned start date: 19/05/2016 Planned end date: 09/06/2016 Start event: T1 End event: T2	
Internal task T1: Budget for the project Internal task T2: Analysis for deploying	Deliverables: EconomicAnalysis.doc	Dates: 31/05/2016

Table 0.1 Work packages

All the deliverables, excluding the ProjectProposalWorkplan.doc and the Simulator.zip are actually sections of the more general document FinalReport.doc, which has been fill gradually.

Milestones	
Project Proposal and Workplan	01/03/2016
Critical Revision	27/05/2016
Delivery and Validation of the Project Report	27/06/2016
TFG Defense	11/07/2016

Table 0.2 Milestones

7.1.1 Gantt diagram

Id	Nombre de tarea
1	**WP1: Project Definition**
2	WBS/ Planning
3	TFG Writing
4	**Define the scenario**
5	Scenario 1
6	Scenario 2
7	**WP2: Literature Review**
8	Convex optimization tools
9	Simultaneous Routing and Resource Allocation
10	Survey on Network Function Virtualization
11	Survey on Open Flow
12	Other mathematical tools
13	**WP3: Joint Routing and Resource allocation in wireless interference networks**
14	Problem formulation
15	**SISO case**
16	Interference avoidance based on Graph Coloring
17	Interference management
18	**MIMO case**
19	Interference management
20	**WP4: Flow admission procedures**
21	Sensitivity Analysis
22	Infeasibility Analysis
23	**WP5: Decomposition methods**
24	Network flow subproblem decomposition
25	Resource allocation decomposition
26	Subnetworks decomposition
27	**WP6: System Level Simulator**
28	**Build a system level simulation**
29	Initial simulation and structure
30	SISO case 1
31	SISO case 2
32	MIMO case
33	Decompositions
34	Other simulations
35	Evaluation of the Key Performance Indicators
36	**WP7: Economic Analysis**
37	Budget for the project
38	Analysis for deploying
39	**Milestones**
40	Project Proposal and Workplan
41	Critical Revision
42	Delivery and Validation of the Project Report
43	TFG Defense

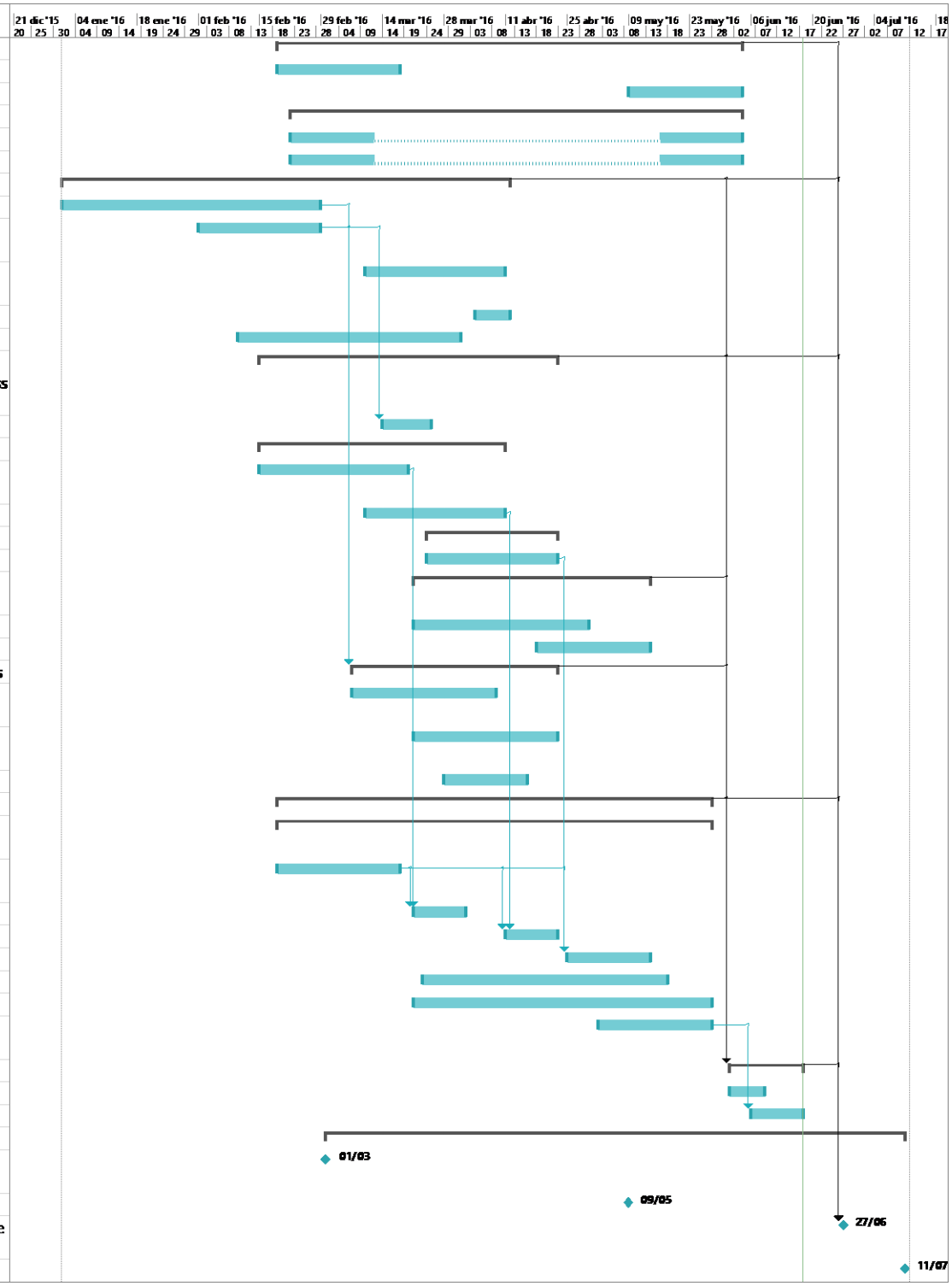

Figure 0.8 Project Gantt diagram

BIBLIOGRAPHY

[1] C. Liang and F. R. Yu, "Wireless Network Virtualization: A Survey, Some Research Issues and Challenges," *IEEE Communication Surveys & Tutorials,* vol. 17, no. 1, pp. 358-380, First Quarter 2015.

[2] A. Hurtado Borràs, J. Palà Solé, D. Camps Mur and S. Sallent Ribes, "SDN Wireless Backhauling for Small Cells," in *IEEE ICC 2015 - Mobile and Wireless Networking Symposium*, 2015.

[3] A. Betzler, F. Quer, D. Camps Mur, I. Demirkol and E. Garcia, "On the Benefits of Wireless SDN in Networks of Constrained Edge Devices".

[4] L. Xiao, M. Johansson and S. P. Boyd, "Simultaneous Routing and Resource Allocation Via Dual Decomposition," *IEEE Transactions On Communications,* vol. 52, no. 7, pp. 1136 - 1144, July 2004.

[5] A. N. Agilent Technologies, *Wireless LAN at 60 GHz - IEEE 802.11ad Explained,* USA, May 30, 2013.

[6] K.-C. Huang and D. J. Edwards, Millimetre Wave Antennas for Gigabit Wireless Communications, John Wiley & Sons, 2008.

[7] S. Boyd and L. Vandenberghe, Convex Optimitzation, New York: Cambridge University Press, 2004.

[8] D. Scheide and M. Stiebitz, "On Vizing's bound for the chromatic index of a multigraph," *Discrete Mathematics,* vol. 309, p. 4920–4925, 2009.

[9] M. Johansson, L. Xiao and Boyd Stephen, "Simultaneous Routing and Power Allocation in CDMA Wireless Data Networks," in *Communications, 2003. ICC'03. IEEE International Conference on*, 2003.

[10] Y. Zhao, S. N. Diggavi, A. Goldsmith and H. V. Poor, "Convex Optimization for Precoder Design in MIMO Interference Networks," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, Monticello, IL, 1-5 Oct. 2012.

[11] S. Boyd, L. Xiao, A. Mutapcic and J. Mattingley, "Notes on Decomposition Methods," Notes for EE364B, Stanford University, Winter 2006-07.

[12] O. Muñoz-Medina, A. Pascual-Iserte, P. Baquero and J. Vidal, "Preemption and QoS Management Algorithms for Coordinated and Uncoordinated Base Stations," in *Signal Processing Advances in Wireless Communications (SPAWC), 2011 IEEE 12th International Workshop on*, San Francisco, CA, 26-29 June 2011.

[13] Z.-Q. Luo and W. Yu, "An Introduction to Convex Optimization for Communications and Signal Processing," *IEEE Journal on Selected Areas in Telecommunications,* vol. 24, no. 8, pp. 1426-1438, 2006.

[14] D. P. Bertsekas and R. G. Gallager, Data Networks, Englewood Cliffs, NJ: Prentice Hall, 1992.

[15] J. M. Gilbert, "Strategies for Multigraph Edge Coloring," *John Hopkins APL Technical Digest,* vol. 23, no. 2 and 3, pp. 187 - 201, 2002.

[16] Z.-Q. Luo, J. F. Sturm and S. Zhang, "Duality Results for Conic Convex Programming," April, 1997.

[17] R. Stridh, M. Bengtsson and B. Ottersten, "System Evaluation of Optimal Downlink Beamforming with Congestion Control in Wireless Communication," *IEEE Transactions On Wireless Communications,* vol. 5, no. 4, pp. 743-751, 2006.

[18] M. C. Grant, . S. P. Boyd and Y. Ye, "CVX: Matlab software for disciplined convex programming (web page and software)," [Online]. Available: http://cvxr.com/cvx. [Accessed 14 04 2016].

[19] E. Weisstein, "Wolfram MathWorld," [Online]. Available: http://mathworld.wolfram.com/. [Accessed 30 04 2016].

[20] Slaviks, "Math StackExchange," [Online]. Available: http://math.stackexchange.com/questions/125136/how-is-the-graph-coloring-problem-np-complete. [Accessed 30 04 2016].

[21] V. R. Cadambe and S. Ali Jafar, "Interference Alignment and Degrees of Freedom of the K-User Interference Channel," *IEEE Transactions on Information Theory,* vol. 54, no. 8, pp. 3425-3441, 2008.

[22] M. Bengtsson and B. Ottersten, "Optimal and Suboptimal Transmit Beamforming," in *Handbook of Antennas in Wireless Communications*, Boca Raton, L. C. Godara, 2001.

[23] L. Chen and S. You, "The Weighted Sum Rate Maximization in MIMO interference Networks: Minimax Lagrangian Duality and Algorithm," *CoRR,* vol. abs/1309.4034, 2013.

[24] R. L. Cruz and A. V. Santhanam, "Optimal Routing, Link Scheduling and Power Control in Multi-hop Wireless Networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, 30 March-3 April 2003.

[25] M. Johansson and L. Xiao, "Scheduling, Routing and Power Allocation for Fairness in Wireless Networks," in *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, 17-19 May 2004.

[26] K. C. Toh, R. H. Tütüncü and M. J. Todd, "SDPT3 – a Matlab software package for semidefinite-quadratic-linear programming, version 4.0".

[27] "Open Networking Foundation - OpenFlow," [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow. [Accessed 14 3 2016].

GLOSSARY

A list of all acronyms in alphabetical order and the meaning they stand for.

BS: Base Station
DCP: Disciplined Convex Programming
FDMA: Frequency Division Multiple Access
InP: Infrastructure Provider
KKT: Karush Kuhn Tucker
KPI: Key Performance Indicator
LP: Linear Programming
MIMO: Multiple Input Multiple Output
MNO: Mobile Network Operator
RAN: Radio Access Network
SDN: Software Defined Networks
SISO: Single Input Single Output
SLA: Service Level Agreements
SRRA: Simultaneous Routing and Resource Allocation
TDMA: Time Division Multiple Access