UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2014-40
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defense held on 21/10/2014 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Apostolos STATHAKIS
Born on 22 April 1983 in Heraklion, Greece

SATELLITE PAYLOAD RECONFIGURATION OPTIMISATION

Abstract

The satellite communications market is a competitive and rapidly evolving one. The interest for services provided by communication satellites, such as television and Internet, has continuously increased over the last years. To better adapt to the market evolution, current and upcoming satellites have become more flexible thanks to reconfigurable components that compose their payload. However, the payloads have additionally increased in size and complexity. As a consequence, their configuration and reconfiguration, which was previously achieved manually by engineers, has become hard, error-prone and time consuming. Many technical constraints have to be satisfied and several operational objectives have to be reached. For instance, when reconfiguring the payload, minimising the number of channel interruptions is of interest, as they impact the provided services to the customers.

This thesis proposes novel efficient optimisation techniques to tackle this challenging optimization problem and help the engineers in their decision making. More precisely, a novel Integer Linear Programming (ILP) optimization model is developed. Several variants of the model are proposed for different operational objectives. Since the problem involves conflicting objectives, multi-objective exact algorithms are additionally applied and compared.

In order to cope with instances that cannot be solved exactly, single-solution and population-based metaheuristics are investigated. These approaches are integrated together with the ILP based exact method, into efficient hybrid algorithms that are designed and implemented in order to further improve the optimisation process. The experimental results demonstrated the effectiveness of the proposed methods. Finally, a modular experimental framework is implemented. The latter incorporates the proposed optimisation techniques and interacts directly with the software tools that are used by engineers for the payload operations.

To family and friends

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Pascal Bouvry for giving me the opportunity to make this PhD at the University of Luxembourg and for the collaboration we had during these years.

I would like to thank the members of my CET committee, Prof. El-Ghazali Talbi and Dr. Grégoire Danoy for their supervision and guidance.

Special thanks to Dr. Grégoire Danoy for his support and advices during these years. His help is greatly appreciated.

Appreciation is also expressed to my advisor at SES Gianluigi Morelli as well as to all payload engineers of SES. My cooperation with them was not only important for my research but also a great and valuable experience for me.

Besides, i have to mention that being part of the team of Prof. Pascal Bouvry was a great pleasure for me. I am thankful to all the members of the team as the interaction with them was very significant to improve my scientific and personal skills.

Last but not least i would like to thank my family and friends for their encouragement and countless support throughout my studies in Luxembourg.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Contents

The industry of communication satellites has rapidly evolved over the last years, as a consequence of an increasing interest of the market for services such as satellite television and Internet. As mentioned in the 17th annual report of satellite industry data published by the Satellite Industry Association (SIA), the global satellite industry revenues (satellite services, launch industry, satellite manufacturing and ground equipment) are augmenting continuously over the last years [3]. As illustrated in Figure 1.1, the revenues had a growth of 35.1% between the years 2008 and 2013. According to the same report, more than half of the operational satellites are communications satellites, whereas the rest are used for navigation, meteorology, scientific and other purposes. The key contributor for the satellite services revenues are the consumer satellite services, namely satellite television, radio and broadband. Especially the satellite television contributed 94% of consumer services. As an example, the number of High Definition Television (HDTV) channels, as depicted in Figure 1.2, had a growth of 14% between the years 2012 and 2013 [3].

The satellite communications service providers operate their fleet of satellites providing services to customers. Their customers include broadcasters, Internet service providers or governmental institutions. As I. Gamvros mentions in [35] the customers "either lease a certain amount of bandwidth or request to transmit a specific amount of traffic between two locations. The provider is then routing the request over a satellite that has the available capacity and is visible from both locations ". The communication payload is the main system that performs the mission of a communication satellite as it is responsible for the reception and the amplification of the signals, the frequency conversion, the appropriate routing and the retransmission of the signals back to Earth. It thus consists of several hardware components, such as amplifiers, multiplexers and channel filters that are interconnected via a large set of switches in charge of relaying signal from a component to another.

The switches used for the routing of the signals compose the payload switch matrices and are controlled via telecommands sent by the ground stations. The switches can be entirely

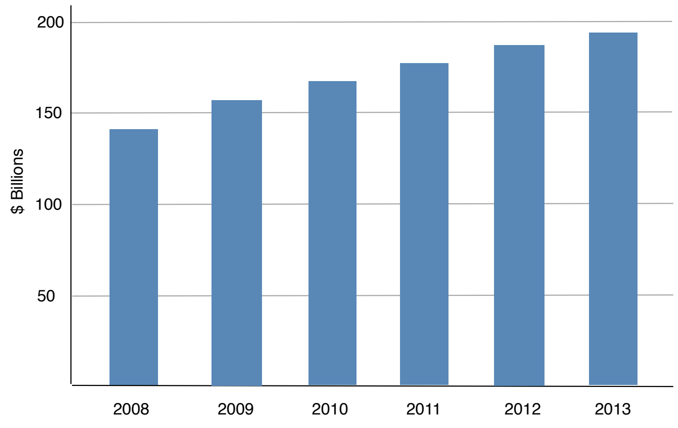


Figure 1.1: Global Satellite Industry Revenues [3].

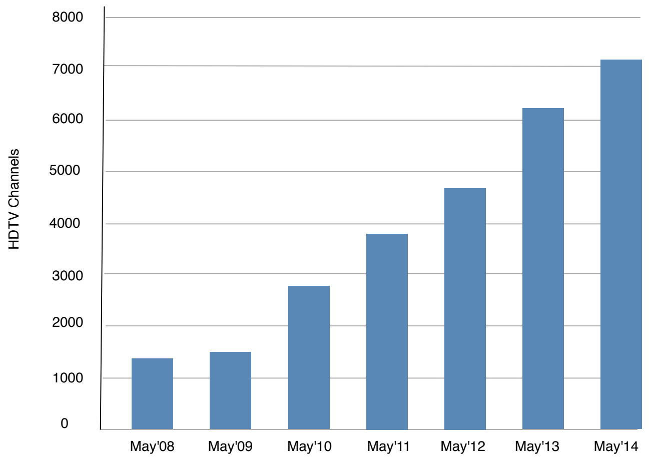


Figure 1.2: Number of HDTV channels [3].

configured, i.e. the position of each switch can be changed, allowing different communication paths for the routing of the signals. This capability permits the service provider to reconfigure the payload depending on new operational needs or due to mechanical or electronic failures that may occur to the payload components during the satellite lifetime.

In order to deal with a dynamic and highly competitive market that is rapidly evolving, the current communication payloads have to ensure flexibility for reconfiguration. This requirement though, comes to the expense of bigger payload sizes with much higher complexity and as a consequence brings to the surface a new optimisation problem of how to efficiently configure and reconfigure the payload of the communication satellites.

1.1 Motivations

With an expected lifetime of 15 years or even more, the current communication satellites require flexibility to answer changing service requirements due to business or operational needs, and recovery capabilities due to hardware failures that may arise during that period. Therefore, the complexity of the payload is rapidly growing and the large size of the switch matrices on modern communications satellites can lead to a large number of solutions with complex routings. As a consequence, the problem of optimal payload configuration and reconfiguration, which was previously solved manually by the engineers with the use of computerised schematics, is now a difficult, time consuming and error-prone process. Many technical constraints have to be satisfied and several operational objectives have to be reached.

Our motivation is to help the engineers to effectively configure and reconfigure the large payload switch matrices of the current communication satellites. Similarly to most of the optimisation problems that arise in the industry, this is a time critical problem and solutions of the highest possible quality are required in reasonable time, in order to prevent long interruptions of the provided services to the customers. Therefore, this thesis proposes efficient optimisation techniques to help the engineers on their decision making.

1.2 List of contributions

The major contributions contained in this PhD thesis include:

1. Problem definition and classification in three related cases, namely the Initial Configuration, the Reconfiguration and the Restoration.

2. A study on the commercial and academic works related to the problem of optimal switch matrix configuration and reconfiguration.

3. Development of the first Integer Linear Programming (ILP) optimisation model for the optimal payload switch matrix configuration and reconfiguration. The model can be used for the application of single and multi-objective exact optimisation algorithms. Several variants of the model allow the optimisation of the different operational objectives. The model permits to deal with all the defined problem cases.

4. Comparison of exact multii-objective algorithms for the initial configuration problem case. We tackle the case of minimising simultaneously the length of the longest channel path and the number of switch changes, aiming to provide to the engineers the set of non-dominated solutions.

5. Application and comparison of metaheuristics for the initial configuration problem case. Population-based and single-solution metaheuristics are applied for the first time to the problem with the aim to solve quickly instances that are not solvable in an exact manner.

6. Design and implementation of hybrid algorithms that integrate the metaheuristics with the proposed ILP based exact method. The target of the hybrids is at first to improve the quality of the solutions obtained by the metaheuristics and in addition to reduce the required by the ILP based exact method computational time.

7. Design and implementation of a generic and flexible optimisation framework that works in conjunction with the internal software tools used by the payload engineers. The framework embeds the developed algorithms and provides solutions to the considered problem. It is applicable to different satellite payload systems.

1.3 Dissertation outline

The remainder of this document is organised as follows:

PART ONE: Related Work, Problem Definition and Experimental Framework
The first part of the dissertation starts with a brief overview on satellite systems in Chapter 2. A description of the communication payload is provided with a focus on the payload switch matrix and its components. Chapter 3 analyses the related works to the considered problem. At first, the different levels of flexibility that are desired by the satellite operators in order to answer the increasing market demands are presented. Focusing on the flexibility at the routing level, the commercial and academic works related to the problem of optimal switch matrix configuration and reconfiguration are detailed. The problem is defined formally in Chapter 4, where the different problem cases are detailed and the considered operational objectives are presented. Lastly, the proposed experimental framework and its modular architecture are described in Chapter 5.

PART TWO: Optimisation with Exact Methods
The Integer Linear Programming (ILP) optimisation model for the switch matrix configuration and reconfiguration problem is detailed in Chapter 6. The variants of the model for each of the defined operational objective are also presented. In addition, the experimental results for the different single-objective problems, with the use of an exact solver, are analysed. Chapter 7 addresses the multi-objective problem where exact multi-objective methods are applied and compared. The numerical results are analysed considering the initial configuration problem case aiming to minimise the length of the longest channel path and the number of switch changes. In both single and multi-objective cases, the limitations of the exact method related to the required computational time are presented.

PART THREE: Optimisation with Metaheuristics
The third part of the thesis deals with the problem of applying approximate methods, in this case metaheuristics. These algorithms are used for the problem instances that are not solvable with the exact method. Approaches for single-solution and population-based metaheuristics are proposed in Chapter 8 and the experimental results are analysed. The hybrid algorithms developed for the considered problem, that integrate the methaheuristics with the proposed ILP based exact method, are presented in Chapter 9. The experimental results demonstrate their efficiency in terms of solution quality and required computational time.

PART FOUR: Conclusions and Perspectives

The work presented in this dissertation is concluded in Chapter 10 and some perspectives for future work are provided.

Part I

Problem Definition and Related Work

Chapter 2

Overview of Satellite Systems and the Communication Payload

Contents

Satellites are widely used in today's communication systems as their main advantage is the wide coverage that enables efficient communications between users located in different regions of the Earth or in isolated areas where there is no terrestrial infrastructure. The main system of a satellite is the communication payload that is responsible for the reception of the signals, the appropriate processing and the retransmission of the signals back to Earth. At first, this chapter provides a brief overview of satellite systems in Section 2.1. The communication payload is described in Section 2.2. Finally, the payload switch matrix and the different components are detailed in Section 2.3.

2.1 A Brief Overview of Satellite Systems

A satellite system, as illustrated in Figure 2.1, consists of a space, a ground and a control segment [49]. The space segment, includes one (or more) satellites in space, each of them containing the platform and the satellite payload, which will be discussed in the next section. The ground segment consists of the user communication devices or the Earth stations that connect to the terrestrial network. The facilities used for the control and the monitoring of the satellites, also named TTC (tracking, telemetry and command) stations are included in the control segment. The type of communication link between a transmitter and a receiver can be uplink (from the Earth stations to the satellites), downlink (from the satellites to the Earth stations), or intersatellite link (between the satellites) [49].

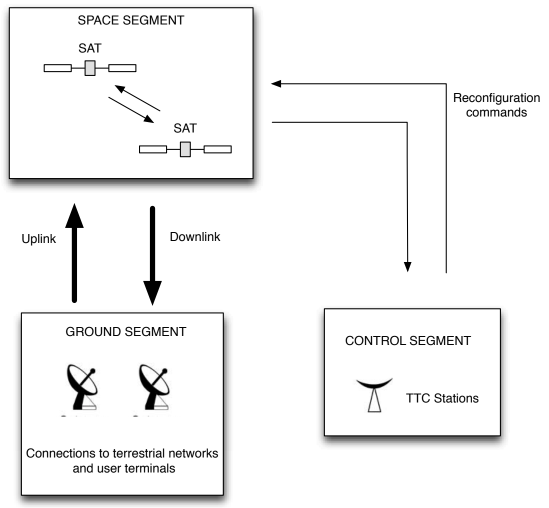


Figure 2.1: A satellite system.

Different frequency bands are allocated to the satellite services according to the International Telecommunication Union (ITU), aiming at efficiently exploiting the limited spectrum. In Table 2.1 the names of the bands, the frequency range and the services that are performed in each of them are summarised [4]. Six frequency bands are used and the main provided services are fixed satellite services (FSS), mobile satellite services (MSS) and broadcast satellite services (BSS). FSS concerns commercial applications that are through Earth stations at fixed locations. BSS is used for individual home reception of TV, radio and data whereas MSS offers voice and data services for ships, aircrafts and individuals [34]. This work deals with FSS and BSS provided services.

Name	Range	Service
L-band	$1.5 - 1.7GHz$	Mobile Satellite Services (MSS)
S-band	$2.0 - 2.7GHz$	MSS, Digital Audio Radio Services (DARS)
C-band	$3.4 - 7.1GHz$	Fixed Satellite Services(FSS)
X-band	$7.25 - 8.4GHz$	Military/Satellite Imagery
Ku-band	$10.7 - 14.5GHz$	FSS, Broadcast Satellite Services (BSS)
Ka-band	$17.7 - 21.2GHz, 27.5 - 31GHz$	FSS Broadband and inter-satellite links

Table 2.1: Frequency Bands and Services.

2.2 The Communication Payload

A communication satellite consists of the payload and the platform. The payload plays the main role in the transmission of the signals whereas the platform is composed by the subsystems that allow the payload functioning, such as the electric power supply. To fulfill its functionalities related to the signals transmission, the payload includes all the necessary electronic equipments such as multiplexers, amplifiers, switches, as well as the receiving and transmitting antennas. As Maral and Bousquet mention in [49] the main functions of the communication satellite payload are as follows:

1. To capture the carriers transmitted, in a given frequency band and with a given polarisation, by the Earth stations.
2. To capture as little interference as possible.
3. To amplify the received carriers while limiting noise and distortion.
4. To change the frequency of the carriers received on the uplinks to that on the downlinks.
5. To provide the power required in a given frequency band at the interface with the transmitting antenna.
6. To radiate the carriers in a given frequency band and with a given polarisation to a given region on the surface of the Earth.
7. For a multibeam satellite, an additional function is routing the carriers from any given uplink beam to any downlink beam.

The bandwidth that is allocated to the service, is divided into several channels (subbands) with the use of an input multiplexer (IMUX), as shown in Figure 2.3. The channels are used for the transmission of the dedicated services and each channel may be associated to one or more customers of the satellite operator. The bandwidth of the channels depends on the band-pass filters used in the IMUX [49].

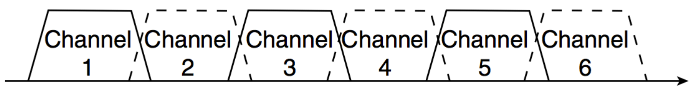


Figure 2.2: Frequency split into several channels [49].

For a multibeam satellite, each beam defines a coverage area on the Earth surface. Figure 2.3 illustrates an example of 5 multiple-shaped beams coinciding with 5 European countries [12].

As aforementioned, for a satellite that features several antenna beams, interconnectivity between the different beams has to be established and this interconnectivity is achieved with the use of the switches. The payload includes a switch matrix having a number of inputs and outputs which connects each uplink beam to each downlink beam. With the use of such switches, that are controlled via telecommands, it is possible to reconfigure the payload. This

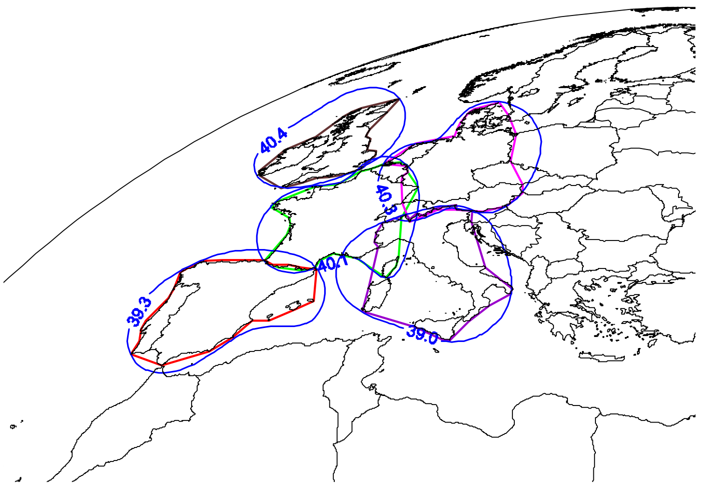


Figure 2.3: Multiple-shaped beams [12].

functionality enables the flexible routing of the channels, as well as the continuous service provision in case of failures.

In Figure 2.4 a simplified block diagram of a communication payload is illustrated [60]. At first, it consists of the receive antenna for the reception of the uplink signal transmitted by the Earth stations (function 1). The input filter is used for filtering the signal from of out-of-band interference and in the front-end block, the low-noise amplifier (LNA) provides gain at the uplink frequency (functions 2, 3). The downconverter (D/C) performs the frequency conversion between the uplink and the downlink (function 4). The switches in the front-end are used for redundancy so that in case of failure, the alternative LNA and D/C will be used. The uplink band is split into several channels with the use of the IMUX. Each channel at the output of the IMUX is directed to one amplifier through the interconnected switches that compose the switch matrix (function 5). After the amplification, the signals are routed to the relevant output multiplexer (OMUX) through the output switch matrix and finally to the transmit antenna for the re-transmission back to Earth (function 6).

2.3 The Payload Switch Matrix

The payload switch matrices are used to route the channels to the appropriate amplifiers for the power amplification and to the corresponding outputs. They enable the flexibility at the connectivity of the channels and allow the payload to be reconfigured. A simplified payload input switch matrix example is provided on the left hand side of Figure 2.5. It contains 8 input channels (as the output of the IMUX), 4 amplifiers and 16 switches of two different types. These components, that are of interest for the considered problem, are described in the following subsections.

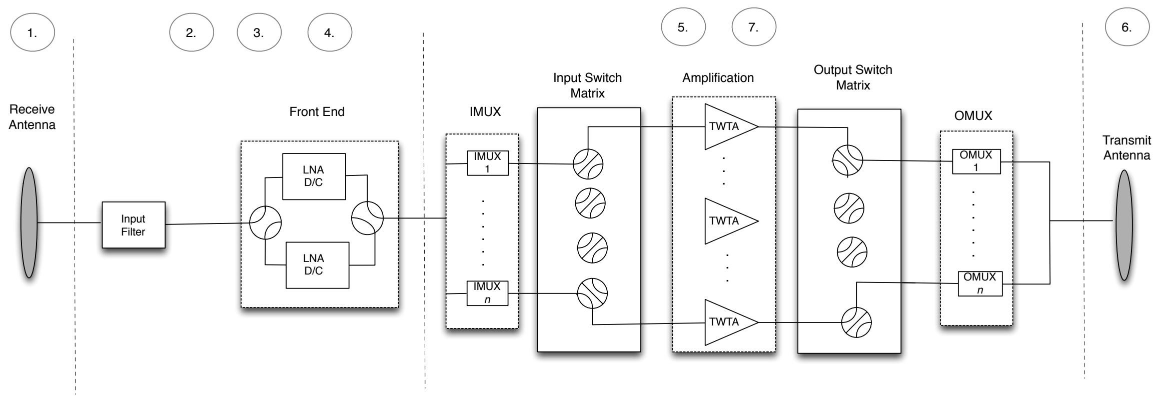


Figure 2.4: Simplified payload block diagram.

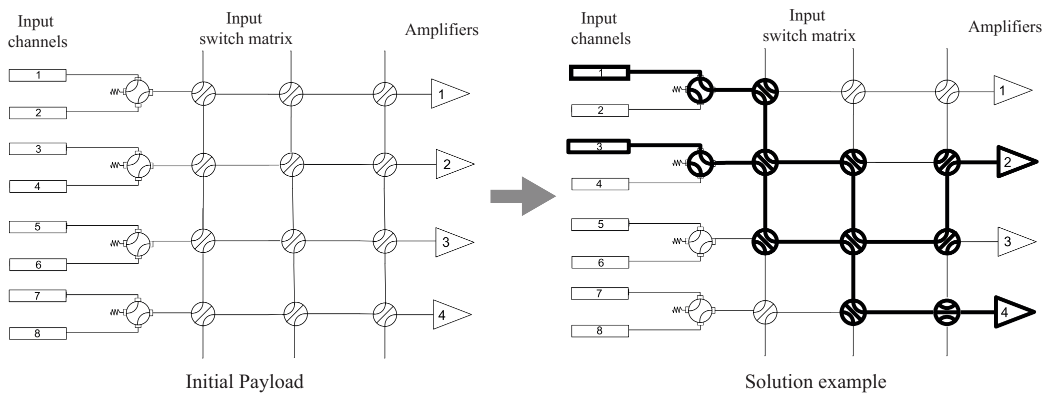


Figure 2.5: Switch matrix simplified example and possible solution for connecting channels 1 and 3 to amplifiers 2 and 4 respectively.

2.3.1 Channels

The channels are characterized by a certain frequency and can be associated to one or more customers for the transmission of their services. Each channel that has to be connected in order to allow the provision of the dedicated services is routed through the switch matrix to an appropriate amplifier. After being amplified, it is routed through the output switch matrix to a corresponding OMUX as illustrated in Figure 2.5. In case of failures in one or more amplifiers or switches, the affected channels have to be rerouted through alternative paths in order to continue the service provision. Each channel is using a unique path in the payload switch matrix.

2.3.2 Amplifiers

Traveling-wave-tube amplifiers (TWTA) are used as the power amplifiers. Each amplifier in the payload may be appropriate only for a specific subset of channels, because of frequency constraints, or suitable for any channel. As illustrated in Figure 2.6, the maximum performance of each amplifier used in the payload is achieved when it is operated at its saturation point, where the maximum amplification of the input signal is achieved. The required Input Power to Saturation (IPS) to drive the amplifier to its saturation point depends not only on the losses cumulated from the used links and switches but also on the channel frequency.

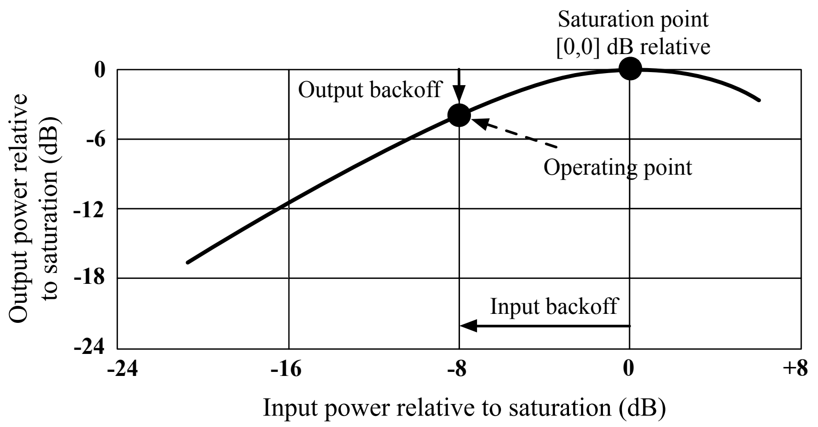


Figure 2.6: Input Power to Saturation and Output Saturated Power.

For a given channel to connect it is of interest for the engineers to find the path to an appropriate amplifier that minimises the required input power to saturation.

2.3.3 Switches

Radio frequency (RF) switches are used for the routing of the signals. Information about the switches and their different types, such as C, R or T type, can be found in [34, 61]. Each switch, depending on its type, has different sets of possible positions. Each position allows different connectivities between the components and therefore different payload topologies. In Figure 2.5 switches of type R and C are used. The 4 possible positions of an R-type switch are shown in Figure 2.7. Other switches, like T-type switches, have only 3 possible positions as shown in Figure 2.8, and in Figure 2.9 the two possible positions of a C-type switch are displayed. The routing of the input channels in the payload is achieved with the switches that can be reconfigured by the ground stations through telecommands. Each switch, depending on its type, cumulates some losses on the transmitted channel.

2.3.4 Links

A link in the payload is a connector between any two payload components. Each switch has 4 neighbouring links while the channels have one neighboring link, as there is only one switch connected to each channel. Each amplifier has one input and one output link. When one switch changes position different links are connected allowing different communication paths. Each link cumulates some losses to the transmitted channel that depend on the frequency of the channel.

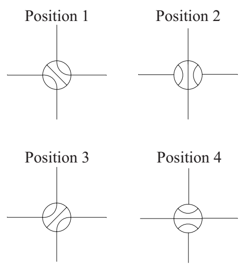


Figure 2.7: Positions of R-type switch.

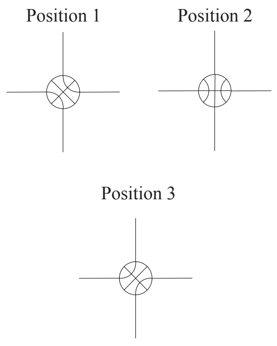


Figure 2.8: Positions of T-type switch.

2.3.5 Topology

The topology of the switch matrix is not necessarily symmetric, as illustrated in the simplified example of Figure 2.5, where a 4∗4 switch matrix is used. Given the operational requirements, the switch matrix is designed by the satellite manufacturer when constructing the spacecraft. The design of the switch matrix topology is made with the aim to minimise the cost (minimum number of required switches), while ensuring all the routing requirements for a given number of channels and amplifiers. Thus, a switch matrix can have any topology and different channels can be placed at different locations of the matrix, i.e some channels may require longer or shorter paths to reach some amplifiers.

2.4 Operational Objectives

When configuring and reconfiguring the payload switch matrix in order to connect a given set of channels, several objectives have to be optimised by the engineers. For instance, changing the position of a switch, while the satellite is operated in orbit, has to be avoided as it may cause the permanent failure of the switch. As a result, minimising the number of switch changes is an important aspect. Besides, each used link and switch cumulate losses on the transmitted channels that affect the needed power for the saturation of the amplifiers. Therefore, finding the paths that minimise the required IPS is of interest. Furthermore, reconfiguration of the switch matrix should be performed without impacting other existing services, or at least, minimising the service degradation. The process should thus minimise the

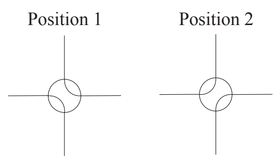


Figure 2.9: Positions of C-type switch.

number of interruptions of the already connected channels that provide services to customers. Lastly, long paths have to be avoided as apart from the losses that they imply, they restrict the future reconfiguration processes. Consequently, minimising the length of the longest channel path (LPL) is another operational objective.

For a given set of channels to connect, Table 2.2 summarises the components that are involved for the calculation of each of the considered objectives.

Objectives	Components involved
Changes	Switches
LPL	Switches/Links
Interruptions	Switches
IPS	Switches, Links, Amplifiers

Table 2.2: Operational objectives and payload components involved.

Chapter 3

Related Works

Contents

In order to cope with the rapid evolutions of the market and to respond to the new commercial opportunities, the communication satellites of the new generation have to achieve several levels of flexibility. This requires equipment flexibility onboard the satellite but also optimal allocation of the resources targeting to the highest economical benefit while ensuring the best possible services to the customers.

This chapter firstly presents the different levels of flexibility that are required by the satellite operators, as presented in [16]. Focusing on one of these levels, the problem of the optimal switch matrix configuration and reconfiguration is presented. The academic works as well as the commercial products on the market that deal with this problem are presented and analysed.

3.1 Flexibility in Satellite Services

One of the main scopes of the required flexibility is to enable the adaptation of the provided services during the lifetime period of the satellite, depending on the market demands. To achieve this purpose, advanced technologies have to be introduced and research works are focusing on the development of such flexible payload architectures.

3.1.1 Different Levels of Required Flexibility

From the satellite operators point of view, the requirements for flexibility can be classified in different levels. As has been described in [16], these levels mainly include:

- Flexibility at orbit location. Orbital relocations can occur during the satellite lifetime and the flexibility to operate the satellite from another orbital location is desired. This level is related to the flexibility on the coverage definition and on the antenna design as well as the frequency plan and the routing, which are presented in the next levels.

- Flexibility at coverage definition. It concerns the shape of the coverage and the number and size of beams in case of multi-beam coverage. The redefintion of a service area, which is related to the antenna reconfigurability, will permit the adaption to the market demands.

- Flexibility at the power allocation. The aim is to assign various power levels to channels to cope with traffic variations. An optimal way to allocate the power resources according to the operational requirements is therefore desired. In order to optimise the power allocation in multibeam satellites, a hybrid genetic algorithm was used in [14]. In this method a sequential application of a Genetic Algorithm (GA) and Simulated Annealing (SA) was performed, where the best solution obtained by the GA together with a random individual are improved by applying SA. The improved solution is then inserted back in the GA that iterates again for some generations. In [57] the authors deal with the optimum power allocation based on traffic demand for multi-beam satellite systems. The problem is modeled as a non-linear convex optimisation problem. The Lagrange duality theory and a heuristic algorithm are used to achieve the final power allocation results.

- Flexibility at the frequency plan. Inside the bandwidth that is allocated to the service and divided in several channels, flexibility applies to the number, the bandwidth and the spacing of the channels. The need for flexibility in reconfiguring the frequency plan, the channel frequencies and bandwidths, in both the uplink and downlink bands, may arise due to different requiremens in the provided services. This flexibility for different channels may require also flexibility at the connectivity and the routing level of the channels, which is ensured by the switch matrices, that are used to route channels from uplink and to downlink beams. For planning an optimal frequency plan, a technique that uses Integer Linear Programming (ILP) with column generation is proposed in [10]. Another optimisation approach that uses ILP formulations together with a greedy heuristic for large-sized problem, was proposed in [43], where the aim of the authors is to maximise the number of users that the system can serve in a service area, while maintaining the possible interference. The channel assignment problem has been also studied in [66] where the proposed approach uses neural network methods. For the same class of problems many heuristic methods have been applied, such as tabu search and genetic algorithms [5].

- Flexibility at the routing level. The fifth level is the flexibility at the connectivity and the routing definition of the channels. The satellite should be able to change routing paths of the channels and functions like full spatial routing (any input channel can be routed to any output), sub-channelization (smaller bandwidth can be selected inside a channel for independent routing) are of interest. This flexibility is currently achieved with the use of reconfigurable switches, that compose the switch matrices, and allow different routings of the channels and interconnectivity between the several beams.

The problem of designing an optimal switch matrix topology that will satisfy the given operational demands is faced by the satellite manufacturers, when constructing the spacecrafts. As the switches are expensive components the target of this optimisation problem is to design a topology that will satisfy all the routing requirements, given a number of channels and amplifiers, while minimising the cost. Research works have tackled this design problem. For example in [20] the authors investigate the design of a switch network capable to route any n inputs to their n outputs, without considering any failure, with the minimum required number of switches in the network. In [22] the authors design low-cost switch matrix topologies considering priorities on the channel inputs, where it should be possible in the network to route p priority channels inputs to the best p quality amplifiers for any set of k faulty and p best quality amplifiers. In some other works the authors derive lower bounds for $N(p, \lambda, k)$, the minimum number of vertices in a tolerant to a restricted number of faults networks where for any choice of at most λ faulty inputs and k faulty outputs, there exist p edge-disjoint paths from the remaining inputs to the remaining outputs [11].

The aforementioned flexibility requirements are summarised in Table 3.1.

Level	Flexibility	Description
1	Orbit location	Same mission from several orbit locations
2	Coverage definition	Shape of coverage / number and size of beams
3	Frequency plan	Number, bandwidth of channels
4	Power	Power allocation based on traffic variation
5	Routing	Channel paths

Table 3.1: Needs for flexibility in communication satellites.

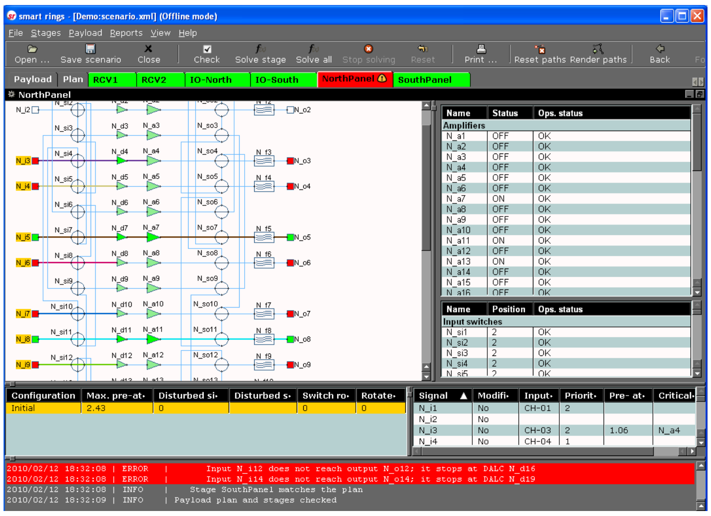
3.2 Flexibility at the Routing Level and Switch Matrix Reconfiguration Optimisation

The focus of this work is related to the flexibility at the connectivity and routing level (fifth level). In current communication satellites, that integrate numerous channels and amplifiers, this flexibility is achieved with the use of large and complex switch matrices that ensure the routing of the different channels and the redundancy in case of failures. As a consequence, from the satellite operators point of view, that is of our interest, given a payload switch matrix topology, the process of finding the best paths for the routing of one or more channels through the switch matrix is a complex problem that requires efficient optimisation techniques. Several technical constraints have to be satisfied while different operational objectives that are of interest for the engineers have to be reached. Only few commercial packages and research works have dealt with this optimisation problem. The following sections detail the commercial softwares that exist on the market for the considered problem as well as the academic works, and analyse their limitations.

3.2.1 Commercial Solutions

Commercial software packages exist for communication satellite payload configuration and reconfiguration like Smartrings [28] and TRECS [1]. As opposed to this work, they consider the full communication payload, and not only the switch matrices. Details about the algorithms and the mathematical models used by these packages are not accessible due to commercial restrictions.

Smartrings applies a generic reconfiguration algorithm to compute all feasible solutions for the defined problem scenario attending to a series of optimisation criteria. Each candidate configuration is presented together with a list of qualifying parameters that caracterise the solution. The payload engineers can rank the solutions that have been generated based on their selected criteria, compare the different configurations based on their qualifying parameters and select the solution that best fulfills the actual operational requirements. The algorithm is controlled by constraints like the number of switches used, the number of the interrupted channels, channels that must not be interrupted or the losses that can be incurred through a channel path. The users can also perform a what-if analysis by defining failure cases, for example on switches or on amplifiers. A graphical editor is used to allow the engineers to create and design a new payload, with the use of a component library. The current payload configuration can be generated from different data sources, such as data from telemetry as well as the operational status file, which is maintained by the payload engineers and contains additional information that may not be included in the telemetry. The combination of both data sources then define the current payload configuration [52]. An example of the provided interface is shown in Figure 3.1.


smart rings - [Demo:scenario.xml] (Offline mode)
File Stages Payload Reports View Help
Open ... Save scenario Close Check Solve stage Solve all Stop solving Reset Print ... Reset paths Render paths Back
Payload Plan RCV1 RCV2 IO-North IO-South NorthPanel SouthPanel
NorthPanel

Name	Status	Ops. status
Amplifiers		
N_a1	OFF	OK
N_a2	OFF	OK
N_a3	OFF	OK
N_a4	OFF	OK
N_a5	OFF	OK
N_a6	OFF	OK
N_a7	ON	OK
N_a8	OFF	OK
N_a9	OFF	OK
N_a10	OFF	OK
N_a11	ON	OK
N_a12	OFF	OK
N_a13	ON	OK
N_a14	OFF	OK
N_a15	OFF	OK
N_a16	OFF	OK

Name	Position	Ops. status
Input switches		
N_si1	2	OK
N_si2	2	OK
N_si3	2	OK
N_si4	2	OK
N_si5	2	OK

Configuration	Max. pre-at‣	Disturbed si‣	Disturbed s‣	Switch ro‣	Rotate‣
Initial	2.43	0		0	0

Signal ▲	Modifi‣	Input‣	Priorit‣	Pre- at‣	Critical‣
N_i1	No	CH-01	2		
N_i2	No				
N_i3	No	CH-03	2	1.06	N_a4
N_i4	No	CH-04	1		

```
2010/02/12 18:32:08 | ERROR    |        Input N_i12 does not reach output N_o12; it stops at DALC N_d16
2010/02/12 18:32:08 | ERROR    |        Input N_i14 does not reach output N_o14; it stops at DALC N_d19
2010/02/12 18:32:08 | INFO     |     Stage SouthPanel matches the plan
2010/02/12 18:32:09 | INFO     | Payload plan and stages checked
```

Figure 3.1: Example of graphical interface of Smartrings [52].

TRECS uses an independent and flexible model definer, that provides a Graphical User Interface (GUI) to edit and construct the different payloads, including a wide variety of payload components. The current configuration of the payload can be loaded into TRECS from a telemetry interface as well as from configuration files. With TRECS one can compare two configurations and any differences are reported. For a problem defined by the user, the algorithm finds all feasible solutions sorted by output signal quality. Furthermore, the algorithm allows the minimisation of the number of changes necessary on the satellite and the insertion of specific constraints like limiting the number of changes to n or considering the first n solutions. The users can also perform a what-if analysis for different cases and each solution is visualised. After the most effective configuration is determined, the tool allows the generation of the corresponding commanding procedure for the reconfiguration of the payload [1, 31]. An example of the provided interface is shown in Figure 3.2.

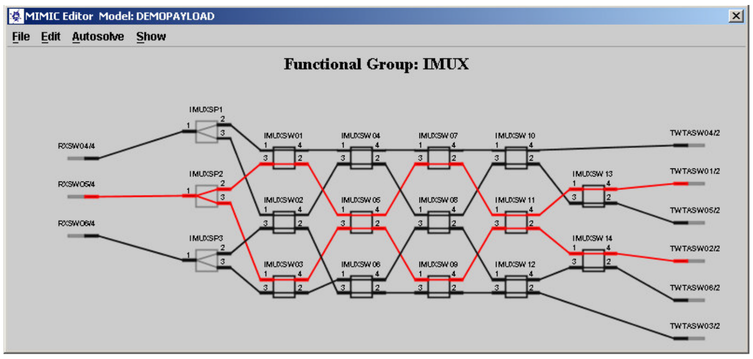


Figure 3.2: Example of graphical interface of TRECS [31].

In these packages, the optimisation is performed either by generating all feasible solutions or by planning the reconfiguration while minimising the number of necessary changes on the satellite. However, computing all feasible solutions implies that additional post-processing is required by the engineers in order to select the best solution. Besides, these tools lack flexibility as they act like black-boxes. The optimisation of new objectives, the use of different solvers or the application of multi-objective exact algorithms, that provide the set of non-dominated solutions based on objective functions defined by the engineers, is not possible. The model cannot be further enhanced, for example including additional constraints, considering new (non-standard) payload components or modelling new objective functions. Another important aspect is that the interaction between these packages and the internally developed tools that are used by the payload engineers is not direct. For instance, the structure of the payload has to be designed using the specific editors that are integrated into these commercial packages, while engineers would prefer to use their internally developed tools. An optimisation framework that will work on top of the operator's software tools, allowing a direct interaction with them, is consequently required.

To summarise, the main limitations of the commercial packages are therefore the following:

- Computation of all feasible solutions or minimisation of the number of necessary changes on the satellite. The optimisation of new objectives or the generation of the Pareto front of solutions for multi-objective problems, based on different objective functions chosen by the engineers is not predicted. This would allow the engineers to choose only among a set of non-dominated solutions the one that better fits their requirements.

- Black Box model. The model is not flexible to allow the insertion of new constraints, objectives or to describe new (possibly non-standard) payload components.

- Black Box algorithm. The algorithm can not be modified in order to use different solvers or to apply multi-objective exact algorithms or heuristics.

- The interaction with the operator's workflow is not direct. The engineers desire to use their own developed softwares and on top of them to add an optimisation framework.

3.2.2 Academic Works

From the academic point of view, only few research works have tackled the considered problem. Recently, a recursive algorithm was proposed to perform a breadth-first-search (BFS) in order to find all possible paths that connect channels to amplifiers [39]. For each channel in the switch matrix, a tree structure is generated that represents all possible combinations of switch connections for each switch in the network. Such a tree is illustrated in Figure 3.3, where $i1$ represents the input channel 1, the node o_i represents a port connect to amplifier i and k_j represents a port connected to another switch.

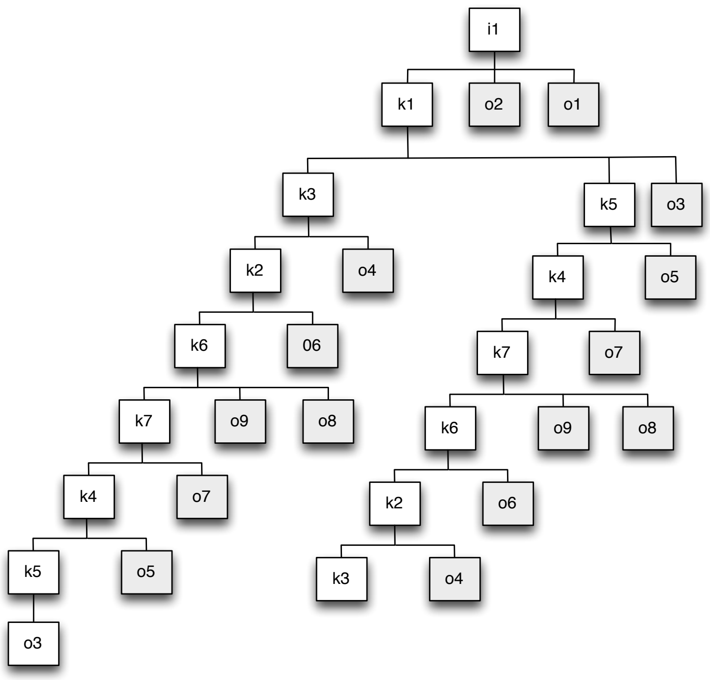


Figure 3.3: Example of a tree structure for one input channel [39].

This tree is searched recursively in order to generate all possible paths. BFS ends when all vertices are traversed. For the above tree structure, 16 possible solutions exist for connecting the single channel. The proposed method is shown to be efficient on small switch networks as it has been tested on a payload structure with 5 channels, 7 switches and 9 amplifiers.

However, it can be expected that for larger problems the BFS algorithm will be limited due to its time complexity, as every vertex and every edge will be explored. The number of feasible solutions increases significantly with the number of switches and the number of channels and amplifiers. As mentioned by the authors, some stopping criteria have to be inserted in the algorithm that can be for example the number of channel interruptions. In current communication payloads like the ones considered in this thesis, switch matrices with up to 100 switches are used. In this case, the set of feasible solutions is significantly large. In addition, computing all feasible solutions is not suitable as with this process solutions that will not be selected by the engineers due to their bad quality (long paths, high losses, numerous switch changes), are computed as well. Besides, from the set of all feasible solutions, that may be too large, additional post-processing is required by the users in order to select the solution that better fits their requirements based on one or more criteria. Therefore, it is more suitable to target on finding the optimal solution according to a well-defined objective function, or a limited set of non-dominated solutions for multi-objective cases.

In a similar work proposed in [60], tree structures are also used for each input channel, to represent all possible paths and possible outputs. The first step of the algorithm consists in making a tree search visiting all nodes of the tree, from the leaves to the root (tree traversal), recovering all paths related to the input channel that has to be connected. The compatibility between the paths is tested in the second step, for example ensuring that two paths do not use the same switch on a different position. Lastly, from the set of all feasible solutions that have been generated, the final configuration is selected based on chosen criteria defined by the users. The approach was tested on a small switch network with 8 switches, 6 channels and 8 amplifiers. Therefore, similarly to the previous work, this approach computes all feasible solutions, and requires post-processing to determine the finally configuration. Finally, both works deal with single objective problem case.

In summary, the main characteristics and limitations of the proposed algorithms for the payload switch matrix configuration and reconfiguration are the following:

- Generation of all feasible solutions. Post-processing is required among the large set of feasible solutions, to select the better one based on the defined criteria.
- Do not deal with multi-objective optimisation.
- BFS algorithm tested on small switch matrices. The algorithm is limited for larger problem sizes due to its time complexity.

3.3 Summary

The satellite operators require different levels of flexibility in order to deal with a dynamic and competitive market. The flexibility at the routing of the channels is currently achieved with large switch matrices that are integrated in the communication payloads. This advantage though, brings to the surface the problem of optimal switch matrix configuration and reconfiguration, which is a complex and time critical process for the engineers.

The existing commercial softwares that deal with the considered problem either generate the set of all feasible solutions or plan reconfigurations while minimising the number of necessary changes on the satellite. Computing all feasible solutions implies that additional post-processing is required by the engineers in order to select the best solution based on their requirements. In addition, these tools lack flexibility as they act like black boxes. The optimisation of new objectives, the use of different solvers or the application of multi-objective exact algorithms, that provide the set of non-dominated solutions based on well-defined objective functions, is not possible. Furthermore, the interaction between these commercial packages and the tools used by the payload engineers for their operations is not direct.

From the academic point of view, only few works deal with the considered problem. Tree structures are used to represent all possible paths for each input channel. Tree search algorithms such as breadth-first-search (BFS) are applied in small switch matrices in order to find all the feasible solutions. For larger problem sizes where the number of feasible solutions increases, it can be expected that these algorithms will be limited due to the time complexity and thus some stopping criteria have to be inserted in the algorithm, such as number of interruptions or the number of switches used. Additionally, post-processing among a large set of feasible solutions is required by the users to select the solution that better fits their requirements based on one or more defined criteria.

In this thesis, as opposed to the existing works, the target is to find the optimal solution or a set of non-dominated solutions for the multi-objective cases, according to well-defined objective functions. An optimisation model is proposed and variants of the model are used for the different operational objectives. Single and multi-objective exact approaches are applied. In addition, we investigate the use of heuristic techniques and hybrid methods, for the large payload problem instances that are not solvable exactly, in order to obtain solutions of good quality very quickly. Lastly, a generic optimisation framework is proposed that embeds the optimisation techniques and allows direct interaction with the tools that are developed and used by the engineers for the payload operations.

Chapter 4

Problem Definition

Contents

The routing of the signals in the payload is achieved through reconfigurable switches organised in switch matrices. These switches can be configured through telecommands sent from the ground stations, i.e. the position of the switches can be changed and each position defines different paths for the routing of the channels. The increasing demands of the market and the modern operational requirements have led to complex payloads with numerous amplifiers and large switch matrices to achieve the required flexibility on the routing of the signals. As a consequence, the process of finding optimal payload switch matrix configurations and reconfigurations during the satellite lifetime, a problem that was previously solved manually, is now becoming a challenging task for the engineers. The problem consists in defining the positions of the all the switches that allow the connection of a given set of channels to the amplifiers, while optimising one or more objectives.

We have proposed a classification of the problem in three related problem cases, namely the *initial configuration*, the *reconfiguration* and the *restoration* case. The initial configuration problem may occur during a planning phase (where the satellite is not operated yet) or while the satellite is operated in orbit (operational phase). The other two problem cases occur during the operational phase. In the following subsections, these problems are presented in details. The operational objectives that are of interest for the engineers are analysed in Section 4.4.

4.1 The Initial Configuration Case

The *initial configuration problem* consists in finding an optimal payload configuration for connecting an initial set of channels in an empty payload, that is without any pre-connected channel paths that carry services to customers. A simplified example of an initial configuration problem is shown in Figure 4.1. It illustrates one possible solution for connecting channel 1 to amplifier 2 and channel 3 to amplifier 4. Channel 1 follows a path composed of 8 links (7 switches used in the path) and channel 3 follows a path composed of 7 links (6 switches used in the path). 11 switches have changed from their initial position.

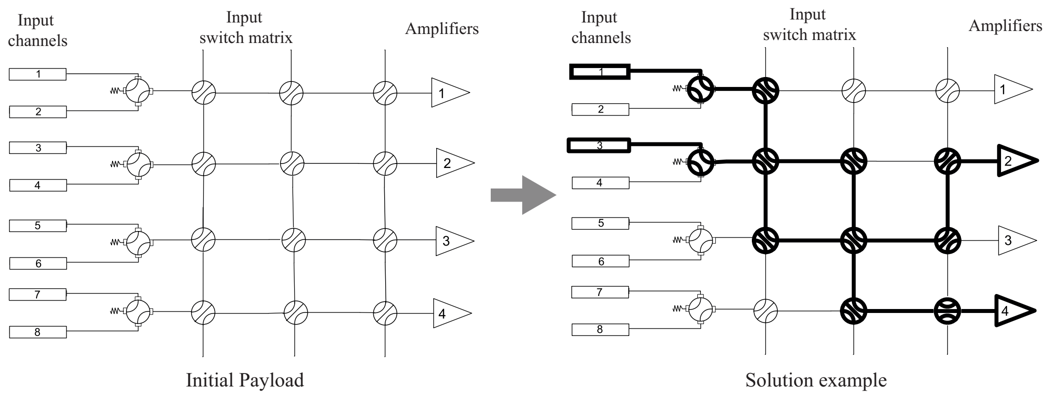


Figure 4.1: Switch matrix simplified example and possible solution for connecting channels 1 and 3 to amplifiers 2 and 4 respectively.

The initial configuration problem can be refined in planning case *(ICpl)* or in operational (in-orbit) case *(ICop)*.

4.1.1 Planning case

In the planning case, the satellite is still on Earth and engineers have to define an initial configuration of the payload based on the channels they want to activate according to their long term business planning. The objective is to set appropriately the initial positions of the switches in order to allow the activation of the planned channels during the satellite lifetime period, without the need of additional reconfiguration. Since the satellite is not in orbit the problem at this stage is not time critical. In addition, some operational objectives, such as the number of switch changes are not of importance yet, as failed switches can still be replaced. The planning case is thus a single objective optimisation problem in which the configuration must be minimum in terms of the length of the channel paths.

4.1.2 Operational case

The initial configuration problem may also occur while the satellite is operated in orbit. This can happen when all the transmitted channels must be replaced by a totally new set of channels, due to some new business needs or technical constraints. In this case, for the new set of channels to connect, the problem becomes multi-objective, since not only the length of the paths has to be minimised but also additional objectives that are of interest while the satellite is in orbit (e.g. switch changes). Contrary to the planning case, finding an optimal solution is a time critical operation. Optimisation approaches should therefore provide a solution in reasonable time, which is set to ten minutes in this work, as defined by the engineers.

4.2 The Reconfiguration Case

The *reconfiguration problem (RC)* occurs when there exists a set of pre-connected channel paths that carry services to customers, and some additional channels must be activated due to new demands. A simple example of such a case is provided in Figure 4.2. On the left hand side, that represents the initial payload configuration, channel 3 is initially connected. In the final payload configuration, channel 6 has to be activated. In the solution example provided on the right hand side of Figure 4.2, channel 3 has been interrupted, i.e. rerouted through an alternative path. Channel 6 has been connected to amplifier 4, whereas amplifier 2 is used by channel 3. The reconfiguration problem case occurs while the satellite is operated in orbit, and as a result, finding an optimal solution to this problem is a time critical process.

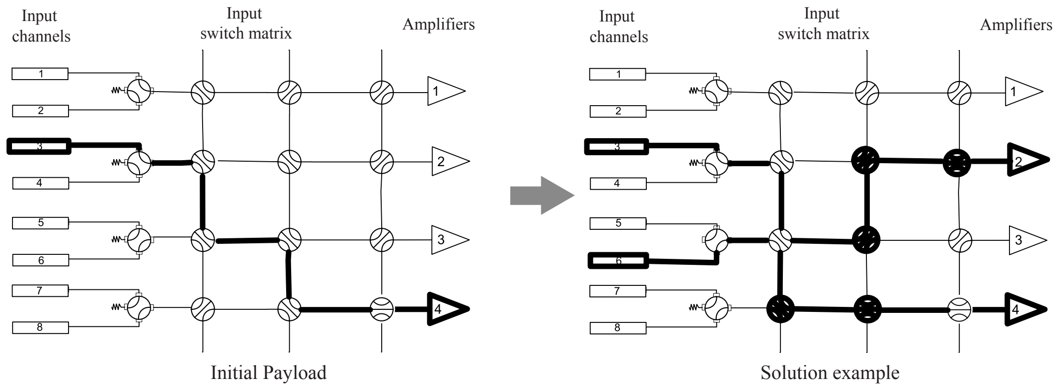


Figure 4.2: Simplified reconfiguration case example and possible solution for connecting channel 6.

4.3 The Restoration Case

The third problem, referred to as the *restoration problem (RS)*, arises when one or more amplifiers and/or switches fail. In that case, the connected channels that are affected by those failures need to be rerouted through alternative paths. A restoration problem case example is shown in Figure 4.3 where a failure in amplifier 4 occurs. In the solution provided on the right hand side of Figure 4.3, channel 3 which was initially connected to the failed amplifier, is rerouted to amplifier 3. None interruption has occurred to the other initially connected channels (channel 1). As the restoration problem case occurs while the satellite is operated in orbit, finding an optimal solution is a time critical operation.

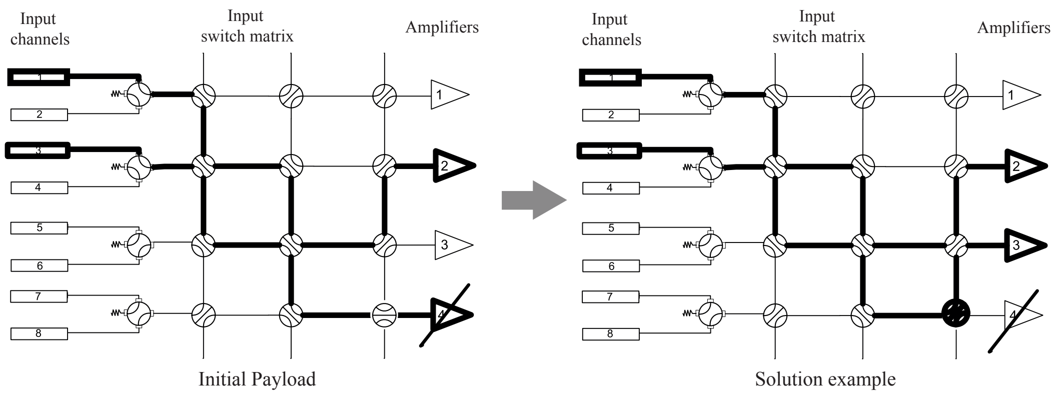


Figure 4.3: Simplified restoration case example and possible solution connecting channels 1 and 3, given the failure of amplifier 4.

4.4 Description of the Operational Objectives

All the aforementioned subproblems have some common and important operational objectives. The problem of finding optimal paths in order to connect a set of channles, i.e. optimal payload switch matrix configurations, comprises several objectives, such as the losses on the signals, the required power or the number of switch changes. An additional objective for the reconfiguration and the restoration problems, is the minimisation of the interruptions of the channels already connected, as this impacts the provided services to the customers. Another objective of high interest for the engineers is the minimisation of the length of the longest channel path (LPL). The formal definition and a more detailed explanation for each of these objectives, are provided in the following subsections of this chapter.

4.4.1 Minimising the Length of the Longest Path

An objective of high interest for the engineers is the minimisation of the length of the longest channel path (LPL). Long paths should be avoided for two reasons. At first, they imply high signal attenuation (losses) on the paths and additionally long paths would reduce the flexibility for future reconfiguration processes (i.e. long paths have higher probability to be interrupted). The problem of minimising the length of the longest path may be time critical, if it occurs during an operational phase, or not if it occurs during a planning phase.

More formally, given n channels to connect, the solution to the problem consists of the set of positions of all the switches used in the switch matrix, that allows the construction of the n paths from each required channel to an amplifier. The objective is to find the solution such

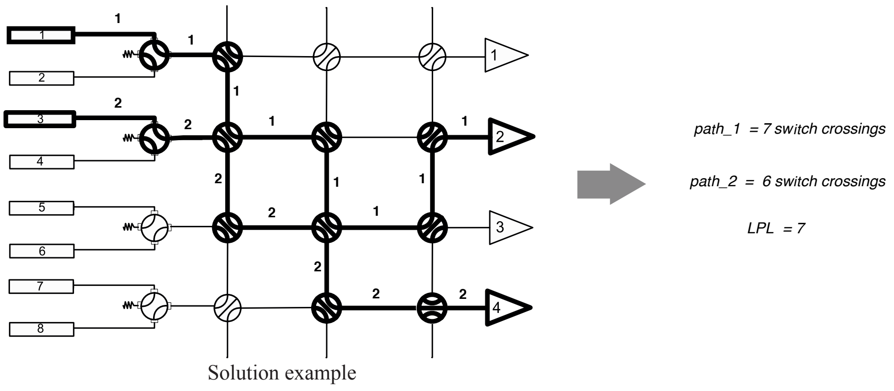


Figure 4.4: Solution example and computation of the length of the longest path

that the length of the longest channel path is the minimum possible. Let k be the number of switches in the switch network and C the set of channels to connect. Let $path_c, c \in C$ be the function returning the length of the channel path c in number of switch crossings or links used by the channel path. The solution vector $p = (pos_1, \ldots, pos_k)$ of size k denotes the position of each switch. The objective is to find a solution p which connects the n channels and minimises:

$$f(p) = \max_{c \in C}\{path_c(p)\} \tag{4.1}$$

An illustration is provided in Figure 4.4, where the first channel follows a path of 7 crossed switches (or 8 links used), and the second channel follows a path with 6 switch crossings (or 7 links used in the path). Thus, in switch crossings the length of the longest path is 7 (or 8 links used).

4.4.2 Minimising the Number of Switch Changes

The number of required switch changes has to be minimised, because changing the position of a switch while the satellite is in orbit may cause its permanent failure. This objective is of interest while the satellite is operated, thus this optimisation problem is time critical.

More formally, given n channels to connect, the solution to the problem consists in finding the set of positions of all the switches used in the switch matrix, that allows the construction of the n paths, from each required channel to an amplifier. Let S of size k be the set of switches in the switch network and C, of size n, the set of channels to connect. Let $pinit_s$ be the initial position of switch $s \in S$. Let $change_s, s \in S$ be a binary variable denoting whether switch s has changed from its initial position or not:

$$change_s = \begin{cases} 1 & \text{if } pinit_s \neq pos_s, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

The solution vector $p = (pos_1, \ldots, pos_k)$ of size k denotes the position of each switch, and the objective is to find a solution p which connects the n channels and minimises:

$$f(p) = \sum_{s \in S} change_s \tag{4.3}$$

As an example, on the right hand side of Figure 4.1, 11 switches have changed from their initial positions. These switches are highlighted with black color.

4.4.3 Minimising the Channel Interruptions

During the in-orbit lifetime of the satellites, additional channels might be activated in the payload, based on commercial considerations, but without impacting other existing services, or at least, minimising the service degradation. As a consequence, the reconfiguration process should try to minimise the number of interruptions of the already connected channels that carry services for customers. An interruption is defined as follows:

Definition 1. *An interruption occurs when the path of a given channel, in the initial configuration, is altered in order to comply with the requirements of the new configuration.*

This objective is of interest for the reconfiguration and restoration problem cases and it is time critical, as it occurs during the operational phase.

More formally, let S be the set of switches, and W be a set of initially connected channels in the payload. Let $path_w, \forall w \in W$, a function returning the set of switches used in path $w \in W$. The variable $interr_w$, denotes whether $path_w$ has been interrupted in the final configuration or not. This variable is therefore defined as follows:

$$interr_w = \begin{cases} 1 & \text{if } \exists s \in path_w \text{ s.t } change_s = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

Given $n \geq |W|$ channels to connect, the solution of the problem consists of the set of positions of all the switches used in the switch matrix, that allows the construction of the n paths from each required channel to an amplifier. The solution vector $p = \{pos_1, \ldots, pos_k\}$ denotes the position of each switch in the final configuration. The objective is to find a solution P which connects the n channels and minimises:

$$f(p) = \sum_{w \in W} inter_w(p) \tag{4.5}$$

On the right hand side of Figure 4.3 where the solution example is displayed, the initially connected channel 3 has been interrupted, as one of the switches used in its path (the one highlighted with the black color) has changed position in the final configuration. Therefore, the number of interruptions in this case is 1.

4.4.4 Minimising the Input Power to Saturation

The maximum performance of each amplifier used in the payload is achieved when it is operated at its saturation point, where the maximum amplification of the input signal is achieved. In order to ensure the highest operational efficiency with the minimum cost, it is desired when configuring the payload, to minimise the required input power to saturation (IPS) for the amplifiers that will be used by the channels. Therefore, the channels have to be routed to the appropriate amplifiers through optimal paths in order to optimise this objective.

For each channel c the choice of the optimal path through the switch matrix that minimises the input power to saturation is influenced by two factors. The first one is the total loss induced by the chosen path, i.e., the loss cumulated from the switches and connectors crossed by a channel c. In addition the loss of each link depends on the channel frequency. Thus, each connector may imply a different loss if it is used by different channels. The loss of each switch is a static parameter independent of the channel frequency. The second factor is the amplifier that the channel c will use. The required power to saturate each amplifier also depends on the frequency. As a result, each amplifier may have a different value of required input power to saturate it, for each different channel. Consequently, both factors should be considered for the calculation of the paths of the final switch matrix configuration. We are here focusing only on the payload switch matrix configuration and we do not consider the front end subsystem of the payload. Thus, we do not include in the optimisation model and in our computations the static parameters, that denote the losses cumulated from the front end part of the payload and the front end gain for each channel.

More formally, let k be the number of switches in the switch network and the set C, of size n, the set of channels to connect. Let S_c be the set of switches crossed by channel $c \in C$ and L_c be the set of links (connectors) crossed by channel $c \in C$. We represent as att_s the losses of the signal from the switch s when it is used by a channel and $att_{l,c}$ the losses of the signal from the link l when is used by channel c. The losses are negative values given in decibels dB. Let:

$$losses_c = \sum_{s \in S_c} att_s + \sum_{l \in L_c} att_{l,c}, \ \forall c \in C \tag{4.6}$$

denote the path losses (i.e. losses on the switches and connectors) of channel c. $ips\_amp_{t,c}$ is the power to saturate the amplifier t that is used by channel c. It is a negative value expressed in dBm. Let thus:

$$ips\_path_c = ips\_amp_{t,c} - losses_c \ \forall c \in C \tag{4.7}$$

denote the computed power of each channel path that is given by the required input power to saturate the amplifier that the channel is connected to minus the losses cumulated by the used links and switches.

The solution vector $P = \{pos_1, \ldots, pos_k\}$ is a vector of size k and denotes the position of each switch. The objective of the problem is thus to find a solution P which connects all n channels and minimises for the sum of attenuation and IPS for all channels for the first case:

$$\sum_{c \in C} (ips\_path_c(p)) \tag{4.8}$$

and minimises the highest sum of attenuation and IPS of one channel for the second case:

$$max_{c \in C}(ips\_path_c(p)) \tag{4.9}$$

As an example, on Figure 4.4, the IPS for amplifier 2 (which is used by channel 1), will be estimated by computing the losses on the links and the switches crossed by channel 1, and the static required power to saturate the amplifier 2, when it is used by the channel 1.

4.5 Summary

The general problem of the satellite payload reconfiguration optimisation was classified in three problem cases, namely the *initial configuration*, which may occur during planning or operational phase, the *reconfiguration* and the *restoration* problem cases. Different operational objectives are of interest for the payload engineers for each of these problem cases. The problem of finding optimal payload switch matrix configurations and reconfigurations may be be time time critical for the engineers. Table 4.1 provides a summary of the proposed classification with the corresponding operational objectives per problem case.

Problem Case	Objective(s) to Optimise	Time Critical
IC_{pl}	LPL	×
	Power	
IC_{op}	LPL	✓
	Changes	
	Power	
RC, RS	LPL	✓
	Changes	
	Power	
	Interruptions	

Table 4.1: Problem Classification

Chapter 5

The Experimental Framework

Contents

The implementation of an experimental framework that will embed the developed optimisation techniques and will interact directly with the operator's internal software tools, is another objective of our work. The following requirements must be satisfied by the framework:

1. Provision of efficient solutions for payload configurations and reconfigurations for all the defined problem cases.
2. Direct interaction with the computerised schematics tools used by the engineers for the payload operations.
3. Applicable to different satellite payload systems.

5.1 The modules of the Experimental Framework

The proposed framework is depicted in Figure 5.1. The computerised schematics (CS) tool makes part of the internal software tools of the operator and is used by the engineers to design the payload structure of the different satellites and to represent the actual configurations. In addition, it contains information about the status of the payload and its components and functionalities to facilitate the different operations that take place during the satellite lifetime. The purpose of the Experimental Framework is to provide optimal or near-optimal solutions to the switch matrix configuration and reconfiguration optimisation problem instances. It enables the application of different single or multi-objective optimisation algorithms. The interaction between the CS tool and the framework, as illustrated in Figure 5.1, implies that the input to the experimental framework is produced by the CS tool and that the output of the experimental framework, that represents the new payload configuration, is loaded to the

CS tool. The new solution configuration can be visualised and be saved as the new actual payload configuration.

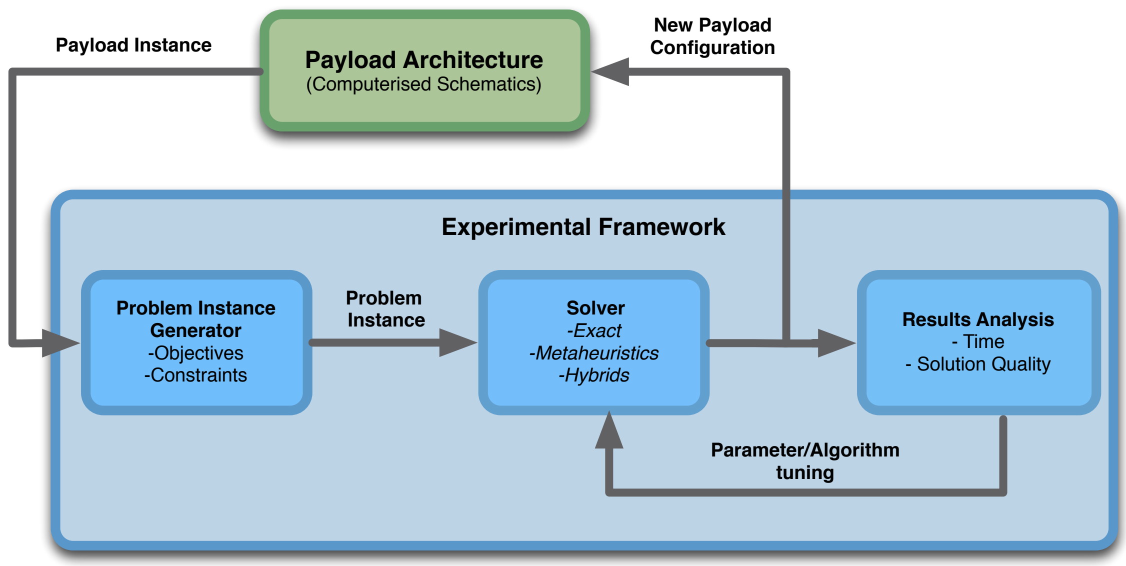


Figure 5.1: The modules of the Experimental Framework.

As illustrated in Figure 5.1, the experimental framework is composed by 3 main modules which are the following:

- The Problem Instance Generator. In this module, the specific optimisation problem is created. For example, within this module, the objective(s) and the constraints of the optimisation problem are defined by the user.
- The Solver. This module provides the final optimal or near-optimal solution(s). Different algorithms may be applied such as single and multi-objective exact methods, metaheuristics or hybrids. The final solution is converted to an appropriate format in order to be loaded to the CS tool.
- The Results Analysis. This module analyses the quality of the generated solutions and the required computational time. If necessary, modifications to the configuration of the solver module can take place based on this analysis. For instance, the algorithm or the parameters used during the solution process can be modified. Thus, this module functions also as an input to the Solver module.

5.2 Interaction with the Operator's Computerised Schematics Tool

The input to the experimental framework is the current payload instance that is generated by the computerised schematics tool in a simple text file. This file describes the current payload configuration and contains all the required information that are needed for the optimisation process. For instance, information about the initial position of all the switches in the payload switch matrix is included, as well as the used amplifiers, the failed amplifiers and switches, the initially connected channels and the status of the channels related to the contract. In addition, all the required data needed for the mathematical computations, for example losses and power values, are contained in the file.

Starting from the current payload instance, as the current configuration, the optimisation problem to solve is defined through the Problem Instance Generator module. In this step, the objective function and other additional constraints, such as the channels to connect or the channels not to interrupt, are determined by the user. The generated optimisation problem is the input to the Solver module which integrates different algorithms that can be used, based on the user selection. The Result Analysis module determines, depending on the quality of the obtained solution or the required computational time, whether modifications should take place in the algorithms or the parameters of the algorithms used. It therefore functions as an input to the Solver module for parameter or algorithms tuning. The final output of the Solver, which consists in the new actual payload configuration, is loaded in the computerised schematics tool.

5.3 Summary

An experimental framework has been proposed and implemented that integrates the different optimisation algorithms and interacts directly with the computerised schematics (CS) tool. The CS tool is used by the payload engineers to describe the current payload configuration and helps them to perform the different payload operations that take place during the satellite lifetime. The experimental framework, that works in conjunction with the CS tool, provides optimal or near-optimal solutions to the defined problem instances, with the application of different optimisation algorithms such as exact methods (single and multi-objective), meta-heuristics or hybrid methods. The solution is converted to an appropriate format and can be loaded to the CS tool in order to be saved as the new actual payload configuration.

Part II

Optimisation with Exact Methods

Chapter 6

Single-Objective Modelling

Contents

The algorithms for solving optimisation problems are mainly classified in exact and in approximate methods. Exact methods obtain optimal solutions but in many cases the required computational time increases significantly with the size of the problem instance. On the other hand, the approximate methods may determine a solution of high quality but there is no guarantee of finding a global optimal solution [64]. Such a classical classification of the optimisation techniques to exact and approximate approaches is illustrated in Figure 6.1 [64].

When considering exact methods, techniques like dynamic programming [21], constraint programming [13] and linear and integer programming based techniques such as branch-and-cut, branch-and-price and branch-and-cut-and-price are widely used [53][45]. Several powerful commercial and non-commercial solvers integrate these techniques to solve exactly integer and mixed integer linear programming optimisation problems in an efficient way. A

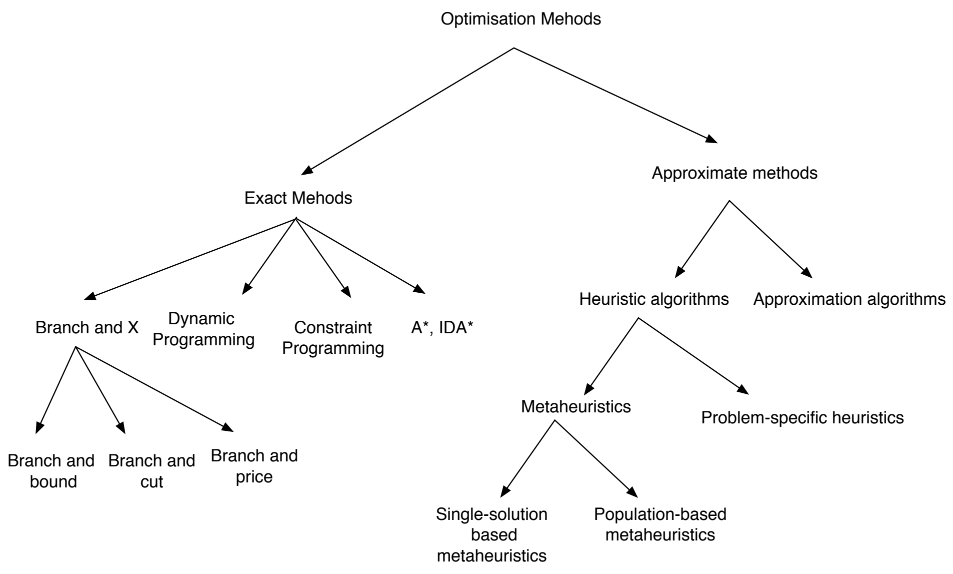


Figure 6.1: Classical classification of optimisation methods.

recent survey on such software is provided in [48] and a comparison of such solvers can be found in [50].

The formulation of the problem and the generation of an optimisation model, is the first step when a solution to an optimisation problem is required. The optimisation model, has three main components which are the objective function, the set of decision variables, and a set of constraints that restrict the values of the decision variables. The solution to the optimisation problem is the set of values of the decision variables for which the objective function reaches the optimal value. Linear programming (LP) is a commonly used model, where the objective function and the set of constraints are linear functions, and the decision variables are continuous. An Integer Linear Program (ILP) is an optimisation problem which involves integer variables. A solution of the following problem is therefore searched:

$$min\{cx \mid Ax \geq b, x \geq 0, x \in \mathbb{Z}^n\} \tag{6.1}$$

where x is the n-dimensional integer variable column vector and $c \in \mathbb{R}^n$. Matrix $A \in \mathbb{R}^{m*n}$ and $b \in \mathbb{R}^m$ define the m constraints of the optimisation problem. Integer and combinatorial optimisation are in details analysed in [53] and [74]. A Mixed Integer Linear Program (MILP) involves a combination of integer and real-valued variables.

A novel ILP optimisation model for the payload switch matrix configuration and reconfiguration problem is proposed in this thesis. Similarities of this problem with the well-known integer network flow problems exist, as several channels (flows) are crossing simultaneously the network, with the aim to find some optimal paths with respect to some objectives. However, the direct application of such well-know network models is not possible, due to the specific properties of the considered problem.

This chapter provides at first a general overview of the classical network flow models, mentioning the similarities and the differences with the problem tackled in this thesis. In

Section 6.2 the proposed ILP model is presented in details for each of the considered operational objectives. The experimental results for the single-objective optimisation problems, using an exact solver, are analysed in Section 6.4.

6.1 Classical Network Flow Problems

Network flow problems have been widely studied in the literature as they occur in many practical applications of different domains, such as telecommunications, engineering and transportation systems [8]. Problems like shortest path, minimum cost, maximum flow, and multicommodity flow problems, constitute the most common classes of practical optimisation problems [23]. Information about this kind of problems can be found for example in [7, 15].

Multicommodity network flow problems involve several flow types (or commodities), that simultaneously use the network. The objective is to flow the commodities through the network, without exceeding the capacities of the edges, while minimising the cost. At the most general level, as indicated in [55], the multicommodity flow formulation can take the following form:

$$\text{minimise} \sum_{l \in L} D_l(f_l) \tag{6.2}$$

$$\text{subject to flow conservation constraints} \tag{6.3}$$

$$\text{plus any additional special constraints,} \tag{6.4}$$

where f_l denotes the total flow on link l, D_l the cost function and L is the set of links (edges) in the network. The flow conservation constraints ensure that the flow that enters into a node equals the amount of flow out of it, unless the node is a source or a sink. The integer multicommodity flow (IMCF) problem results when the flow of a commodity may use only one path from origin to destination [17]. The IMCF problem formulation is defined over the network G comprised of node set N and arc set A. Binary decision variables x are included, where x_{ij}^k equals 1 if the entire quantity of commodity k is assigned to arc ij, and equals 0 otherwise. Apart from the flow conservation constraints, the capacity constraints of the optimisation problem, ensure that the flow along an edge should not exceed the capacity of the edge. A path-based formulation is also used for modelling this type of problems, but in that case an exponential number of variables is needed [62]. A detailed formulation of the IMCF problem can be found for example in [17]. In the field of telecommunications, these models are widely used for optimal routing and network design [51, 71]. Such a mathematical model formulation was also proposed for Clos Type symmetrical networks based on identical square crossbar switches for three, five and seven stages [41]. The proposed algorithms for solving the general integral multicommodity network flow problem are based on branch-and-cut and branch-and-price techniques [18, 17, 72, 26] as well as column generation [55]. The maximum integer multicommodity flow consists of sending distinct integer flows between k given pairs of terminal nodes of an undirected graph with edge capacities, with the aim to maximise the total amount of flow sent. A special case arises when all edge capacities are set to 1 and the problem consists of finding the maximum number of edge-disjoint paths between the k terminals [25, 29].

Similarities between the above mentioned cases, and the problem considered in this thesis exist. Different channels use simultaneously the network. Different costs (losses) may also be associated to each link when it is used by a channel. Each channel uses a single path from the origin to its destination and each link can be used only by one channel. The conservation constraints for each switch are by default satisfied (what enters in a switch node goes out of it). However, such optimisation models can not be directly applied to the considered problem, due to the specific properties of the payload switch matrix. The key difference is the notion of the switch position. The classical models are defined over a static network topology, but in the switch matrix when one switch changes position the topology of the network is modified, which means that some edges are not available anymore while new edges are formed. Consequently, the flow propagation constraints for each position of each switch have to be defined. When a channel crosses one switch, these constraints will describe which links are available, based on the switch position. Therefore, the flow propagation constraints will ensure the valid propagation of the signals through the switch network based on the switch positions. Conservation constraints have to be added for the amplifiers, if both input and output switch matrices are used. Furthermore, there may exist amplifiers that can not be used by any channel (failed amplifiers) or that can be used only by a specific subset of channels. The solution vector of the problem is the set of positions of the switches, that allows the connection of a given set of channels while optimising the defined objective(s).

The main differences between the classical network flow problems and the payload switch matrix optimisation problem are summarised in Table 6.1.

	Classical Network Flow Problems	Payload Switch Matrix Problem
Constraints	Flow conservation	Flow propagation
Network Topology	Static	Dynamic
Solution representation	Flows/Paths	Switch Positions

Table 6.1: Main differences between classical network flow problems and payload switch matrix optimisation problem.

Therefore, respecting the above mentioned specificities, a novel mathematical optimisation model is developed for the payload switch matrix configuration and reconfiguration problem in order to optimise the defined objectives. The mathematical model is detailed in the following section.

6.2 An ILP Model for Payload Switch Matrix Configuration and Reconfiguration

The basic variables of the proposed model are the binary flow variables that are associated to each link and the integer position variables that are associated to each switch. Distinct flow values are assigned to each link that is neighbour to each channel that has to be connected. These flow values are propagated through the switch network, according to the flow propagation constraints that are defined for each component of the payload, e.g. for each switch. The capacity of each link is equal to one, i.e. a physical link can be used only by one channel.

More formally, the proposed model consists of the following constants, sets, variables and constraints.
Constants and Sets:

Let define:

- q as the number of channels to be connected.
- n as the number of states (positions) of the switches of the network (e.g. if R switches with 4 states are used, n=4).
- P as a set of size n, with integer values from 0 to $n-1$ representing the maximum number of positions a switch can have. The position is relative. It refers to the number of steps the switch should take from its current state to reach the new state.
- S as the set of all switches.
- T as the set of all amplifiers.
- C as the set of all channels.
- $L = CL \cup TL \cup SL$ as the set of all links. L consists of the following subsets:
  - CL as the set of all links connected to the channels. As only the input switch matrix is considered it holds $CL = CL_{in}$, where CL_{in} of size $|C|$, the set of all links neighbouring with the channel filters on the input switch matrix. Let cl_{in_c} be the link connected with channel c.
  - TL as the set of all links connected to amplifiers. As only the input switch matrix is considered $TL = TL_{in}$, with TL_{in} of size $|T|$, the set of input links of all amplifiers. Let tl_{in_t} be the input link of amplifier t.
  - SL as the set containing the links between any two switches.
- l_o as a special link, or link 0, when signal can not be propagated. For instance in Figure 6.2 with a switch in state 2, link a is connected with l_0.
- Let $C_{conn} \subseteq C$, of size q, be the set of channels to connect. We represent each channel to connect with an integer value. Thus, let the elements of this set to be $1, 2, \ldots, q$, representing the $1^{st}, 2^{nd}, \ldots q^{th}$ channel to connect respectively.
- $CL_{conn} \subseteq CL_{in}$ of size q, as the set of links neighbouring with the input channels that will be connected. Let the elements of this set to be $l_1, l_2, \ldots, l_q$, indicating that the neighbouring link of the channel to connect c_1 is l_1, the neighbouring link of the channel to connect c_2 is l_2 etc.
- The matrix M of size $|S| * |P| * |L| * |L \cup \{l_0\}|$, describes the links that are connected for all possible positions of all switches:

$$m_{s,p,l_i,l_j} = \begin{cases} 1 & \text{if } l_i \text{ is connected with } l_j \text{ via switch s at position p,} \\ 0 & \text{otherwise.} \end{cases}$$

Variables

- Let define *Pos* of size $|S|$, as the solution vector, providing the position for each switch. The initial state of each switch is considered as $pos_s = 0$. For instance, if an R-type switch (with 4 states) is initially in state 3, and in the solution $pos_s = 1$, then the switch will have to move to state 4.
- Integer vector *Flow* of size $|L|$, showing the flow value (from 0 to q) that is carried by each link. It follows:
$$flow_l = \begin{cases} x & \text{with } 0 < x \leq q, x \in \mathbb{Z} \quad \text{if link } l \text{ is used by the } x^{th} \text{ channel to connect,} \\ 0 & \text{otherwise.} \end{cases}$$
- Binary vector B of size $|S| * |P|$, is used to activate or de-active the flow propagation constraints, such that:
$$b_{s,p} = \begin{cases} 1 & \text{if } pos_s = p, \\ 0 & \text{otherwise.} \end{cases}$$
- Boolean vector *Ampused* of size $|T|$, indicating whether an amplifier is active (used) or not.
- Binary variable $flow_{l,c}$, $\forall l \in L$, $\forall c \in C_{conn}$, indicating whether link l is used by channel c or not:
$$flow_{l,c} = \begin{cases} 1 & \text{if } l \text{ is used by channel } c, \\ 0 & \text{otherwise.} \end{cases}$$

Constraints

- To start the flow distribution, flow values are assigned to each link l_i from the set CL_{conn} that will carry the flow value of the i^{th} channel:
$$flow_{l_i,i} = 1, \quad \forall l_i \in CL_{conn}.$$
- The integer variable $flow_l$ must be equal to:
$$flow_l = \sum_{c \in C_{conn}} c * flow_{l,c}, \quad \forall l \in L.$$
- To avoid flow paths starting from an input channel and returning to another input channel, flow values must be set to 0 for all unused input channels:
$$flow_{l_i} = 0, \quad \forall l_i \in \{\{CL_{in}\} - \{CL_{conn}\}\}.$$
- To ensure that a link is either not used or used by only a single channel, the following constraint is added:
$$\sum_{c \in C_{conn}} flow_{l,c} \leq 1, \quad \forall l \in L.$$

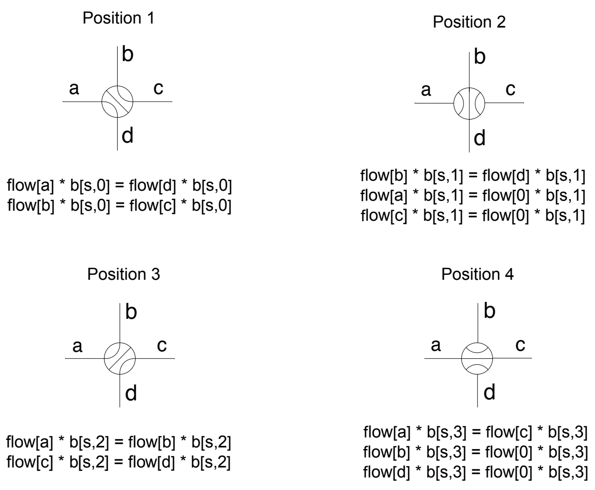

Position 1
b
a
c
d
flow[a] * b[s,0] = flow[d] * b[s,0]
flow[b] * b[s,0] = flow[c] * b[s,0]
Position 2
b
a
c
d
flow[b] * b[s,1] = flow[d] * b[s,1]
flow[a] * b[s,1] = flow[0] * b[s,1]
flow[c] * b[s,1] = flow[0] * b[s,1]
Position 3
b
a
c
d
flow[a] * b[s,2] = flow[b] * b[s,2]
flow[c] * b[s,2] = flow[d] * b[s,2]
Position 4
b
a
c
d
flow[a] * b[s,3] = flow[c] * b[s,3]
flow[b] * b[s,3] = flow[0] * b[s,3]
flow[d] * b[s,3] = flow[0] * b[s,3]

Figure 6.2: Positions of a R-type switch and the corresponding non-linear expression of the flow propagation constraints. The initial position of the switch is position 1.

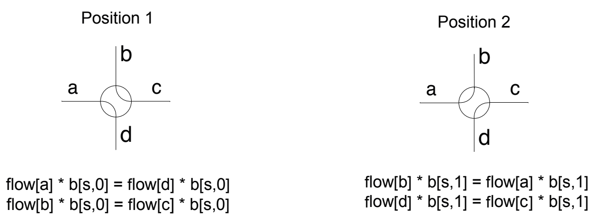

Position 1
b
a
c
d
flow[a] * b[s,0] = flow[d] * b[s,0]
flow[b] * b[s,0] = flow[c] * b[s,0]
Position 2
b
a
c
d
flow[b] * b[s,1] = flow[a] * b[s,1]
flow[d] * b[s,1] = flow[c] * b[s,1]

Figure 6.3: Positions of a C-type switch and the corresponding non-linear expression of the flow propagation constraints. The initial position of the switch is position 1.

- The flow propagation constraints are expressed for each position of each switch and describe the established connections at each case. It is ensured that based on each switch position, the connected links have the same flow value. Figure 6.2 illustrates the non-linear expressions for the flow propagation constraints for all states of an R-type switch. The corresponding constraints for C and T switch types are illustrated in Figures 6.3 and 6.4 respectively. The linear equivalent constraints can be expressed as follows:

$$flow_{l1} + b_{s,p} * q - q \leq flow_{l2} \leq flow_{l1} - b_{s,p} * q + q :$$
$$\forall s \in S, \forall p \in P, \forall l_1, l_2 \in (L \cup \{l_0\})^2, \text{ such as } m_{s,p,l_1,l_2} = 1.$$

- The following constraint ensures that only one position is selected for each switch:

$$\sum_{p \in P} b_{s,p} = 1, \quad \forall s \in S.$$

$$pos_s = \sum_{p \in P} p * b_{s,p}, \quad \forall s \in S.$$

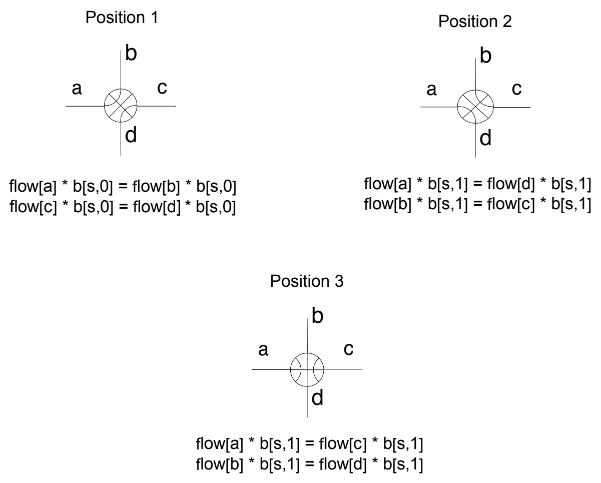

Position 1
b
a
c
d
flow[a] * b[s,0] = flow[b] * b[s,0]
flow[c] * b[s,0] = flow[d] * b[s,0]
Position 2
b
a
c
d
flow[a] * b[s,1] = flow[d] * b[s,1]
flow[b] * b[s,1] = flow[c] * b[s,1]
Position 3
b
a
c
d
flow[a] * b[s,1] = flow[c] * b[s,1]
flow[b] * b[s,1] = flow[d] * b[s,1]

Figure 6.4: Positions of the T-type switch and the corresponding non-linear expression of the flow propagation constraints. The initial position of the switch is position 1.

- An amplifier is used if its input link has a positive flow value. The number of active amplifiers has to be equal to the number of channels to connect:

$$ampused_t * q \geq flow_{tl_{in_t}}, \quad \forall t \in T.$$

$$\sum_{t \in T} ampused_t = q.$$

- Link $\{l_0\}$ never carries any signal.

$$flow_{l_0} = 0.$$

The complete version of the ILP model is provided below:

Complete ILP model

$Variables:$	
$pos_s \in \mathbb{Z}$	$\forall s \in S$
$flow_l \in \mathbb{Z}$	$\forall l \in L \cup \{l_o\}$
$b_{s,p} \in \{0;1\}$	$\forall s \in S, \forall p \in P$
$ampused_t \in \{0;1\}$	$\forall t \in T$
$flow_{l,c} \in \{0;1\}$	$\forall l \in L, \forall c \in C_{conn}$
$Constraints:$	
$flow_{l_i,i} = 1$	$\forall l_i \in CL_{conn}$
$flow_{l_i} = 0$	$\forall l_i \in \{\{CL_{in}\} - \{CL_{conn}\}\}$
$flow_{l1} + b_{s,p} * q - q \leq flow_{l2} \leq flow_{l1} - b_{s,p} * q + q$	$\forall s \in S, \forall p \in P, \forall (l_1, l_2) \in (L \cup \{l_0\})^2$, s.t. $m_{s,p,l_1,l_2} = 1$.
$\sum\limits_{p \in P} b_{s,p} = 1$	$\forall s \in S$
$pos_s = \sum\limits_{p \in P} p * b_{s,p}$	$\forall s \in S$
$ampused_t * q \geq flow_{tl_{in_t}}$	$\forall t \in T$
$\sum\limits_{t \in T} ampused_t = q$	
$\sum\limits_{c \in C_{conn}} flow_{l,c} \leq 1$	$\forall l \in L$
$flow_l = \sum\limits_{c \in C_{conn}} c * flow_{l,c}$	$\forall l \in L$
$flow_{l_0} = 0$	

6.3 Modelling the Operational Objectives

The proposed model presented in Section 6.2, includes the variables and the necessary constraints to ensure valid solutions to the payload configuration and reconfiguration problem. The model can be easily extended to formulate all the operational objectives that are of interest for the engineers. For each of the objectives that have been detailed in Section 4.4, the proposed variants of the optimisation model are here detailed.

6.3.1 Minimising the Length of the Longest Path

To minimise the length of the longest path, decision variables that denote the length of each channel path are inserted in the optimisation model. More precisely, the following variables, constraints and objectives, have to be added to the ILP model presented in Section 6.2, in order to minimise the length of the longest channel path.

Variables

- An integer variable $path_c$, $\forall c \in C_{conn}$, is introduced that denotes the length of the path of channel c.
- Integer variable z indicating the length of the longest channel path.

Constraints

- The length of the path of each channel must be equal to the sum of all links that carry the channel:

$$path_c = \sum_{l \in L} flow_{l,c}, \quad \forall c \in C_{conn}.$$

- The length of each path must be smaller or equal than the length of the longest path:

$$path_c \leq z, \forall c \in C_{conn}$$

Objectives

- The objective is to minimise the length of the longest channel path:

$$Min \quad z$$

The complete ILP model for minimising LPL is provided in Appendix 1.

6.3.2 Minimising the Number of Switch Changes

To minimise the number of switch changes, decision variables that indicate wether a switch changes position or not, have to be included in the optimisation model. More precisely, the following variables, constraints and objectives, have to be added to the ILP model presented in Section 6.2.

Variables

- Binary vector *Change* of size $|S|$, indicating whether a switch needs to change from its initial state or not.

$$change_s = \begin{cases} 1 & \text{if } pos_s > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Constraints

- We ensure that $change_s = 1$, if and only if $pos_s > 0$:

$$change_s * n \geq pos_s, \quad \forall s \in S.$$

$$change_s \leq pos_s, \quad \forall s \in S.$$

Objectives

$$Min \sum_{s \in S} change_s$$

The complete ILP model for minimising the number of switch changes is displayed in Appendix 1.

6.3.3 Minimising the Channel Interruptions

To minimise the number of channel interruptions, a pre-processing step is necessary. In this step, the set of switches that are used by each initially connected channel has to be extracted. In the final solution, as long as one of these switches has changed position, the corresponding channels have been interrupted. More precisely, the following constants, variables and constraints have to be added in the optimisation model described in Section 6.2.

Constants and sets
Let define:

- Let Q_{init} be the set of initially connected channels. Let the elements of this set to be $1, 2, \ldots, q_{init}$.
- Let $q_{final} \geq q_{init}$, representing the total number of channels to be connected at the final configuration.
- The set $S_q \subseteq S$ be the set of switches used by path q, $\forall q \in Q_{init}$:
- In this case the output payload switch matrix is considered as well. Let thus $tl_{out_t}, \forall t \in T$, be the output link of amplifier $t \in T$ and $cl_{out_c}, \forall c \in C$, be the neighbor link of channel $c \in C$ in the output switch matrix.

Variables

- Let a binary variable $interr_q$, defined $\forall q \in Q_{init}$, denoting wether initial path q has been interrupted or not. The path is interrupted as long as one of its used switches has changed positions. It thus holds:

$$interr_q = \begin{cases} 1 & \text{if } \sum_{s \in S_q} change_s > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Constraints

- The following constraint ensures that a path has been interrupted as long as at least one of the switches it crossed has changed position:

$$interr_q \geq change_s, \forall s \in S_q, \forall q \in Q_{init}$$

- Since the output switch matrix is used, the flow propagation constraints must be also defined for each amplifier:

$$flow_{tlin_t} = flow_{tlout_t}, \forall t \in T$$

- Additionally, since the output switch matrix is here used, the flow value on the input channel must be equal to the flow value of the neighbor link of the corresponding output channel:

$$flow_{clin_c} = flow_{clout_c}, \forall c \in C$$

Objective

- The objective is to minimise the number of channel interruptions. Therefore:

$$Min \sum_{q \in Q_{init}} interr_q$$

The complete ILP model for minimising the number of channel interruptions can be found in Appendix 1.

6.3.4 Minimising the Input Power to Saturation

To minimise the Input Power to Saturation (IPS), different parameters have to be included in the mathematical model that provide the losses cumulated from the switches and the links crossed by the signals, as well as the input power to saturate each amplifier for each channel. This research focuses only on the payload switch matrix configuration and does not consider the front end subsystem of the payload. As a result, the static parameters, that denote the losses cumulated from the front end part of the payload and the front end gain for each channel, are not included in the optimisation model and are not considered in the computations.

In the first problem case the total IPS is minimised, whereas in the second problem case the objective is the minimisation of the highest IPS. The following parameters, sets, variables and constraints have to be added to the optimisation model.

Constants, Sets and Parameters

- Let $att_{l,c}$, $\forall l \in L, \forall c \in C$ a parameter specifying the loss in the signal when the link $l \in L$ is used by channel $c \in C$.
- Let att_s, $\forall s \in S$ a parameter specifying the loss in the signal when switch $s \in S$ is used.
- Let $IP_{t,c}$, $\forall t \in T, \forall c \in C$ a parameter specifying the required power to saturation of amplifier $t \in T$ for channel $c \in C$.
- Finally, let L_s, $\forall s \in S$, to represent the set of the 4 links that are neighboring to the switch $s \in S$.

Variables

- Binary vector $Twused_{t,c}$ of size $|T| * |C_{conn}|$, indicating whether amplifier $t \in T$ is used by channel $c \in C_{conn}$ or not. Thus, it will hold that:

$$twused_{t,c} = \begin{cases} 1 & \text{if } t \text{ is used by channel } c, \\ 0 & \text{otherwise.} \end{cases}$$

- Binary vector $used\_sw_{s,c}$ of size $|S| * |C_{conn}|$, indicating whether switch $s \in S$ is used by channel $c \in C_{conn}$ or not. Thus, it will hold that:

$$used\_sw_{s,c} = \begin{cases} 1 & \text{if } s \text{ is used by channel } c, \\ 0 & \text{otherwise.} \end{cases}$$

- The variable ips_path_c, $\forall c \in C_{conn}$, indicates the input power to saturation corresponding to the channel c.
- The variable s denotes the maximum required input power to saturation among all the channel paths.

Constraints

- The binary variable $twused_{t,c}$ will be 1 if the input link of amplifier $t \in T$ is used by channel $c \in C_{conn}$.

$$twused_{t,c} \geq flow_{tlin_{t,c}}.$$

- The variable $twused_{t,c}$ will be 0 if the amplifier is not used:

$$ampused_t = \sum_{c \in C_{conn}} twused_{t,c}, \forall t \in T.$$

- The following constraints ensure that a switch is used by a channel if the channel is using any of its neighbor links:

$$4 * used\_sw_{s,c} \geq \sum_{l \in L_s} flow_{l,c}, \quad \forall s \in S, \forall c \in C_{conn}.$$

$$used\_sw_{s,c} \leq \sum_{l \in L_s} flow_{l,c}, \quad \forall s \in S, \forall c \in C_{conn}.$$

- The computed power of each channel path is given by the required input power to saturate the amplifier that the channel is connected to minus the losses cumulated by the used links and switches:

$$ips\_path_c = -\sum_{l \in L} att_{l,c} * flow_{l,c} - \sum_{s \in S} att_s * used\_sw_{s,c} +$$

$$\sum_{t \in T} ips_{t,c} * twused_{t,c}, \quad \forall c \in C_{conn}.$$

- The variable s represents the highest input power to saturation among all the channel paths:

$$s \geq ips\_path_c, \forall c \in C_{conn}.$$

Objective

- The objective of the first approach is to minimize the total required power:

$$Min \sum_{c \in C_{conn}} ips\_path_c$$

- The objective of the second approach is to minimize the highest required power:

$$Min \quad s$$

The complete ILP optimisation model for minimising the Input Power to Saturation (IPS) is provided in Appendix 1.

6.4 Experimental Results

This section presents the experimental results for the single-objective optimisation problems, for each of the defined operational objectives. At first, information about the experimental setup and the considered payload instances is provided and then the numerical results are analysed.

6.4.1 Experimental Setup

All the experiments were performed using the HPC platform facilities of the University of Luxembourg [70] on an homogenous type of nodes with Xeon 6C, 2,26GHz on a single core with 2GB of RAM. The operating system was Debian GNU/Linux 6.0.9 and kernel 3.2.0-0.bpo.4-amd64. No core affinity was set. A single CPU core was used in order to create a benchmark that will be used for the comparisons of the experimental results. The exact solver used was IBM ILOG CPLEX 12.0.0 [2]. In case of modifications on the above experimental setup for specific experiments, these are detailed in the corresponding subsection of the experimental results. Information about the payload instances tackled is provided in Table 6.2.

	Small Payload	Large payload
Number of switches / Type	50 / R	100 / R
Number of amplifiers	23	35
Maximum nb. of channels to connect	23	35
Nb. of channels to connect	8, 13, 18, 23	
Nb. of channels sets per size	30	

Table 6.2: Payload instances.

As denoted in this table, two realistic payload switch matrices were used, one with 50 switches of type R, with up to 23 amplifiers and another one with 100 switches of type R and up to 35 amplifiers. The number of amplifiers indicates the maximum number of channels that can be connected. 30 different sets of channels to connect of size 8, 13, 18 and 23 were uniformly chosen at random. An instantiation of the experimental framework is illustrated in Figure 6.5.

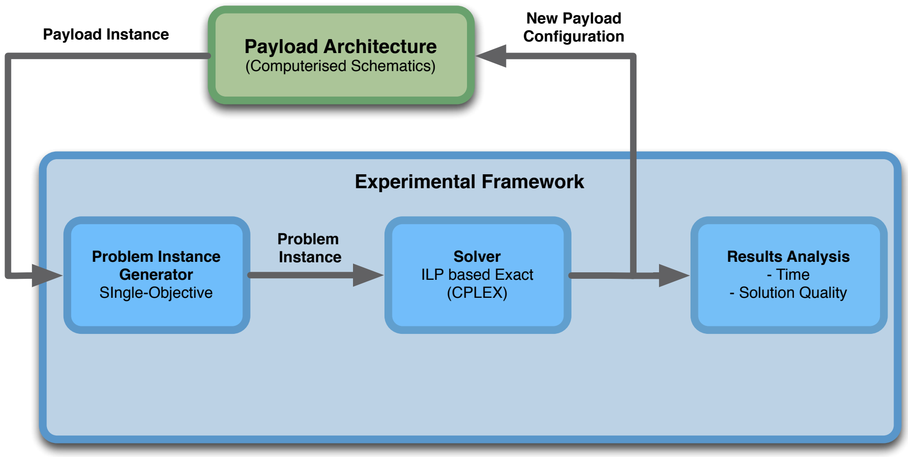


Figure 6.5: Instantiation of the experimental framework for single-objective optimisation using exact methods.

6.4.2 Minimising the Longest Path Length

The problem of minimising LPL can arise either during the operational phase or during the planning phase. Thus, two termination conditions have been considered for these experiments. The first one was set to 120 hours, defined by technical reasons. This permits at first to assess if the considered instances are solvable exactly, and if so, to analyse the upper bound of the required computational time. The second termination condition was set to 10 minutes which is defined by the engineers, and is used when the problem occurs during the operational phase.

6.4.2.1 50 switches payload

The results when minimising LPL on the 50 switches payload are provided in Table 6.3, for each of the above mentioned termination conditions. At first, the hit rate is displayed, which denotes the percentage of the instances solved exactly. Besides, the average fitness value, the average computational time as well as the minimum and the maximum required computational time in seconds are provided.

The first remark is that when minimising LPL, not only instances could be solved exactly, even after using the first termination condition of 120 hours. When connecting 8 channels, 93.33% of the instances were solved whereas for the sets of 18 and 23 channels the hit rate is 56.66%. With the termination condition of ten minutes, the hit rate is significantly lower. 90% of the instances were solved when 8 channels are connected, whereas for the sets of 18 channels to connect the hit rate drops to 6.66%. Finally, none instance could be solved when connecting 23 channels.

A high standard deviation of the required computational time is observed in all instances that indicates the influence of the set of channels to connect. As an example, when 8 channels are connected, the average required time is 6013.36 seconds with maximum time among the solved instances 168288.7 seconds and minimum time only 0.19 seconds. When 13 channels

	Channels	Hitrate	Avg. Fitness	Avg. Time	Min Time	Max Time
120 hours	8	93.33%	2.85 $\pm$ 0.52	6013.36 $\pm$ 31802.98	0.19	168288.7
	13	36.66%	3.63 $\pm$ 0.50	42564.75 $\pm$ 73931.85	1.02	253018.9
	18	56.66%	3.94 $\pm$ 0.42	46213.47 $\pm$ 47202.44	1.61	135928.7
	23	56.66%	4.94 $\pm$ 0.24	32371.45 $\pm$ 36075.48	1549.16	116771.1
10 min	8	90%	2.81 $\pm$ 0.48	3.19$\pm$12.89	0.19	67.63
	13	13.33%	3 $\pm$ 0	10.67$\pm$13.64	1.02	30.57
	18	6.66%	3 $\pm$ 0	4.52$\pm$4.12	1.61	7.44
	23	0%	–	–	–	–

Table 6.3: Minimising LPL on 50 switches payload.

are connected, the standard deviation of the time reaches 73931.85 seconds. As previously mentioned, the switch matrix, which is designed in order to fulfill the operational requirements with the minimum cost, is not symmetric and the channels can be placed at different locations. Therefore, based on the selected set of channels to connect, some instances may be solved easier compared to others. Figure 6.6 illustrates the average required time for the set of instances that had the worst fitness value (highest LPL) and the set of instances with the best fitness values (lowest LPL) considering the 50 switches payload and the termination condition of 120 hours. It can be observed that the instances with the worst fitness are the most difficult to solve while the instances solved in the shortest time have smaller fitness values. For instance, for the case of 13 channels to connect, the instances with the best fitness were solved in average after 8.12 seconds where the solved instances with the worst fitness required in average 66883 seconds.

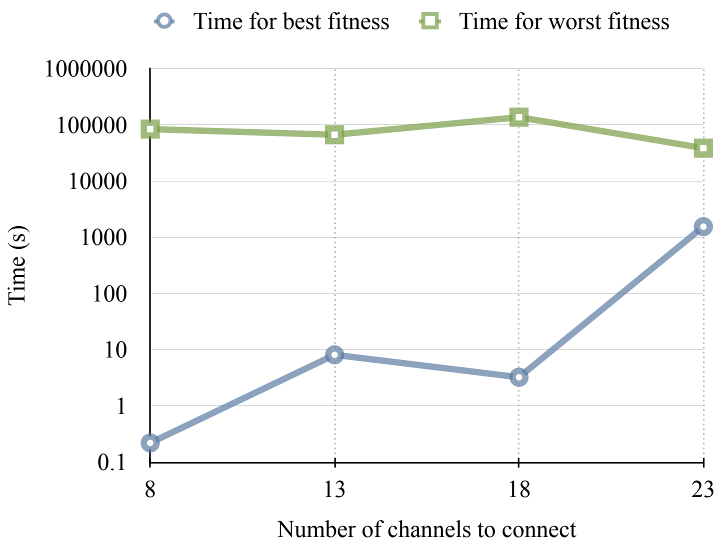


Figure 6.6: 50 switches payload. Average time (in seconds) of the set of instances with the worst (maximum) and best (minimum) fitness.

It should be observed that for the case of 8 channels to connect, the maximum time among the solved 90% of the instances, when considering the ten minutes termination condition, is only 67.63 seconds, and the median value is very low, i.e 0.34 seconds. Only one additional instance was solved within the 120 hours termination condition, after 168288.7 seconds. As can be seen in the boxplot provided in Figure 6.7, that visualises the time distribution on the 50 switches payload, few outliers represent the instances that required the higher computational time while the median values remain low in most cases. More precisely, the median values are 25555.57 seconds for 13 channels to connect and 17217.96 and 14440.51 for 18 and 23 channels to connect respectively.

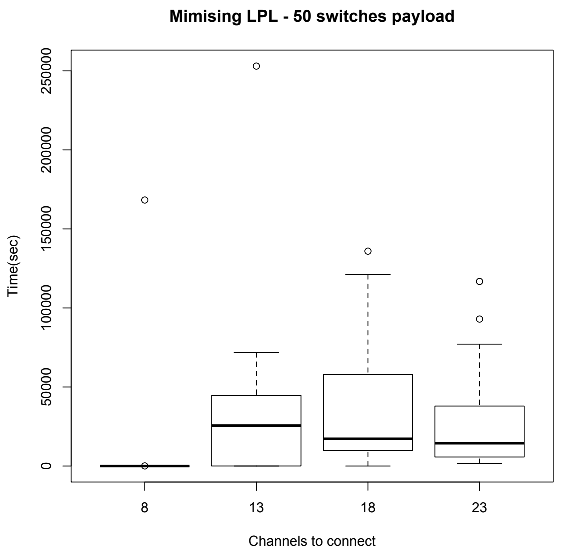


Figure 6.7: Time distribution on 50 switches payload when LPL is minimised.

6.4.2.2 100 switches payload

The experimental results on the 100 switches payload are presented in Table 6.4. Using the 120 hours termination condition, 70% of the instances of 8 channels to connect were solved whereas for the sets of 18 and 23 channels the hit rate is only 20% and 10% respectively. Considering the termination condition of 10 minutes, 66.66% of the instances were solved of size 8 channels to connect, whereas for the sets of 13 and 18 channels the hit rate is 30% and 10% respectively. None instance was solved for the cases of 23 channels to connect.

As denoted from the high standard deviation, the selected set of channels to connect influences the required computational time. For the sets of 13 channels to connect the standard deviation of the time is 6061.92 seconds. For the cases of 18 channels the average computational time is 20349.53 with standard deviation 46475.84 seconds. As illustrated in Figure 6.8, similarly to the 50 switches payload, the instances with the worst fitness value (highest LPL) are the ones that required the highest computational time in average, whereas the instances with the best fitness (lowest LPL) were solved faster. When connecting 8 channels, the instances with the worst fitness required 2402 seconds whereas the ones with the lowest

	Channels	Hitrate	Avg. Fitness	Avg. Time	Min Time	Max Time
120 hours	8	70%	2.95 ± 0.49	257.05 ± 940.13	0.16	4314
	13	33.33%	3.3 ± 0.67	2016.76 ± 6061.92	0.94	19266.2
	18	20%	3.16 ± 0.408	20349.53 ± 46475.84	5.53	115158.7
	23	10%	5 ± 0	184541.9 ± 161483	50781	363929.2
10 min	8	66.66%	2.9 ± 0.44	54.21 ± 144.24	0.16	489.68
	13	30%	3.11 ± 0.333	100.15 ± 120.42	0.94	344.33
	18	10%	3 ± 0	64.256 ± 86.16	5.53	163.17
	23	0%	–	–	–	–

Table 6.4: Minimising LPL on 100 switches payload.

LPL only 0.32 seconds. For the sets of 18 channels to connect the easiest instances required 789 seconds whereas the ones with the longest paths required in average 115158 seconds. Lastly, for the sets of 23 channels to connect, the fitness of the solved instances was identical and thus no distinguish is done between the instances with the highest and the lowest fitness.

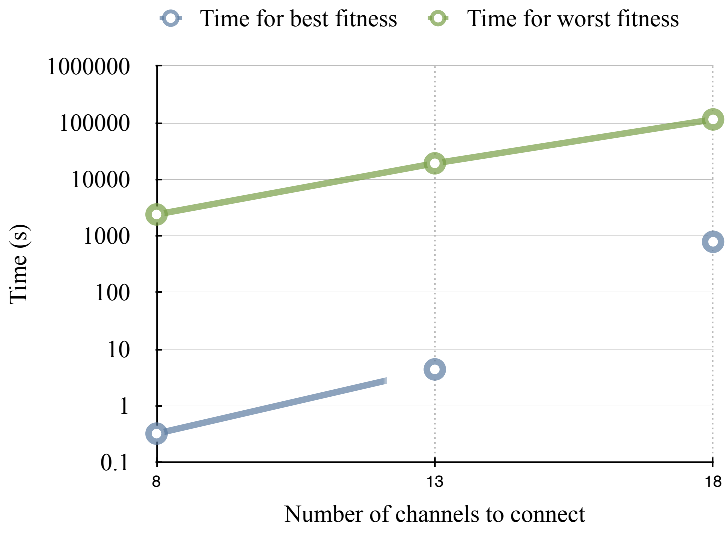


Figure 6.8: 100 switches payload. Average time (in seconds) of the set of instances with the worst(maximum) and best(minimum) fitness.

6.4.3 Minimising the Number of Switch Changes

When optimising the number of switch changes, the initial set of positions of the switches, referred to as initial payload status, influences the solution quality. For this reason, 30 initial payload statuses per case, chosen uniformly at random, are additionally considered in our experiments when this objective is optimised. A total of 900 instances is therefore optimised for each size of channels to connect. As the operational case is of interest, a termination condition of ten minutes was used. In Table 6.5 the numerical results when the number of switch changes is minimised, for both payload sizes, are presented.

6.4.3.1 50 switches payload

The hit rate is 100% in all cases for the 50 switches payload, i.e. all tackled instances were solved exactly. When 8 channels are connected to the 50 switches payload, the fitness value is in average 5.97 switch changes and for 23 channels to connect the fitness reaches an average value of 19.82 switch changes. The computational time for connecting the sets of 8 channels on the 50 switches payload is only 1.04 seconds, with maximum value 6.85 seconds, whereas when 23 channels are connected the average required time is 20.76 seconds with maximum value 377.95 seconds. It can be observed that the maximum required computational time for these instances, is significantly lower compared to the termination condition.

Minimising this objective is an easier problem compared to minimising LPL. Indeed, the hit rate when optimising the LPL is significantly lower, on the same problem instance sizes. On the 50 switches payload, the computational time on the solved instances when minimising LPL, is between 1500 and 12000 times higher compared to the computational time when the number of switch changes is minimised.

	Channels	Hitrate	Avg. Fitness	Avg. Time	Min Time	Max Time
50sw	8	100%	5.97 ± 1.33	1.04 ± 1.055	0.18	6.85
	13	100%	10.01 ± 1.70	3.65 ± 3.73	0.28	45.36
	18	100%	14.21 ± 2.02	11.53 ± 13.89	0.47	222.29
	23	100%	19.82 ± 1.97	20.76 ± 28.01	0.57	377.95
100sw	8	100%	6.69 ± 1.65	1.52 ± 1.65	0.23	15.24
	13	100%	11.65 ± 2.80	13.56 ± 34.18	0.4	494.39
	18	97%	16.43 ± 3.15	50.37 ± 82.93	0.81	593.97
	23	43.22%	21.87 ± 4.27	215.29 ± 155.910	3.47	582.32

Table 6.5: Minimising number of switch changes.

In order to evaluate the influence of the initial payload status, when the number of switch changes is minimised, Table 6.6 provides results for each of the 30 instances of 23 channels to connect on the payload switch matrix with 50 switches. The columns present respectively the average and standard deviation of the fitness, the minimum and maximum fitness, the average and standard deviation of the computational time and the minimum and maximum computational time for the 30 different payload statuses. A remarkable standard deviation is observed in both fitness and computational time. The minimum and the maximum difference in the required time is 33.36 seconds and 376.17 seconds respectively with an average of 100.754 sec. The minimum difference in the value of the objective function is 6 changes and the maximum 13, with an average of 8 switch changes.

Furthermore, the most difficult instances, i.e. the instances 18 and 26 that required the highest computational time (46.52 seconds and 45.80 seconds respectively) are the ones with the highest fitness values (21.4 and 21.26 switch changes respectively). The easiest problem instance, i.e. instance 28 that is solved in average after 11.50 seconds, is the one with the minimum average fitness of 18.56 changes. This correlation is illustrated in Figure 6.9.

Instance	Avg. Fitness	Min Fitness	Max Fitness	Avg. Time	Min Time	Max Time
1	19.03 ± 1.65	15	23	11.50 ± 10.40	0.79	46.96
2	19.66 ± 2.27	15	24	18.94 ± 17.29	1.28	72.41
3	19.23 ± 1.83	15	23	13.36 ± 11.83	1.17	45.61
4	20.23 ± 1.67	15	23	20.35 ± 18.66	2.39	92.87
5	19.23 ± 1.83	15	23	22.48 ± 31.57	0.86	147.59
6	19.63 ± 1.88	16	23	16.76 ± 13.73	1.64	59.15
7	19.4 ± 2.06	15	22	14.47 ± 12.55	0.71	52.59
8	20.36 ± 2.52	15	28	31.54 ± 49.03	3.52	246.67
9	20.83 ± 1.76	17	24	33.14 ± 23.52	3.34	118.97
10	19.73 ± 1.92	17	25	20.60 ± 19.42	2.7	93.09
11	19.96 ± 1.80	16	23	23.53 ± 28.36	1.7	152.46
12	19.43 ± 1.67	17	23	21.44 ± 23.33	2.79	118.12
13	19.9 ± 1.66	16	23	14.72 ± 12.82	2.96	52.08
14	19.5 ± 1.90	16	24	16.19 ± 12.85	2.28	53.85
15	19.56 ± 1.63	17	23	21.57 ± 20.87	1.43	92.11
16	20.50 ± 1.69	17	25	27.31 ± 30.85	2.53	160.21
17	19.3 ± 1.85	15	23	13.52 ± 11.37	2.44	43.21
18	21.4 ± 2.19	18	28	46.52 ± 49.00	4.76	208.33
19	19.7 ± 1.93	16	24	21.11 ± 21.76	1.02	90.57
20	19.63 ± 1.67	16	22	13.60 ± 9.87	1.32	39.36
21	20.2 ± 2.04	15	24	19.58 ± 16.57	2.75	66.23
22	20.1 ± 1.53	16	23	17.16 ± 11.39	1.03	49.54
23	19.96 ± 2.07	14	23	17.76 ± 11.00	1.39	38.94
24	20.4 ± 1.92	16	24	35.76 ± 69.80	1.78	377.95
25	19.56 ± 1.59	16	22	12.28 ± 9.45	2.53	40.18
26	21.26 ± 1.74	18	25	45.80 ± 63.31	5.8	344.42
27	19.9 ± 2.07	14	24	13.00 ± 8.14	2.72	36.35
28	18.56 ± 1.94	14	22	11.50 ± 10.04	0.7	34.06
29	19.16 ± 2.15	16	25	12.50 ± 11.69	0.57	59.09
30	19.23 ± 2.07	13	23	14.77 ± 12.25	2.37	52.92

Table 6.6: Quantifying the influence of the initial payload status: Minimising number of switch changes for 30 different payload statuses when connecting 23 channels on 50 switches payload.

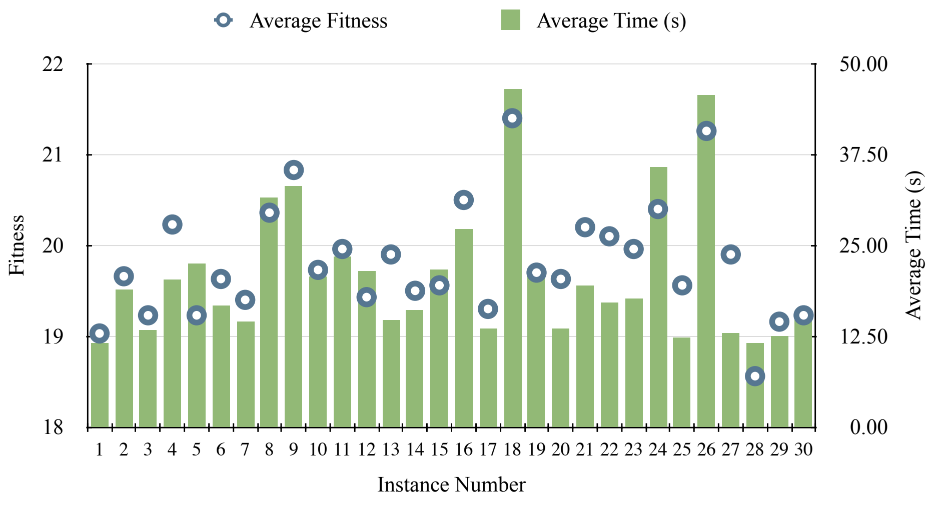


Figure 6.9: Correlation between average fitness and average time (in seconds) for each instance of the 50 switches payload.

6.4.3.2 100 switches payload

For the larger payload with 100 switches, as can be seen from Table 6.5, the hit rate is 100% when the sets of 8, and 13 channels are connected, whereas for the cases of 18 and 23 channels to connect the hit rate is 97% and 43.22% respectively. For the sets of 8 and 23 channels to connect the fitness is 6.69 and 21.87 switch changes respectively. The average time when connecting 8 channels is 1.52 seconds, with a maximum of 15.24 seconds, whereas for the sets of 13 channels to connect the average time reaches 13.56 seconds with a maximum of 494.39 seconds. When 18 channels are connected, the corresponding values of the average and maximum time are 50.37 and 593.97 seconds respectively, whereas for the sets of 23 channels to connect, the time reaches 215.76 seconds with a maximum of 582.32 seconds.

6.4.4 Minimising the Channel Interruptions

The proposed mathematical model was implemented in AMPL modeling language and experiments were conducted using an ILP solver with a time limit: CPLEX 11 on a machine with two Intel Xeon X5355 2,66 GHz processors with 4 cores each, 32GB of RAM and running Ubuntu 8.04. As indicated in Table 6.9, experiments were performed on a realistic satellite payload with a matrix of 185 switches (input and output switch matrix) allowing the maximum connection of 35 channels. 3 initial configurations with 15, 21 and 24 pre-connected channels were selected uniformly at random. For each initial configuration, 20 different sets of 2, 3 and 4 channels to add were chosen uniformly at random.

The numerical results are presented in Table 6.8. In the first column the number of initially connected channels is provided, whereas the number of channels to add is displayed in the second column. The third column denotes the average and standard deviation of the computational time required to solve the 20 different instances of each size. In the last column the hit rate is indicated, i.e. the percentage of cases where the optimal solution was found within the given time limit.

Number of switches / Type	185 / R,T.C
Number of amplifiers	35
Maximum nb. of channels to connect	35
Nb. of initially connected channels	15, 21, 24
Nb. of initial configurations per size	3
Nb. of channels to connect	2, 3, 4
Nb. of channels sets per size	20

Table 6.7: Payload instances.

Initial Channels	Added Channels	Avg. Time	Hit Rate(%)
15	2	12.117 $_{\pm 28.082}$	100
15	3	30.271 $_{\pm 36.701}$	95
15	4	31.046 $_{\pm 47.473}$	85
21	2	10.133 $_{\pm 8.820}$	90
21	3	23.995 $_{\pm 31.105}$	95
21	4	28.238 $_{\pm 27.419}$	85
24	2	39.263 $_{\pm 65.673}$	100
24	3	39.105 $_{\pm 30.609}$	85
24	4	79.449 $_{\pm 1119.55}$	65

Table 6.8: Minimising the number of channel interruptions.

The average value of the objective function is $0.817_{\pm 0.711}$ number of channel interruptions among all the performed experiments. This value is small due to the relatively small number of initially connected channels. The required computational time increases with the number of channels to connect in 5 out of 6 cases, the exception being for the instance with 24 initial channels and 3 added channels. As an example, for the configuration with initially 15 connected channels the required time when 2 channels are added is 12.117 seconds with 100% hit rate. When 3 and 4 channels are added the computational time reaches 30.271 and 31.046 seconds respectively and the hit rate is 95% and 85% respectively. The standard deviation of the computational time is very high which denotes its dependency on the chosen set of channels to be connected. The highest standard deviation (1119.55 seconds) occurs when 4 channels are added to the configuration with 24 initially connected channels. Besides, the hit rate is dependent on the number of added channels. Indeed, the highest hit rate, i.e. 100%, is obtained on the simplest instances with 2 channels to connect. Conversely, the worst hit rate, 65%, is obtained on the instance with 4 channels to connect on the payload with 24 pre-connected channels.

The variation of the time and the hit rate when 2, 3 and 4 channels are connected is illustrated in Figure 6.10. The average hit rate, when connecting 2 channels is 96.66%, whereas for the cases of 3 and 4 channels to connect, the hit rate is 91.66% and 78.33% respectively. The lower hit rate of 65% occurred when 24 channels are initially connected and 4 channels have to be added to the final configuration. The average required time when 2 channels are connected, considering all tackled instances, is 20.504 seconds whereas for the

	1st Payload	2nd Payload
Number of switches / Type	50 / T	100 / T
Maximum nb. of channels to connect	23	35
Nb. of channels to connect	8, 13, 18, 23	
Nb. of channels sets per size	30	

Table 6.9: Payload instances.

sets of 3 and 4 channels to connect the computational time is 93.371 and 138.733 seconds respectively.

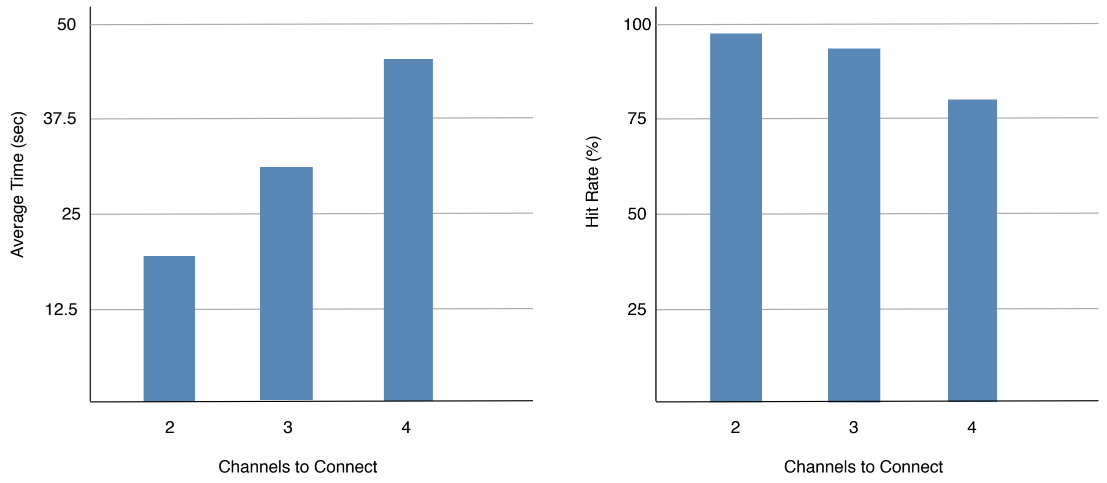


Figure 6.10: Minimising Channel Interruptions. Average Time and Hit Rate variation over all the tackled instance sizes.

6.4.5 Minimising the Input Power to Saturation

In this case the difference compared to the previously considered payload instances is the type of switches used. As shown in Table 6.9 we tackled two realistic payloads, the first one with 50 switches of type T and 23 amplifiers, and the second one with 100 switches of type T and 35 amplifiers. 30 different sets of 8, 13, 18 and 23 channels were chosen uniformly at random. The initial configuration operational problem case was considered.

6.4.5.1 Total IPS

In Table 6.10 the results of the first approach, that minimises the total IPS, are presented for both payloads. In this table, the hit rate, which indicates the percentage of instances that were solved exactly, is provided, as well as the average and standard deviation of the fitness value and of the required computational time.

When minimising the total IPS the hit rate is in all cases 100% for both payload sizes. In addition, the average computational time is very low compared to the termination condition.

	Channels	Hit Rate(%)	Fitness	Time(sec)
50 sw	8	100%	-236.541 ± 6.640	1.352 ± 1.613
	13	100%	-382.871 ± 5.385	3.939 ± 6.305
	18	100%	-526.025 ± 6.725	9.689 ± 18.385
	23	100%	-670.140 ± 5.385	30.463 ± 41.907
100 sw	8	100%	-232.388 ± 5.882	2.844 ± 3.095
	13	100%	-371.612 ± 8.919	17.709 ± 24.704
	18	100%	-520.548 ± 9.384	34.720 ± 71.388
	23	100%	-670.807 ± 5.266	276.565 ± 145.166

Table 6.10: Minimising Total IPS

For example, when connecting 23 channels, which is the maximum number of channels to connect on the payload with 50 switches, the required average time is only 30.463 seconds whereas for the cases of connecting 18 channels the average time is only 9.689 seconds. The corresponding time for connecting 23 channels in the payload with 100 switches is 276.565 seconds and 34.720 seconds for 18 channels to connect.

As expected the time increases with the number of channels to connect. For example, the average time is 1.352 seconds when 8 channels are connected and reaches 30.463 seconds for the case of 23 channels. For the payload with 100 switches, the average time is 2.844 seconds and 276.565 seconds for 8 and 23 channels to connect respectively. The high standard deviation indicates the influence of the selected set of channels to connect to the computational time. The boxplots provided in Figure 6.11 and Figure 6.12 visualise the distribution of the time when the total IPS is minimised for both payloads, as all the instances have been solved in this case. The median values are very low and few outliers represent the few instances that require the highest compuntational time. These outliers represent the instances with the highest fitness in average as well. For example, on the 50 switches payload, for the sets of 18 channels to connect, the outliers have an average fitness -520.970 compared to -526.802 for the rest instances. For 23 channels, the outlier has fitness of -663.581 compared to -670.544 for the rest instances. The same holds on the 100 switches payload where, for 18 and 23 channels to connect, the average fitness of the outliers is -512.141 and -668.963 respectively compared to -522.229 and -671.012 for the rest instances.

The average value of the total IPS for the payload with 50 switches, when 8 channels are connected, is -236.541 and reaches -670.140 for 23 channels to connect. For the payload with 100 switches, the fitness for connecting 8 channels is -232.388 and reaches -670.807 for the cases of 23 channels to connect.

6.4.5.2 Highest IPS

The results of the second approach, that minimises the highest IPS, are displayed in Table 6.11. Minimising the highest IPS is a more difficult optimisation problem. As can be seen in Table 6.11 for the 50 switches payload, the hit rate for the cases of 13 channels to connect is 76.666% and for 18 channels only 6.666% of the instances were solved exactly within the ten minutes termination condition. None instance was solved exactly for the sets of 23 channels to connect. For the larger payload with 100 switches, the hit rate is 90% when 8 channels

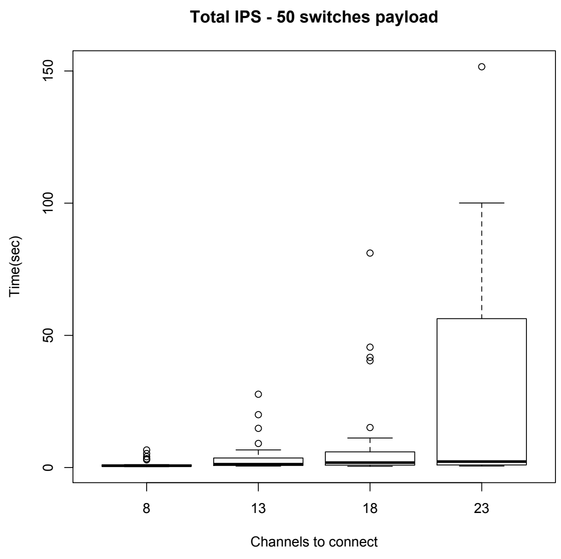


Figure 6.11: Time distribution on 50 switches payload when the total IPS is minimised.

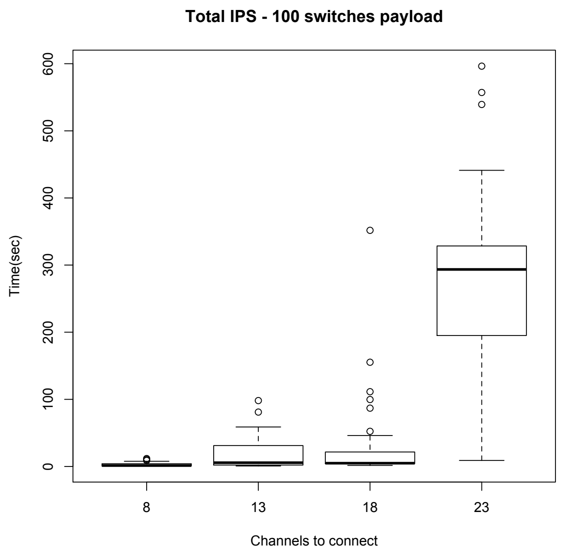


Figure 6.12: Time distribution on 100 switches payload when the total IPS is minimised.

are connected and 20% for the cases 13 channels to connect. Only 3.333% of the instances were solved for the size of 18 and none instance for the sets 23 channels to connect.

Logically, the computational time is higher compared to the first optimisation problem where the total IPS is minimised, in all cases. For example, on the payload with 50 switches, 286.525 seconds were required in average for the cases of 13 channels to connect and 311.13 seconds for the cases of 18 channels compared to 3.939 seconds and 9.689 seconds respectively, when the total IPS is minimised. For the solved instances we observe that the required

	Channels	Hit Rate(%)	Fitness	Time(sec)
50 sw	8	100%	$-26.128_{\pm 0.552}$	$109.295_{\pm 106.326}$
	13	76.666%	$-25.652_{\pm 0.342}$	$286.525_{\pm 173.468}$
	18	6.666%	$-25.361_{\pm 0}$	$311.13_{\pm 213.461}$
	23	0%	–	–
100 sw	8	90%	$-25.429 \pm_{0.894}$	$204.511 \pm_{202.385}$
	13	20%	$-25.180 \pm_{0.768}$	$271.745 \pm_{122.278}$
	18	3.333%	$-25.917 \pm_{0}$	$227.980 \pm_{0}$
	23	0%	–	–

Table 6.11: Minimising Highest IPS

computational time when minimising the highest IPS is between 30 and 80 times higher compared to the required computational time when the total IPS is minimised. For the 100 switches payload the average required computational time when 8 channels are connected is 204.511 seconds and for the cases of 13 channels to connect the average time reached 271.745 seconds compared to 2.844 and 17.709 seconds when the total IPS is minimised. For the solved instances it is observed that the computational time when minimising the highest IPS is between 7 and 80 times higher compared to the required computational time when the total IPS is minimised.

Since minimising the total IPS is an easier optimisation problem, we propose to investigate the fitness gap between the highest IPS found when the total IPS is minimised and the corresponding optimal value which is obtained when the highest IPS is minimised. Values are presented in Figure 6.13 for the 50 switches and 100 switches payload respectively. The value of highest IPS is displayed for the commonly solved instances of both approaches. In this figure the values of the total IPS correspond to the average value of the highest IPS when the total IPS is minimised, and the values of the highest IPS denote the optimal values. As can be observed from these figures, when the highest IPS is minimised the value is always lower. For the 50 switches payload, the fitness difference in the fitness when connecting 8 channels is 0.053 (from -26.075 down to -26.128) and for the cases of 13 and 18 channels to connect the the gap is 0.337 and 0.818 respectively. For the 100 switches payload the gap in fitness when 8 and 13 channels are connected is 0.118 and 0.064 respectively. Lastly, for the cases of 23 channels to connect the difference is 0.383.

To summarise the analysis of the experimental results, the proposed ILP based exact method provided valid results for minimising the total IPS and the highest IPS, for a given set of channels to connect. The first approach, i.e. minimising the total IPS, was solved significantly faster, as all considered instances were solved exactly within the defined time constraint. However, the drawback of this approach is the small degradation that occurs between the value of the highest IPS computed by this approach, compared to the optimal value of the highest IPS, which is obtained by the second approach, i.e. minimising the highest IPS.

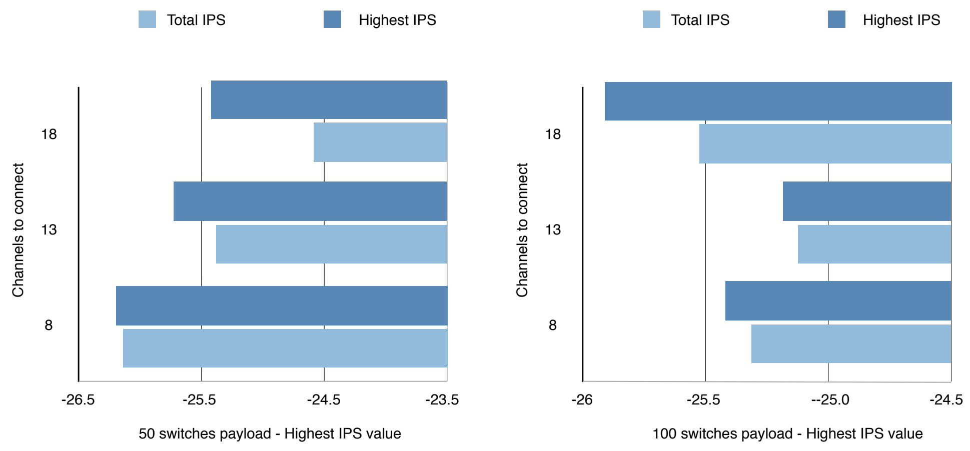


Figure 6.13: Average highest IPS on the commonly solved instances.

6.5 Summary

This chapter proposes an Integer Linear Programming (ILP) optimisation model for the payload switch matrix configuration and reconfiguration problem. Variants of the model are developed for the optimisation of the considered operational objectives such as the length of the longest channel path, the number of switch changes, the channel interruptions, and the minimisation of the total and the highest Input Power to Saturation (IPS) for the used amplifiers. The model allows to deal with different problem cases, such as initial configuration, reconfiguration and restoration.

The experimental results for all the considered oeprational objectives, with the use of an exact solver, demonstrated the validity and the efficiency of the method as well as the limitations in terms of required computational time.

The problem of minimising the length of the longest channel path (LPL) was shown to be the most difficult problem to solve exactly. For instance, on a payload size with 50 switches and 23 amplifiers, considering the planning phase of the initial configuration problem, only 56.66% of the tackled instances were solved exactly when connecting the sets of 18 and 23 channels. On the contrary, minimising the number of switch changes is a significantly easier objective to solve. All the instances of the same size, have been efficiently solved. The computational time when minimising LPL is in average between 1500 and 12000 times higher compared to the required computational time when the number of switch changes is minimised. Concerning the power, when minimising the total Input Power to Saturation (IPS) all the instances have been solved exactly. The computational time when minimising the highest IPS is between 30 and 80 times higher compared to the required computational time when the total IPS is minimised for the 50 switches payload. It was remarked a gap though between the highest IPS value computed when the total IPS is optimised compared to the optimal highest IPS value in all cases. In all the experiments a high standard deviation on the computational time is observed, which denotes the influence of the chosen set of channels to connect. In average the instances that require the highest computational time are those with the highest fitness values.

The following chapter focuses on multi-objective optimisation. With the use of the developed optimisation models, exact multi-objective algorithms are applied and compared for the first time to the considered problem. The focus is on the first two objectives, namely the length of the longest channel path and the number of switch changes. The aim is to provide to the engineers the Pareto front of solutions, allowing them to select the one that better fits their requirements.

Chapter 7

Multi Objective Optimisation using Exact Methods

Contents

A large part of the optimisation problems that arise in many sectors of the industry are multi-objective as they have more than one conflicting objectives to be simultaneously optimised. In general, the multi-objective optimisation problems can be divided in two main categories. The continuous optimisation problems, where the solutions are encoded with real-valued variables, and the problems where the solutions are encoded using discrete variables. The Multi Objective Combinatorial Optimisation (MCOP) problems belong to the latter class [19]. At first, this chapter provides some common definitions related to the multi objective optimisation.

Definition 1. *A multi-objective optimisation problem (MOP) may be defined as:*

$$MOP = \begin{cases} minF(x) = (f_1(x), f_2(x), \dots, f_n(x)) \\ s.c \quad x \in S \end{cases}$$

where n $(n \geq 2)$ is the number of objectives, $x = (x_1, \dots, x_k)$ is the vector representing the decision variables, and S represents the set of feasible solutions associated with equality and inequality constraints and explicit bounds. $F(x) = (f_1(x), f_2(x), \dots, f_n(x))$ is the vector of objectives to be optimised. In the case of MCOP problems, the vector $x = (x_1, \dots, x_k)$

has a finite number of possible values, i.e., each variable is defined in a discrete and bounded interval.

Let $Y = F(S)$, to represent the realisable points in the objective space, and $y = (y_1, \dots, y_n)$, with $y_i = f_i(x)$ is a point of the objective space.
If the optimum for each objective function is known then the ideal vector can be defined.

Definition 2. *The ideal vector vector $y^* = (y_1^*, \dots, y_n^*)$ is the vector which optimises each objective function, i.e.: $y_i^* = min(f_i(x)), x \in S$.*

$$\forall i \in \{1, \dots, n\} : u_i \leq v_i \wedge \exists i \in \{1, \dots, n\} : u_i < v_i$$

However, in most real life cases the objectives are in conflict and as a result, contrary to the single-objective case, the multi-objective problems do not admit a unique optimal solution but a set of so-called non-dominated solutions.

Definition 3. *An objective vector $u = (u_1, \dots, u_n)$ is said to dominate $v = (v_1, \dots, v_n)$ (denoted by $u \prec v$) if and only if no component of v is smaller than the corresponding component of u and at least one component of u is strictly smaller, that is,*

$$\forall i \in \{1, \dots, n\} : u_i \leq v_i \wedge \exists i \in \{1, \dots, n\} : u_i < v_i$$

If none of the solutions dominates the others then the solutions are non-dominated. An illustration of non-dominated solutions for a minimisation problem with two objectives is provided in Figure 7.1.

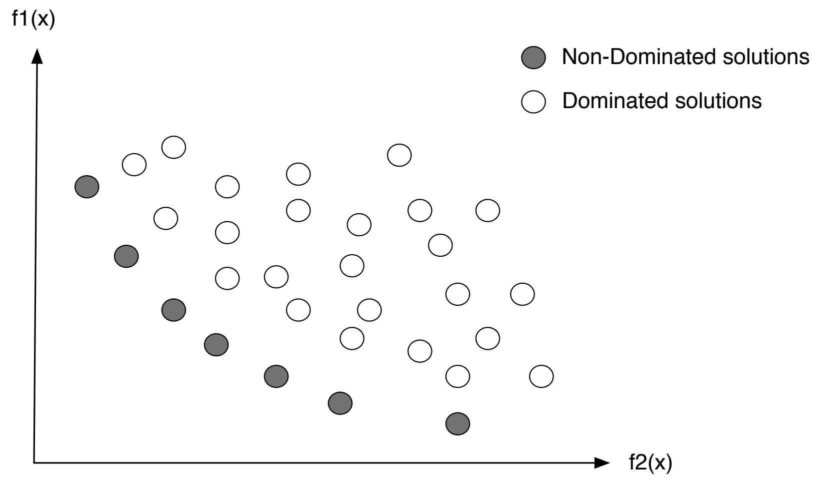


Figure 7.1: Example of non-dominated solutions.

A Pareto optimal solution denotes that it is not possible to find a solution that improves one objective without decreasing the quality of at least another objective.

Definition 4. *A solution $x^* \in S$ is Pareto optimal if for every $x \in S$, $F(x)$ does not dominate $F(x^*)$, that is, $F(x) \nprec F(x^*)$.*

Definition 5. *A Pareto solution is called supported if it can be obtained by optimising a linear combination of the objectives.*

A MOP may have a set of solutions known as the Pareto optimal set whereas the Pareto front is the image of the Pareto set in the objective space.

Definition 6. *For a given* $MOP(F, S)$*, the Pareto optimal set is defined as* $P^* = \{x \in S / \nexists x' \in S, F(x)') \prec F(x)\}$.

Definition 7. *For a given* $MOP(F, S)$*, and its Pareto optimal set* P^**, the Pareto front is defined as* $PF^* = \{F(x), x \in P^*\}$.

This chapter deals with the multi-objective case for the switch matrix configuration problem. The following section provides an overview of the existing in the literature exact optimisation methods for solving the multi-objective combinatorial optimisation problems. The methods that are applied to the considered problem, namely the ϵ-constraint method and the adaptive ϵ-constraint method are detailed in Section 7.1.1 and Section 7.1.2 respectively, whereas the ILP optimisation model is described in section 7.2. Finally, the experimental results are analysed in Section 7.3.

7.1 Multi-objective exact methods

Few exact algorithms have been proposed in the literature for solving exactly the multi-objective combinatorial optimisation problems. These algorithms guarantee to find the full set of Pareto solutions, and their performance relies on the efficiency of the used method for solving the single objective optimisation problems. They are therefore limited in the size of the tackled problem instances and the number of objectives.

The Two-Phase-Method (TPM) is a general framework for solving bi-objective combinatorial optimisation problems [68]. Starting by determining the extreme points of the front, in the first phase of the algorithm, all the supported solutions are found with the optimisation of aggregations in the form $\lambda_1 f_1 + \lambda_2 f_2$. In the next step, the algorithm recursively explores the existence of a supported solution between two given supported solutions. Once all supported solutions have been found, the second phase of the method consists in searching all the triangles, underlying each pair of adjacent supported solutions, in order to find all the non-supported solutions. The algorithm was applied initially for a bi-objective assignment problem, but has also been applied to different problems including network flow problems [59]. For problems with more than two objectives, a generalisation of the two-phase-method was proposed in [56].

The Parallel Partitioning Method (PPM) is another technique that determines the whole Pareto front in three stages [46]. At first, the extreme Pareto optimal solutions are found in order to limit the search space. The second step consists in equally splitting the search space in order to find a subset of well distributed Pareto solutions. For each split, a specific Pareto solution is computed, by optimising one objective while the other one is limited. During this step, supported as well as non-supported solutions are found. Finally in the last stage, the Pareto solutions that have not been found during the previous stages are computed, by reducing the search space using the solutions found on the second stage. This method was successfully applied on a bi-objective permutation flowshop problem [46]. The method k-PPM is an extension of PPM for $k > 2$ objectives. It was evaluated on a three-objective permutation flowshop problem [32].

In this study, two other well-known bi-objective exact algorithms are compared, namely the ϵ-constraint method [69] and the adaptive ϵ-constraint method [44]. Initial experiments on the considered problem have demonstrated that the computation of the extreme points is costly in terms of computational time. For this reason, it was chosen to apply the adaptive ϵ-constraint method as in this case the computation of the two extreme points is not a priori required. Furthermore, as the initial experiments demonstrated that the number of Pareto optimal solutions on the fronts is very low, it would be more suitable to split and explore the search space using a predefined constant. It is thus chosen to compare with the original ϵ-constraint method. Both the ϵ-constraint and the adaptive ϵ-constraint methods transform the bi-objective problem as a set of constrained single-objective problems. One objective function is used as the objective while the second objective function is used as constraint, which permits to obtain different solutions in the front. In Section 7.1.1 the ϵ-constraint is presented in details whereas the adaptive ϵ-constraint method is described in Section 7.1.2.

7.1.1 The ϵ-constraint method

The *ϵ-constraint technique* is based on the optimisation of one of the objectives, defined as the primary one, while considering the other objectives as constraints bound by some allowable levels ϵ_i. Supposing there is a procedure, namely $opt(f, \epsilon, \epsilon')$ that returns the optimal solution x of the following constrained problem:

$$\begin{aligned} lex\ min\ f(x) &= (f_1(x), f_2(x)) \\ subject\ to\ \epsilon_i &< f_2(x) \le \epsilon_i' \\ x &\in X \end{aligned}$$

The "lex min" denotes the lexicographic minimisation of the objectives. The pseudocode of the ϵ-constraint method is provided in Algorithm 1.

Algorithm 1 ϵ-constraint method

```
1:  Input: Objective bounds $\underline{f}, \overline{f} \in \mathbb{R}$ and increment $\delta \in \mathbb{R}$
2:  $P := \emptyset$
3:  $\epsilon := \overline{f}$
4:  while $\epsilon \ge \underline{f}$ do
5:      $x := opt(f, \epsilon - \delta, \epsilon)$
6:      if $\nexists X' \in P$ such that $x' \succ x$ then
7:          $P := P \cup \{x\}$
8:      end if
9:      $\epsilon := \epsilon - \delta$
10: end while
11: Output: Set of Pareto-optimal decision vectors $P$
```

The lower and upper bounds $\underline{f}, \overline{f}$ are known for each objective. At each iteration of the algorithm, the constrained single objective problem is solved (line 5) and if a new solution is found which is not dominated, the solution is added to the Pareto set (lines 6, 7). The constraint bounds of the second objective are modified with a predefined constant δ (line 9). The algorithm stops when the region between the lower and upper bounds has been searched.

7.1.2 The adaptive ϵ-constraint method

The adaptive ϵ-constraint method additionally uses information from the objective space (when available) during the search. In this case the bounds of the objectives do not need to be known. Supposing there is a procedure, called $opt(f, \epsilon')$ that solves the following constrained problem:

$$\begin{aligned} lex\ min\ f(x) = (f_1(x), f_2(x)) \\ subject\ to\ f_2(x) < \epsilon' \\ x \in X \end{aligned}$$

and returns the optimal solution x of the problem or null in case of unfeasibility. The pseudocode of this method is provided in Algorithm 2.

At each iteration of the algorithm, the constrained single objective problem is solved (line 4) and if a new non-dominated solution is found it is added to the Pareto set (lines 5, 6). The constraint bound for the second objective function is modified based on the found value $f_2(x)$ (line 8). As soon as the problem is infeasible the algorithm stops.

Algorithm 2 adaptive ϵ-constraint method

```
1: $P := \emptyset$
2: $\epsilon := \infty$
3: while $opt(f, \epsilon)$ not infeasible do
4:     $x := opt(f, \epsilon)$
5:     if $\nexists X' \in P$ such that $x' \succ x$ then
6:         $P := P \cup \{x\}$
7:     end if
8:     $\epsilon := f_2(x)$
9: end while
10: Output: Set of Pareto-optimal decision vectors $P$
```

7.2 The Mathematical Model

The purpose is to minimise simultaneously the length of the longest channel path and the number or required switch changes when configuring the payload. This problem may occur during the operational phase of the initial configuration problem case as has been described in Chapter 4. Long paths have to be avoided as they imply high losses on the signals (accumulated from the switches and the links) as well as restrictions for future reconfiguration processes that may be required. The switch changes have to be minimised as moving the a switch from its current position may cause its permanent failure. Therefore the scope is to obtain the Pareto front and allow the user to choose the solution that better fits the operational requirements.

The ILP model is based on the models presented in Sections 6.3.1 and 6.3.2 and integrates these two objectives. The number of switch changes was chosen to be the objective to optimise and the length of the longest channel path to be the objective function to constraint. This is due to the fact that minimising the number of switch changes is an easier objective to solve, as has been shown in Section 6.4.3. The complete version of the ILP model is provided below:

Complete ILP model

$$
\begin{array}{ll}
Variables: & \\
z \in \mathbb{Z} & \\
pos_s \in \mathbb{Z} & \forall s \in S \\
change_s \in \{0;1\} & \forall s \in S \\
flow_l \in \mathbb{Z} & \forall l \in L \cup \{l_o\} \\
b_{s,p} \in \{0;1\} & \forall s \in S, \forall p \in P \\
ampused_t \in \{0;1\} & \forall t \in T \\
flow_{l,c} \in \{0;1\} & \forall l \in L, \forall c \in C_{conn} \\
path_c \in \mathbb{Z} & \forall c \in C_{conn}
\end{array}
$$

$$
\begin{array}{l}
ObjectiveFunctions: \\
Min\ z \text{ (objective we chose to constraint)} \\
Min \sum\limits_{s \in S} change_s
\end{array}
$$

$$
\begin{array}{ll}
Constraints: & \\
flow_{l_i,i} = 1 & \forall l_i \in CL_{conn} \\
flow_{l_i} = 0 & \forall l_i \in \{\{CL_{in}\} - \{CL_{conn}\}\} \\
flow_{l1} + b_{s,p} * q - q \leq flow_{l2} \leq flow_{l1} - b_{s,p} * q + q & \forall s \in S,\ \forall p \in P,\ \forall (l_1, l_2) \in (L \cup \{l_0\})^2, \text{ s.t. } m_{s,p,l_1,l_2} = 1. \\
\sum\limits_{p \in P} b_{s,p} = 1 & \forall s \in S \\
pos_s = \sum\limits_{p \in P} p * b_{s,p} & \forall s \in S \\
change_s * n \geq pos_s & \forall s \in S \\
change_s \leq pos_s & \forall s \in S \\
ampused_t * q \geq flow_{tl_{in_t}} & \forall t \in T \\
\sum\limits_{t \in T} ampused_t = q & \\
\sum\limits_{c \in C_{conn}} flow_{l,c} \leq 1 & \forall l \in L \\
flow_l = \sum\limits_{c \in C_{conn}} c * flow_{l,c} & \forall l \in L \\
path_c = \sum\limits_{l \in L} flow_{l,c} & \forall c \in C_{conn} \\
path_c \leq z & \forall c \in C_{conn} \\
flow_{l_0} = 0 &
\end{array}
$$

7.3 Experimental Results

This section analyses the experimental results in terms of solution quality and required computational time. At first, information about the experimental setup and the problem instances tackled is provided. The numerical results are then detailed in Section 7.3.2.

7.3.1 Experimental Setup

An instantiation of the experimental framework is displayed in Figure 7.2. The operational case of the initial configuration problem is considered. Two exact multi-objective algorithms are applied, namely the ϵ-constraint method and the adaptive ϵ-constraint method. The exact solver used is IBM ILOG CPLEX 12.0.0 [2].

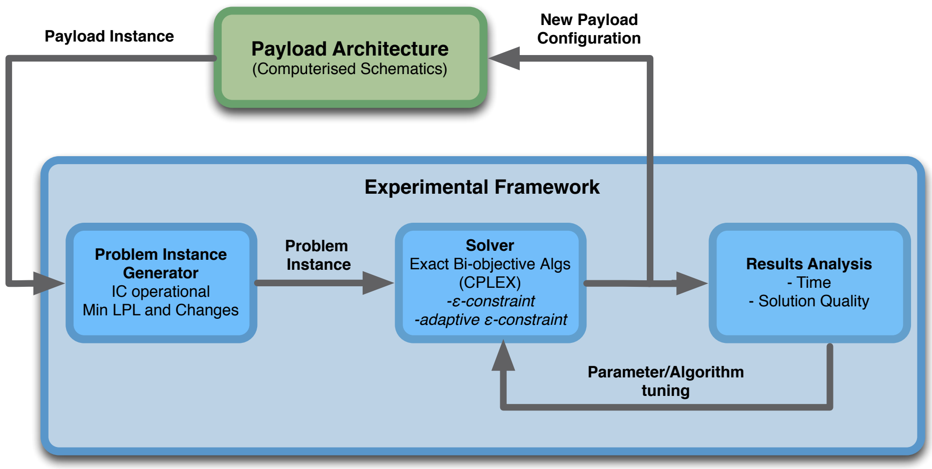


Figure 7.2: Instantiation of the experimental framework for minimising LPL and switch changes in the initial configuration operational case using exact methods.

For both exact algorithms, the number of switch changes was chosen to be the objective to optimise, while constraining the length of the longest path. This is justified from the fact that the number of switch changes is an easier objective to solve, as has been shown in Section 6.4.

Information about the tackled payload instances is provided in Table 7.1. Two realistic payload switch matrices were used, the first one with 50 switches of type R and 23 amplifiers, and the second one with 100 switches of type R and 35 amplifiers. 30 different sets of channels to connect of size 8, 13, 18 and 23 were chosen uniformly at random. As the objective of switch changes is of interest, we also considered 30 different payload statuses. As a result, each algorithm runs in total for 900 times for each size of channels to connect.

	Small Payload	Large payload
Number of switches / Type	50 / R	100 / R
Maximum nb. of channels to connect	23	35
Nb. of channels to connect	8, 13, 18, 23	
Nb. of channels sets per size	30	
Nb. of initial payload statuses	30	

Table 7.1: Payload instances.

For the ϵ-constraint method we chose to set the constant $\delta = 1$ to iteratively vary the constraint bound of the second objective. This is explained from the small number of Pareto optimal solutions in our instances, the low spread of solutions, and the integer nature of the variable.

All the experiments were performed using the HPC platform facilities of the University of Luxembourg [70], on an homogenous type of nodes with Xeon 6C, 2,26GHz on a single core with 2GB of RAM. The operating system was Debian GNU/Linux 6.0.9 and kernel 3.2.0-0.bpo.4-amd64. No core affinity was set. As the objective of switch changes is of interest while the satellite is operated in orbit, the termination condition was set to 10 minutes in all the experiments.

7.3.2 Numerical Results

In this section the numerical results are analysed for both methods. The hit rate, which denotes the percentage of instances where the full Pareto front was found within the termination condition, and the required computational time are compared.

7.3.2.1 50 switches payload

The numerical results are summarised in Table 7.2 and Table 7.3. In the first table, the hit rate and the number of Pareto points found on the fronts is analysed and in the second table the time required by the two algorithms is compared.

From Table 7.2, it can be observed that in terms of hit rate, the adaptive ϵ-constraint method performs better than the ϵ-constraint method. For the cases of 8 channels to connect, the adaptive method reached 97.77% compared to 73.11% for the ϵ-constraint method. When 13 channels are connected, the hit rate of the adaptive method remains relatively high, i.e. 72.22%, whereas it drops to only 3.44% for the ϵ-constraint. For the cases of 18 channels to connect, the adaptive ϵ-constraint performs still better than the ϵ-constraint with a hit rate of 4.11% compared to 2.66%. Lastly, when 23 channels are connected, none instance was solved by the the ϵ-constraint method, whereas the adaptive ϵ-constraint method solved 0.22% of the instances.

	Channels	Hitrate	Avg. Pareto	max Pareto
epsilon	8	73.11%	1.643 $\pm$ 0.720	4
	13	3.44%	1.733 $\pm$ 0.520	3
	18	2.66%	1.391 $\pm$ 0.583	3
	23	–	–	–
adaptive	8	97.77%	1.658 $\pm$ 0.71	4
	13	72.22%	1.622 $\pm$ 0.707	4
	18	4.11%	1.416 $\pm$ 0.603	3
	23	0.22%	1 $\pm$ 0	1

Table 7.2: Hit Rate comparison of the exact methods and average and maximum number of Pareto solutions found.

The reason of the poor performance of the ϵ-constraint method is the necessity of calculating the extreme points. This requirement implies the calculation of the length of the longest channel path which is a difficult objective to solve, as has been demonstrated in Section 6.4.2. For this reason, the required computational time and therefore the hit rate are significantly affected. On the other hand, the adaptive ϵ-constraint method is more efficient. The number of switch changes is an easier objective to solve and the lexicographic optimisation, required by this method, does not influence significantly the computational time.

The number of the Pareto solutions found, among the solved instances, is very low. For 8 channels to connect, it is in average 1.643 and 1.658 for the ϵ-constraint method and the adaptive ϵ-constraint method respectively. For the cases of 13 channels to connect, the average number of Pareto points is 1.622 for the adaptive ϵ-constraint method and 1.733 for the ϵ-constraint method, as the adaptive method solved more instances with few Pareto points on the front. When 18 channels are connected, the average number of points on the Pareto front is 1.416 and 1.391 for the adaptive and the ϵ-constraint method respectively. In all cases, the instance with the maximum number of Pareto points is found by the adaptive ϵ-constraint method. For example, for the case of 13 channels to connect, the adaptive method managed to solve instances with 4 Pareto points in the front, whereas for the ϵ-constraint method the maximum number of Pareto points among the solved instances was 3. An example of a generated front with 4 Pareto optimal solutions is displayed in Figure 7.3.

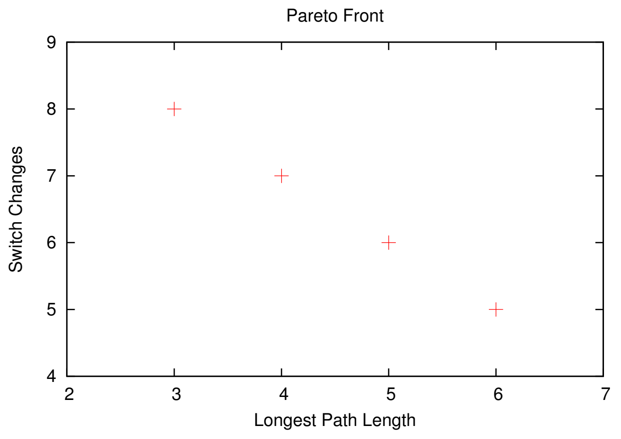


Figure 7.3: Example of a generated Pareto front with 4 optimal solutions when connecting 8 channels.

In Table 7.3 the average required time for the solved instances is presented for both methods. Even if the percentage of solved instances by the adaptive ϵ-constraint method is higher, the average required time is in most cases lower compared to the time needed by the ϵ-constraint method. More precisely, for 8 channels to connect the adaptive method required in average 49.75 seconds compared to 87.90 seconds needed by the ϵ-constraint method. When 13 channels are connected, the average time required by the adaptive method remains quite low with 58.80 seconds compared to 176.15 seconds for the adaptive ϵ-constraint approach. Only for the cases of 18 channels to connect, the average computational time for the adaptive ϵ-constraint method is 151.26 seconds, whereas the corresponding time for the ϵ-constraint method is 143.98 seconds. This however, is due to the difference of the hit rate as different instances were solved by the two methods.

	Channels	Time	min Time	max Time
epsilon	8	87.90 $\pm$ 135.517	0.64	591.01
	13	176.15 $\pm$ 157.032	3.33	588.73
	18	143.98 $\pm$ 144.297	7.78	483.84
	23	–	–	–
adaptive	8	49.75 $\pm$ 96.717	0.21	593
	13	58.80 $\pm$ 116.799	0.21	593
	18	151.26 $\pm$ 180.427	4.03	598.47
	23	501.02 $\pm$ 31.084	479.04	523

Table 7.3: Time(sec) comparison of the exact methods.

For the instances that have been commonly solved, the adaptive ϵ-constraint method performs faster in all cases. As can be seen from the results that are summarised in Table 7.4, for 8 channels to connect, the adaptive ϵ-constraint method required in average 29.469 seconds compared to 87.032 seconds required by the ϵ-constraint method, for the commonly solved instances, i.e. a difference of 57.563 seconds.

Channels	adaptive ϵ-constraint	ϵ-constraint
8	29.469 $\pm$ 69.657	87.032 $\pm$ 135.073
13	89.779 $\pm$ 160.388	176.149 $\pm$ 157.032
18	83.443 $\pm$ 119.913	135.721 $\pm$ 141.622

Table 7.4: Time(sec) comparison on the commonly solved instances.

For the commonly solved instances of 13 channels to connect, the average computational time for the adaptive and the ϵ-constraint method were respectively 89.779 and 176.149 seconds, i.e. a difference of 86.37 seconds. Finally, for the commonly solved instances of 18 channels to connect, the needed time was 83.443 and 135.721 seconds for the adaptive and the ϵ-constraint method respectively, i.e a time difference of 52.278 seconds. An illustration of the time difference between the two methods on the commonly solved instances is shown in Figure 7.4.

7.3.2.2 100 switches payload

For the larger payload with 100 switches, the hit rate and the average and maximum number of Pareto solutions found on the fronts, are presented in Table 7.5.

The better performance of the adaptive ϵ-constraint method compared to the ϵ-constraint method is confirmed from these results. When connecting 8 channels the hit rate is 52.11% for the adaptive ϵ-constraint method whereas for the ϵ-constraint method it is 28.44%. The hit rate decreases significantly when 13 channels are connected, as the ϵ-constraint method solved only 0.55% of the instances, and the adaptive method 2.88%. For the cases of 18 channels to connect, the ϵ-constraint method did not solve any instance whereas the adaptive ϵ-constraint method solved 0.22%. Finally none of the larger instances, i.e. 23 channels to connect, could be solved within the 10 minutes deadline. The average number of Pareto

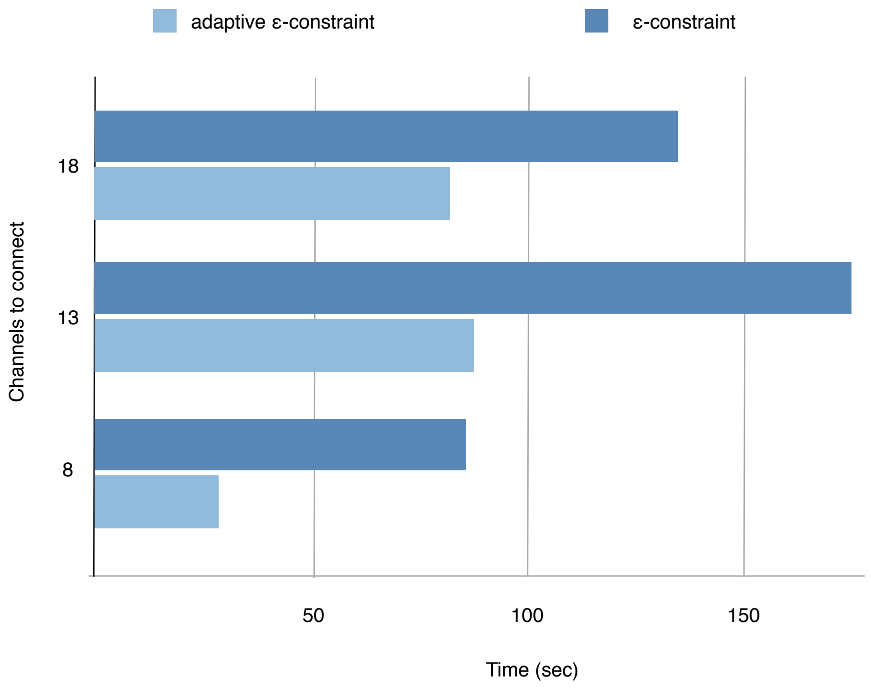


Figure 7.4: Time comparison on the commonly solved instances.

	Channels	Hitrate	Avg. Pareto	max Pareto
epsilon	8	28.44%	1.046 $\pm$ 0.211	2
	13	0.55%	1 $\pm$ 0	1
	18	–	–	–
	23	–	–	–
adaptive	8	52.11%	1.059 $\pm$ 0.246	3
	13	2.88%	1 $\pm$ 0	1
	18	0.22%	1 $\pm$ 0	1
	23	–	–	–

Table 7.5: Hit Rate comparison of the exact methods and average and maximum number of Pareto solutions found in the fronts.

solutions found in the front is low, with an average of 1.046 for the ϵ-constraint method and 1.059 for the adaptive ϵ-constraint method for 8 channels to connect.

The adaptive ϵ-constraint method solved the instances with the higher number of Pareto solutions on the front. As an example, for the cases of 8 channels to connect, the maximum number of Pareto optimal solutions found by the ϵ-constraint method is 2, whereas with the adaptive ϵ-constraint the maximum number of Pareto optimal solutions is 3. An illustration of such a front, with 3 optimal solutions, is provided in Figure 7.5.

In Table 7.6 the computational time required by these methods is displayed. It can be observed that the adaptive ϵ-constraint method performed faster in all cases despite the

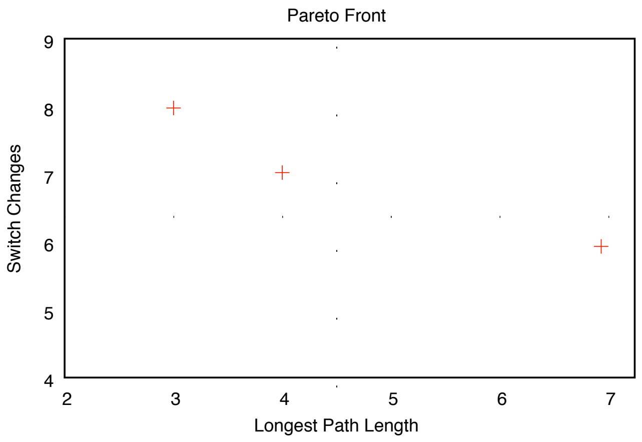


Figure 7.5: Example of a generated Pareto front with 3 Pareto optimal solutions when connecting 8 channels.

higher hit rate. For example, when 8 channels are connected, the average required time for the ϵ-constraint method is 110.43 seconds as opposed to 104.14 seconds for the adaptive ϵ-constraint method. When 13 channels are connected, the ϵ-constraint method required 154.24 seconds in average and the computational time for the adaptive ϵ-constraint method is 118.80 seconds.

	Channels	Time	min Time	max Time
	8	110.43 $\pm$ 162.009	1.110	597.790
epsilon	13	154.24 $\pm$ 66.287	76.3	240.640
	18	-	-	-
	8	104.14 $\pm$ 161.528	0.570	597.50
adaptive	13	118.80 $\pm$ 162.821	1.910	552.52
	18	105.13 $\pm$ 138.904	6.91	203.35

Table 7.6: Time(sec) comparison of the exact methods.

The computational time of the commonly solved instances is displayed in Table 7.7. For the cases of 8 channels to connect, the computational time required by the ϵ-constraint method for these instances is 109.72 seconds whereas for the adaptive ϵ-constraint method is 50.69, i.e. a difference of 59.03 seconds. For the commonly solved instances of 13 channels to connect, the adaptive ϵ-constraint method required 32.19 seconds compared to 154.24 seconds for the ϵ-constraint method, i.e a difference of 122.05 seconds. Figure 7.6 illustrates the time difference between the two methods, for the commonly solved instances.

Channels	adaptive ϵ-constraint	ϵ-constraint
8	50.69 $\pm$ 109.307	109.725 $\pm$ 161.927
13	32.19 $\pm$ 40.491	154.244 $\pm$ 66.287

Table 7.7: Time(sec) comparison on the commonly solved instances.

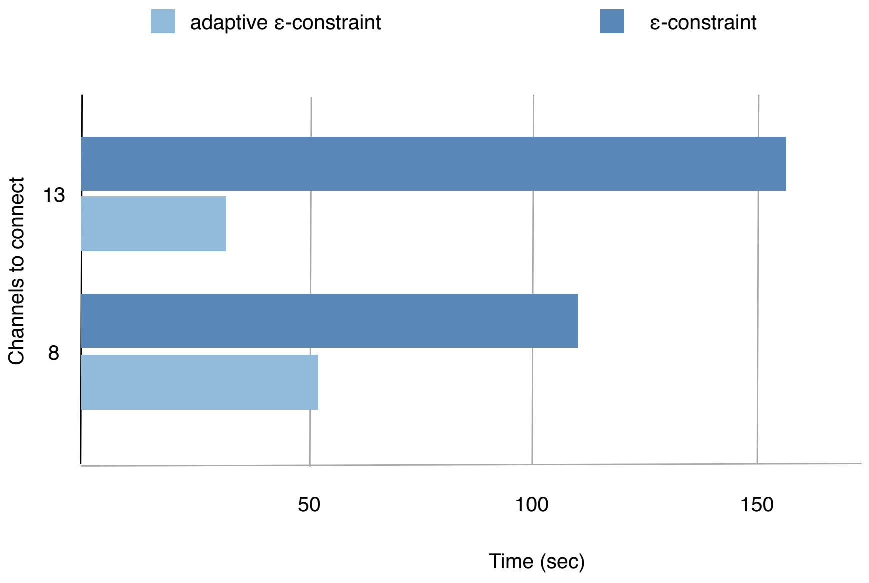


Figure 7.6: Time comparison on the commonly solved instances.

7.4 Summary

This chapter deals with the multi-objective exact optimisation of the payload switch matrix configuration problem. Two exact mutli-objective algorithms have been compared, namely the ϵ-constraint and the adaptive ϵ-constraint method. The aim is to minimise simultaneously the length of the longest channel path and the number of switch changes, considering the initial configuration problem case.

The experimental results, on two realistic payload sizes, demonstrated that the adaptive ϵ-constraint method performed better compared to the ϵ-constraint method as it managed to solve more instances within the termination condition. In addition, the adaptive method performed faster on the commonly solved instances. Besides, the instances with the higher number of Pareto optimal solutions on the fronts were solved with the use of the adaptive ϵ-constraint method.

The main limitation of the ϵ-constraint method is the requirement of the optimisation of the length of the longest path (LPL) individually, as one of the two extreme points. Minimising the LPL has been shown to be a difficult problem to solve and therefore this computation influences significantly the required time. This effect is more significant as the number of Pareto optimal solutions on the fronts is very low. On the contrary, for the adaptive ϵ-constraint method this bound does not have to be known a priori. The number of switch changes is an easier objective to solve and the lexicographic optimisation required by the adaptive ϵ-constraint method does not influence significantly the computational time. For the same reason it can be expected that the other combinatorial bi-objective exact methods, namely the PPM and the TPM, will be limited in their performance as the calculation of the two extreme points is part of these algorithms.

This chapter concludes the part of the thesis related to the exact methods. In the previous two chapters it was experimentally shown that with the proposed ILP based exact approach, time critical problem instances of large size may not be solvable in an exact manner, within a reasonable time for the engineers. Therefore, the following chapter focuses on applying and developing approximate methods, in this case metaheuristics, targeting to obtain fast solutions of good quality. Single-solution and population-based metaheuristics are investigated. The aim is to minimise the length of the longest path, as it has been experimentally shown to be the hardest objective to solve exactly. In addition, the initial configuration problem is considered, as in this case a larger search space has to be efficiently explored.

Part III

Optimisation with Metaheuristics

Chapter 8

Optimisation with Metaheuristics

Contents

Metaheuristics make part of approximate algorithms as has been shown in the classical classification of the optimisation methods, displayed in Figure 6.1. These algorithms are widely used to solve difficult industrial and scientific optimisation problems which can not be solved exactly within a reasonable time. Heuristic approaches are therefore applied in order to achieve fast solutions of good quality. Detailed information about metaheuristics can be found in [36, 24].

This category of problem solving techniques includes well-known methods like simulated annealing [6], tabu search [37] or various population-based methods like genetic algorithms [40]. Metaheuristics can be classified in two main categories, single-solution metaheuristics, that improve a unique solution and population-based metaheuristics where an iterative improvement of a population of solutions takes place.

In this chapter, single-solution and population-based metaheuristics are applied for the payload switch matrix configuration optimisation problem. In single solution based metaheuristics, at each iteration, a generation of a set of candidate solutions and a replacement of the current single solution is performed, as illusrtated in Figure 8.1. In the first step, candidate solutions are generated with the use of some operators and in the second step one candidate solution is chosen to replace the current solution. This process iterates until a given stopping criterion, for example the number of iterations. Examples of such metaheuristics are local search [6], tabu search or simulated annealing.

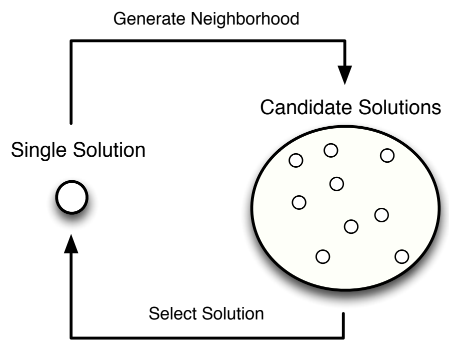


Figure 8.1: Single-solution based Metaheuristics

On the other hand, starting from an initial population of solutions, the population-based metaheuristics, at each iteration generate a new population and apply a replacement technique to the current population, as represented in Figure 8.2. This process iterates until a given stopping criterion, for example the number of iterations or a time limit. Examples of population-based metaheuristics are genetic algorithms, ant colony optimisation [33] and particle swarm optimisation [42].

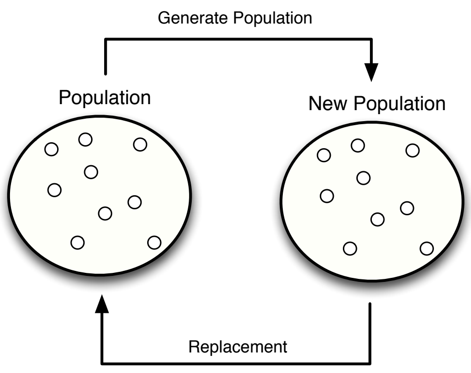


Figure 8.2: Population-based Metaheuristics

The motivation of using metaheuristics in the payload configuration optimisation problems, arises from the difficulty to solve a significant amount of problem instances in an exact manner. As a consequence, several methods to overcome this limitation are proposed in this chapter. More precisely, a local search method is implemented and genetic algorithms are applied. Section 8.2 and Section 8.3 present in details the proposed approaches, whereas the experimental results are analysed in Section 8.4.

8.1 Problem Case Tackled and Formulation of the Objective Function

Our focus is on the initial configuration (IC) problem case. This problem is more difficult compared to the reconfiguration (RC) and restoration (RS) cases, as in the IC problem, a larger search space has to be efficiently explored. On the contrary, in the RC and RS cases, there are already connected channels that the engineers do not desire to interrupt in the new configuration and thus the search space of the optimsation problem is smaller.

Furthermore, the aim is to minimise the length of the longest channel path because it has been experimentally shown in Chapter 6 to be the most difficult operational objective to solve exactly.

For the application of metaheuristics the focus in this thesis is on the 50 switches payload. For this payload size, there are instances that can not be solved exactly, therefore approximate methods are required. Besides, for many instances of this size, the exact solutions are known and this allows us to better analyse and compare the quality of solutions obtained by the metaheuristics. The use of a realistic payload of 50 switches implies that theoretically the maximum length of a path can be 100 (one switch can be used 2 times by a channel path). Therefore, to assign a fitness value to the candidate solutions, we propose the following objective function:

$$F = r_c + \frac{lpl}{1000} \tag{8.1}$$

where r_c is the number of selected channels that have not been connected to an amplifier (the number of the paths that have not been found) and lpl is the longest path length, i.e. the number of switches used in the longest path that has been constructed. Using F we consider thus a minimisation problem. Such an objective function permits to penalise unsatisfactory solutions proportionally to the number of channels that remain to be connected.

8.2 Switch Matrix Configuration Optimisation using Single-Solution Metaheuristics

The first proposed approach is based on single-solution based metaheuristics. A local search (LS) algorithm is proposed to find fast approximate payload configurations. This method is detailed in the following subsections.

8.2.1 Local Search Algorithm

The pseudocode of a simple local search (also called hill climbing algorithm [64]) is provided in Algorithm 3. At first, an initial solution is randomly generated and evaluated. A set of candidate neighbors is produced from the current solution, with the application of a defined operator (line 4). A neighbor is selected and evaluated (lines 5, 6) and if it improves the current solution it is selected to replace the current solution (line 8). In this case, the first candidate neighbor that improves the solution is selected. This process is repeated until no improvement in the quality of the solution occurs. The evaluation of the solutions is performed with a greedy method that is analysed in Section 8.2.3.

Algorithm 3 Pseudocode of the Local Search Algorithm

```
 1: Generate initial solution $s$
 2: Evaluate($s$)
 3: Do
 4:   Generate Candidate Neighbors ($N(s)$)
 5:   Select random non-visited $s' \in N(s)$
 6:   Evaluate($s'$)
 7:   If $s'$ better than $s$:
 8:       $s = s'$
 9:   else if exists non visited neighbor:
10:       go to 5
11:   else
12:       break
13: While better solution found
14: Output Final solution.
```

8.2.2 Solution Representation and Operators

Given a set of channels to connect the local search addresses the considered problem as a permutation problem. Indeed, the ordering of channels to connect will influence the quality of the final solution as each channel that is connected is occupying an available amplifier and is fixing the positions of the switches it uses. As a consequence, the next channels to connect, are restricted in both their final destination (amplifier) and the available switch positions that can use.

For n channels to connect, the solution of the local search algorithm is represented as a permutation of size n. Starting from a random solution (random permutation), at each iteration the current solution is replaced by a neighbour. The neighbors are generated by applying the exchange operator, where two arbitrarily selected elements are swapped to obtain a new solution as shown in Figure 8.3.

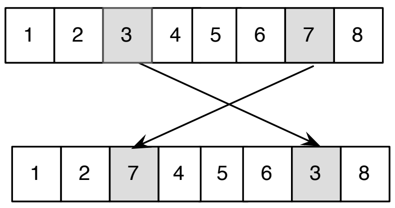


Figure 8.3: Exchange operator example.

The selection of the candidate solution is performed using the first improvement strategy. In this strategy the first neighbor that is better than the current solution is chosen. Other potential strategies that could be used, is the best improvement strategy, where the best neighbor is selected. The selection of the first improvement strategy, which implies that the exploration of the neighborhood is not exhaustive, is justified from the fact that the purpose is to obtain a fast approximate solution to the problem.

8.2.3 Solution Evaluation

To evaluate each solution a greedy method has been developed. Given the ordering of the channels to connect, the greedy method constructs the required paths and returns the value of the fitness function. The switch matrix can be represented as a graph $G = (V, E)$, where V is the set of payload components (switches, channels, amplifiers) and E the set of connectors (links) between them. The length of each link is considered to be equal to 1. Each component $u \in V$, has some coordinates (x, y) and at most 4 neighbour nodes (vertices) namely u_s, u_w, u_e, u_n, where s,w,e,n stand for south, west, east and north respectively. The pseudocode of the greedy method used for finding the path for each channel is provided in Algorithm 4.

Algorithm 4 Pseudocode of a greedy method to construct the channel paths

```
1:  for each channel c to connect do
2:      channel node c; path_c = {}
3:      save_All_neighbours()
4:      dest = closest_to_c_available_amplifier
5:      current = switch_node_neighbouring_to_c
6:      path_c.add(current)
7:      prec_node = channel node c
8:      while (current != dest) do
9:          next_node = closest_to_dest_available_neighbour()
10:         if (next_node) then
11:             updateNeighbourhood(prec_node, current, next_node)
12:             prec_node = current
13:             current = next_node
14:             path_c.add(current)
15:         else
16:             path_c.clear
17:             break
18:         end if
19:     end while
20:     if (path_c is empty) then
21:         reset_All_neighbours()
22:     else
23:         dest.available = false
24:         return path_c
25:     end if
26: end for
```

Initially, the destination amplifier of the channel is chosen to be the closest available amplifier based on the Manhattan distance. It is assumed that any channel can be connected to any amplifier. If this is not the case, the destination will be chosen among the suitable amplifiers for each channel. The next step is to set as current node the unique switch that is neighbor to the channel and the current node is added to the path (lines 5, 6). The channel node is then set as precedent node (line 7). To find the next node, the available neighbors of the current node are retrieved and the closest one to the destination is selected (line 9). Given the localities of the next and the precedent nodes compared to the current node, the position of the current switch is defined. The *updateNeighbourhood* function will then update the available neighbours (line 11). In this function, the current switch defines which of its neighbours are available from now on for future use. Besides, each of the neighbours of the current switch determines whether the current switch is still considered as one of the available neighbours. The same process is repeated until the destination is reached. As long as a channel path is constructed, the used amplifier is set as not available (line 23) and the path is saved. If the path can not be found, the neighbours of each switch are reset to the initial status (line 21) for being used by the next channel to connect. Figure 8.4 illustrates the integration between the greedy algorithm and the local search.

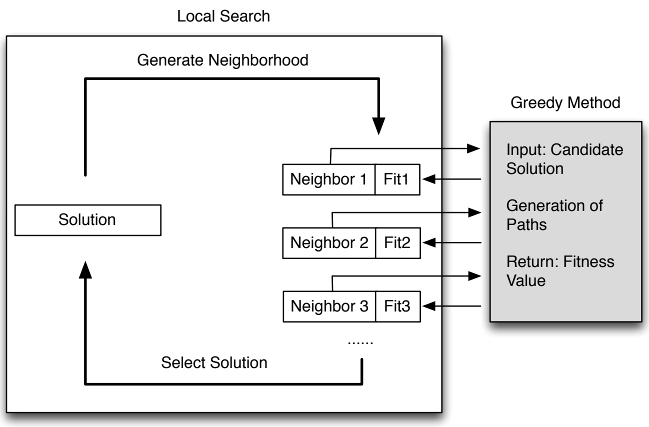


Figure 8.4: Integration of the greedy method with the local search

8.3 Switch Matrix Configuration Optimisation using Population-based Metaheuristics

The second approach is based on evolutionary algorithms. The main search components for designing an evolutionary algorithm are the solution representation, i.e. how is the solution to the optimisation problem encoded, the initialisation of the starting population, the selection strategy, i.e. which parents will be chosen for reproduction, the reproduction strategy, i.e. the crossover and mutation operators, the replacement strategy for the new offsprings as well as the objective function and the termination condition.

As metaheuristics are applied for the first time to the considered problem, it was chosen to use a simple generational genetic algorithm. Furthermore, a more advanced genetic algorithm is applied, namely the cellular genetic algorithm, where a structured population is evolved. The motivation is to examine which metaheuristic provides the more suitable results and to integrate them on the hybrid algorithms that will be examined at the next chapter of this dissertation.

8.3.1 Generational Genetic Algorithm

Genetic algorithms (GAs) are stochastic population-based metaheuristics that have been successfully applied to many difficult optimisation problems and follow the principles of the darwinian theory [38]. They are based on the evolution of a population. The population is composed by candidate solutions, the individuals, which are associated to a fitness value based on a defined objective function. Initially, this population is usually generated randomly. At each iteration (generation), individuals are selected based on their quality and modified with the application of genetic operators (i.e. crossover, mutation) to generate new offsprings. A replacement scheme is finally applied to determine which individuals of the population will survive. This process is iterated until a stopping criteria holds, like a certain time limit or a number of generations. In Figure 8.5 the main functions of a genetic algorithm are represented.

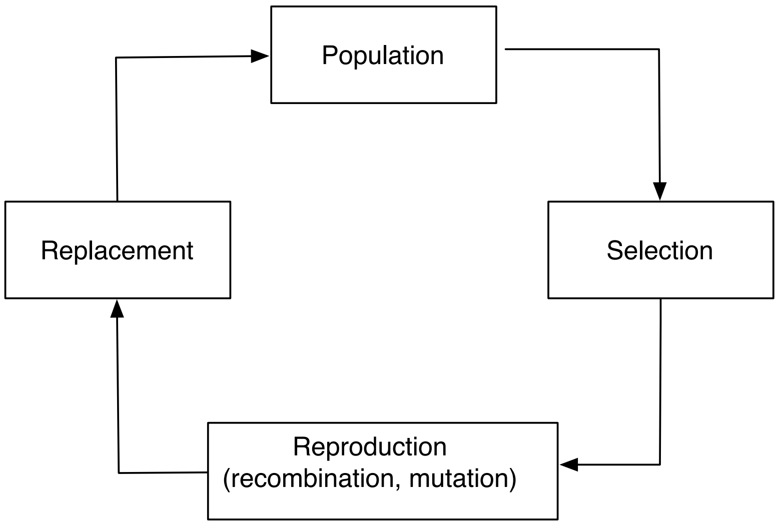


Figure 8.5: Iterative process of genetic algorithms

Generational GAs (genGAs) are a type of panmictic algorithm, i.e. individuals are grouped into a single structureless population also referred to as panmixia. Individuals can thus mate with any other individual in the population. The genGA is a (μ, λ)-GA, where the newly generated individuals are placed in an auxiliary population which will replace the current population when it is completely filled, i.e., when the number of newly generated solutions is equal to the size of this auxiliary population. In our case, the size of both the auxiliary and the current population is the same ($\mu = \lambda$).

Algorithm 5 presents the pseudocode of the generational GA. The population is first randomly initialized (line 2). Each generated individual is then evaluated using the fitness function defined for the tackled problem (line 3). The genetic loop then starts in line 4 until some predefined termination condition is met, e.g. a number of fitness function evaluations. "Parent" individuals are selected using some stochastic selection operator to construct the mating pool (line 6). Genetic variation is then ensured by the crossover (also called recombination) and mutation operators, both applied with some probability, which permit to visit other search space regions (line 7 and 8). The obtained offspring is then evaluated and inserted into the auxiliary population (line 9 and 10). The offspring population will become the current one once full (line 12).

Algorithm 5 Pseudocode of a canonical genGA

```
1:  proc Evolve(genga)    // Parameters of the algorithm in 'genga'
2:  GenerateInitialPopulation(genga.pop);
3:  Evaluation(genga.pop);
4:  while ! StopCondition() do
5:     for i←0 to genga.popSize do
6:        parents ← Selection(genga.pop);
7:        offspring ← Recombination(genga.Pc,parents);
8:        offspring ← Mutation(genga.Pm,offspring);
9:        Evaluation(offspring);
10:       Add(auxiliary_pop,offspring);
11:    end for
12:    genga.pop ← auxiliary_pop;
13: end while
14: end proc Evolve
```

8.3.2 Cellular Genetic Algorithm

Contrary to panmictic GAs like the generational GA, the cellular genetic algorithm (cGA) [9] uses a structured population. Individuals are spread in a two dimensional toroidal mesh and are only allowed to interact with their neighbors. An illustration of the cGA breeding loop is presented in Figure 8.6 with individuals arranged on a 5X5 toroidal grid and the neighborhood of the center individual, linear 5, is presented as dashed lines. Other examples of typically used neighborhoods is provided in Figure 8.7, where the label Ln (linear) is used for neighborhoods composed by the n nearest neighbors in a given axial direction (north, south, west and east), while the label Cn (compact) is used to designate the neighborhoods containing the $n-1$ nearer individuals to the considered one (in horizontal, vertical and diagonal directions).

A canonical cGA follows the pseudo-code presented in Algorithm 6. The population is usually structured in a regular grid of d dimensions ($d = 1, 2, 3$) and a neighborhood is defined on it. The algorithm iteratively considers as current each individual in the grid (line 5) and individuals may only interact with individuals belonging to their neighborhood (line 6), so parents are chosen among the neighbors (line 7) with a given criterion. Crossover and mutation operators are applied to the individuals (lines 8, 9) with probabilities P_c and

P_m, respectively. Afterwards, the algorithm computes the fitness value of the new offspring individual (or individuals) (line 10) and inserts it (or one of them) instead of the current individual in the population (line 11) following a given replacement policy. This loop is repeated until a termination condition is met (line 4).

Algorithm 6 Pseudocode for a canonical cGA

```
1:  proc Evolve(cga)        //Algorithm parameters in 'cga'
2:  GenerateInitialPopulation(cga.pop);
3:  Evaluation(cga.pop);
4:  while ! StopCondition() do
5:     for individual ← 1 to cga.popSize do
6:        n_list←Get_Neighborhood(cga,position(individual));
7:        parents←Selection(n_list);
8:        offspring←Recombination(cga.Pc,parents);
9:        offspring←Mutation(cga.Pm,offspring);
10:       Evaluation(offspring);
11:       Add(position(individual),offspring,cga);
12:    end for
13: end while
14: end proc Steps_Up;
```

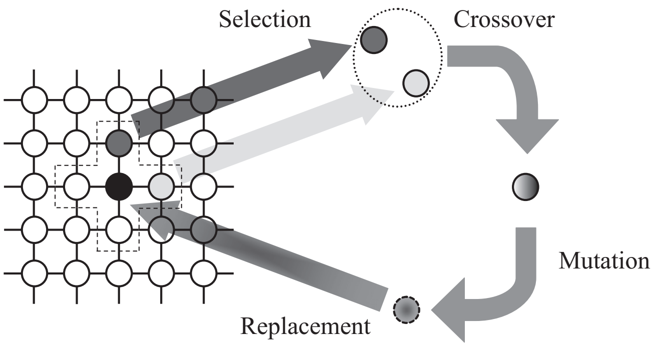


Figure 8.6: cGA reproduction cycle with 5 × 5 population and L5 neighborhood.

8.3.3 Solution Representation and Operators

One individual is represented as the set of positions of all the switches used in the payload switch matrix. A binary encoding is used where a switch position is encoded using two bits. The binary vector is thus of size $2 * n$, with n the number of switches in the payload swich matrix. An example of such a solution encoding is provided in Figure 8.8 where a vector of 8 bits is used to describe the position of 4 switches.

The initial population of individuals is generated randomly. Concerning the selection strategy, the tournament selection was applied, that consists in randomly selecting k individuals. The parameter k is called the size of the tournament group. A tournament is then applied to the k members of the group to select the best one. A binary tournament selection

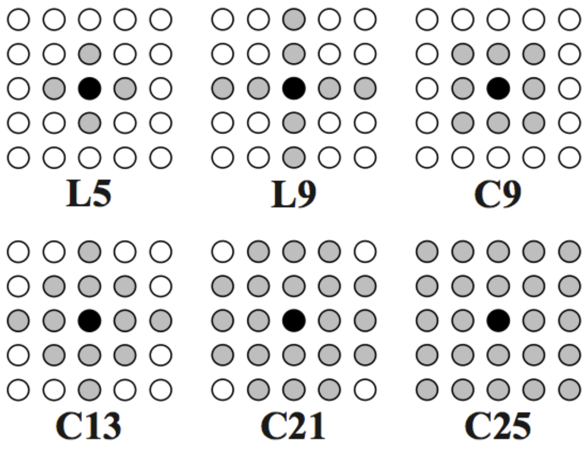


Figure 8.7: Typically used neighborhoods in cEAs

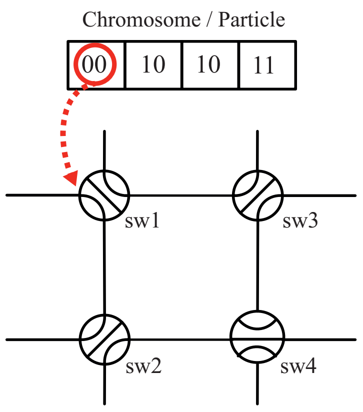


Figure 8.8: Solution encoding example.

(i.e. $k = 2$) was used in this work and an example of such a strategy is provided in Figure 8.9, for a minimisation problem.

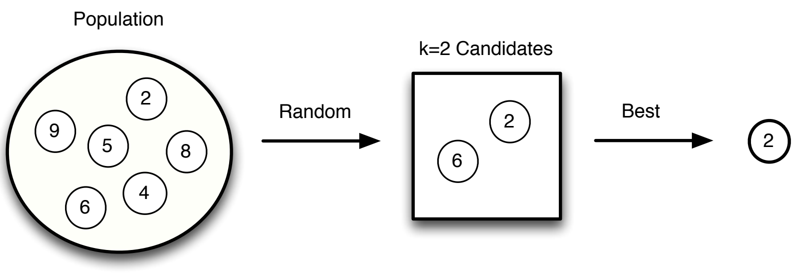


Figure 8.9: Example of a binary tournament selection strategy.

During the reproduction, operators such as the mutation and the crossover are applied.

The probability p_m defines the probability to mutate each element of the representation. In the binary representation, the commonly used mutation is defined as the flip operator, which is applied in our work. The basic crossover operator is the 1-point crossover, where a crossover point k is randomly selected and the two parent chromosomes are interchanged at this point to produce the two offsprings. In the n-point crossover, n crossover sites are randomly selected. An illustration example of the 1-point and 2-point crossover operators is provided in Figure 8.10, whereas in the uniform crossover, each element of the offspring is selected randomly from either parent [64].

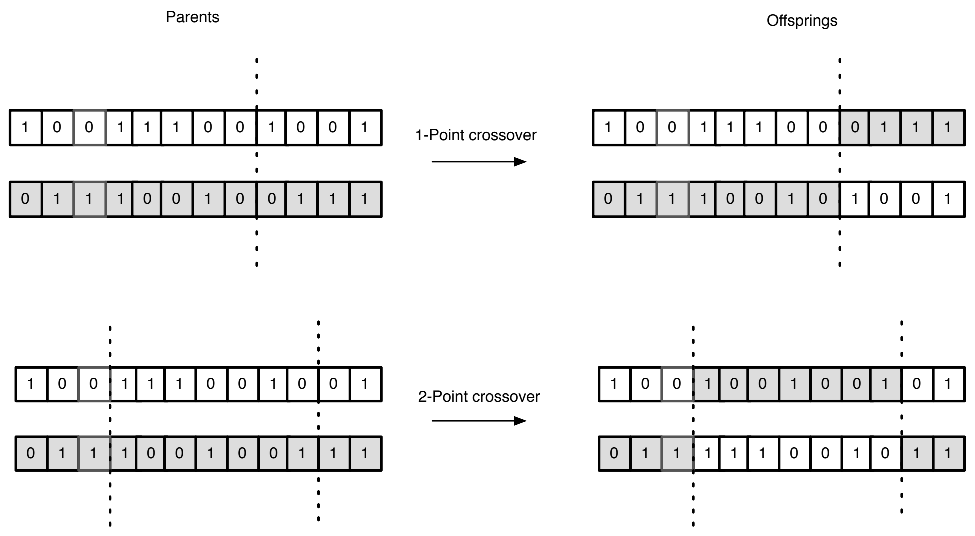


Figure 8.10: Examples of 1-point and 2-point crossover.

8.3.4 Solution Evaluation

Each individual represents a static graph. From the switch matrix topology and the positions of the switches the graph can be generated as shown in Figure 8.11. Each switch is represented with 4 vertices in the graph. The edges between those 4 vertices are generated depending on the position of the switch. The edges between the different switches are statically defined, based on the topology of the switch matrix. Using this corresponding graph, each individual is evaluated. If a connected component of the graph includes a channel vertex and an amplifier vertex then the path is valid. A solution is thus valid if it exists such a connected component for each selected channel. Due to the payload design, a valid connected component can only represent a unique path from channel to amplifier and its length will be equal to its total number of used switches.

8.4 Experimental Results

The experimental results of the described approximate methods are presented and analysed in this section. At first, information about the experimental setup is provided and in Section 8.4.2 the numerical results are detailed.

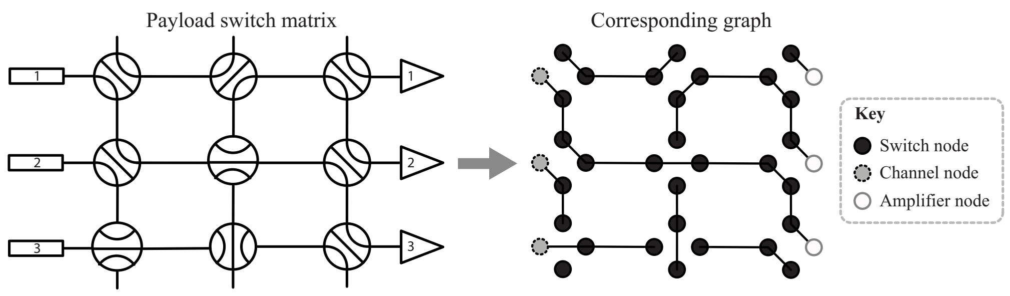


Figure 8.11: Example of switch matrix representation as a graph.

8.4.1 Experimental Setup

All the experiments were carried out using the HPC facility of the University of Luxembourg [70], each algorithm run being executed on a single CPU core of an Intel Xeon L5640 at 2.26GHz. The operating system was Debian GNU/Linux 6.0.9 and kernel 3.2.0-0.bpo.4-amd64. No core affinity was set. The implementation of the metaheuristic algorithms was done using the ParadisEO framework v.1.3 [27] . The parameters of the local search and of the genetic algorithms are provided in Table 8.1. The selection strategy is the first improvement strategy and the neighbourhood is generated using the exchange operator. For the cellular genetic algorithm, we used default parameters, i.e. two point crossover with probability p_c=0.8 and bit-flip mutation with $p_m = \frac{1}{chrom\_length}$. One parent is the current individual for cGA and the other parent is selected using the binary tournament strategy. The neighbourhood used is L5.

	Term. condition	10 min
LS	Algorithm	Hill Climbing
	Selection Strategy	First Improvment
	Neighbor operator	Exchange
genGA, cGA	Population	49, 7×7 (cGA)
	Selection	Binary tournament (BT), Current indiv. +BT (cGA)
	Neighborhood	L5 (cGA)
	Crossover	DPX, p_c=0.8
	Mutation	bit flip, $p_m = \frac{1}{chrom\_length}$
	Replacement strategy	Replace if better (cGA)
	Elitism	1 individual

Table 8.1: Algorithm Parameters

In Table 8.2, information about the tackled payload instances is provided. The experiments were conducted on a switch matrix with 50 switches of type R and 23 amplifiers (therefore the maximum number of channels to connect is 23). The 30 different sets of channels to connect of size 8, 13, 18 and 23 to connect are the same as the ones used in the exact methods.

An instantiation of the Experimental Framework is shown on Figure 8.12. The operational initial configuration problem case is considered and as objective the minimisation of the

Numb. of Switches	50
Switch Type	R type
Numb. of Amplifiers	23
Channels to Connect	8,13,18,23
	30 random sets per size

Table 8.2: Payload instances

longest path length.

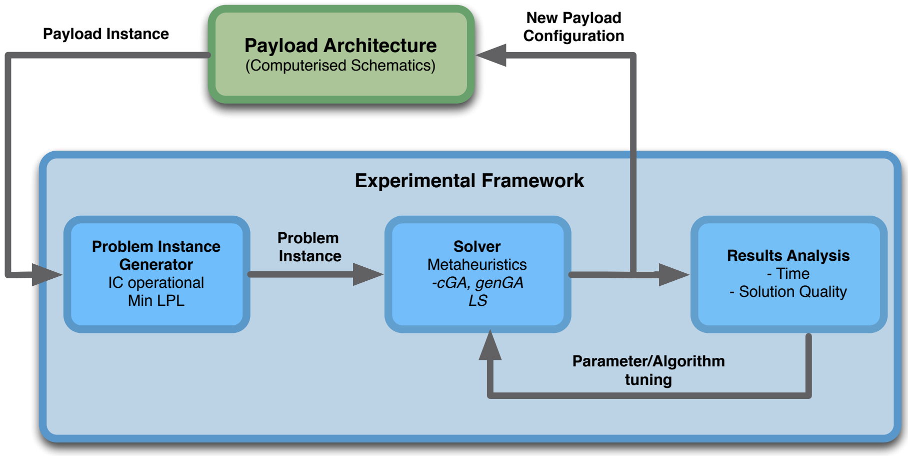


Figure 8.12: Instantiation of the experimental framework for minimising LPL in the initial configuration operational case using metaheuristics.

8.4.2 Numerical Results

The experimental results are summarised in Table 8.3 and Table 8.4 where the best values are highlighted with a grey background. Concerning the comparison between LS, genGA and cGA, the hit rate, that denotes in this case the percentage of instances where valid solutions were found, is displayed in Table 8.3. A valid solution is a solution where all paths are constructed for all the required channels to connect. As demonstrated from these results, cGA outperforms the other methods in all cases. More precisely, the hit rate of cGA is 100% when connecting 8, 13 and 18 channels and 80.888% for the cases of 23 channels to connect. For genGA the hit rate is also 100% when 8, 13 and 18 channels are connected but 79% for the cases of 23 channels. For the LS the hit rate is only 90%, 42.222%, 23.333% and 4.888% for 8, 13, 18 and 23 channels to connect respectively.

In terms of fitness value, cGA outperforms genGA and LS as well in all cases. The average and standard deviation of the fitness values of these algorithms are presented in Table 8.4. In more details, cGA has fitness 0.00195 and 0.00317 for the cases of 8 and 13 channels to connect compared to 0.00197 and 0.00320 obtained by genGA. When 18 and 23 channels are connected, cGA reached 0.00367 and 0.19879 respectively compared to 0.00371 and 0.21874

Channels	LS	genGA	cGA
8	90	100	100
13	42.222	100	100
18	23.333	100	100
23	4.888	79	80.888

Table 8.3: Hit Rate(%) Comparison

for genGA. The poorest performance is obtained in all cases by the LS alone. For instance, its fitness for connecting 8 channels is 0.10346 and 2.84213 for the cases of 23 channels to connect.

Channels	LS	genGA	cGA
8	$0.10346_{\pm 0.3}$	$0.00197_{\pm 0.0006}$	$0.00195_{\pm 0.0006}$
13	$0.69998_{\pm 0.69}$	$0.00320_{\pm 0.0006}$	$0.00317_{\pm 0.0006}$
18	$1.18435_{\pm 0.88}$	$0.00371_{\pm 0.0007}$	$0.00367_{\pm 0.0007}$
23	$2.84213_{\pm 1.33}$	$0.21874_{\pm 0.4}$	$0.19879_{\pm 0.4}$

Table 8.4: Fitness Comparison

In order to evaluate the quality of the solutions obtained by the population-based metaheuristics and the optimal solutions found using the ILP based exact method we propose Table 8.5. In this table, the optimal values in switch crossings, considering the termination condition of 120 hours, are presented. The cGA outperforms genGA in all cases for these instances. More precisely, for the sets of 8 channels to connect, the average fitness of genGA is 0.00188 compared to 0.00187 obtained by cGA. For the instances of 13 channels to connect, the fitness of genGA 0.00291, whereas cGA has fitness 0.00284. Finally, for the sets of 18 and 23 channels to connect genGA has average fitness 0.00357 and 0.19940 compared to 0.00350 and 0.19370 obtained by cGA respectively.

However, in all cases cGA provided solutions with lower quality compared to the optimal ones. The difference in fitness between the exact and the cGA increases with the number of channels to connect. In more details, for the cases of 8 channels to connect, the exact has fitness 0.00185 compared to 0.00187 for cGA. The difference in fitness increases for the cases of 13 is 0.00021, whereas for the cases of 18 channels to connect the difference in fitness increases to 0.00056 and for the cases of 23 channels to connect a significant difference of 0.18976 occurs.

Although cGA obtained better results than genGA in terms of fitness and hit rate, this was not detected as statistically significant after applying the Wilcoxon test [73]. The test stated though that LS proposes significantly worse results (with 95% confidence) than genGA and cGA for all the sizes of channels to connect. This is represented in Table 8.6 where it is depicted by (▲) the fact that genGA and cGA outperform LS with statistical confidence.

The convergence time, that denotes the average time needed by genGA and cGA to find

Channels	Exact	genGA	cGA
8	0.00185	0.00188	0.00187
13	0.00263	0.00291	0.00284
18	0.00294	0.00357	0.00350
23	0.00394	0.19940	0.19370

Table 8.5: Fitness Comparison on the commonly solved instances.

	cGA	genGA	LS
cGA	–	–	▲
genGA		–	▲

Table 8.6: Statistical confidence for the performance of the metaheuristics.

the best solution, is another important aspect that indicates the better performance of cGA. In all cases cGA converged faster compared to genGA. An illustration of the difference in the convergence time between these algorithms is provided in Figure 8.13.

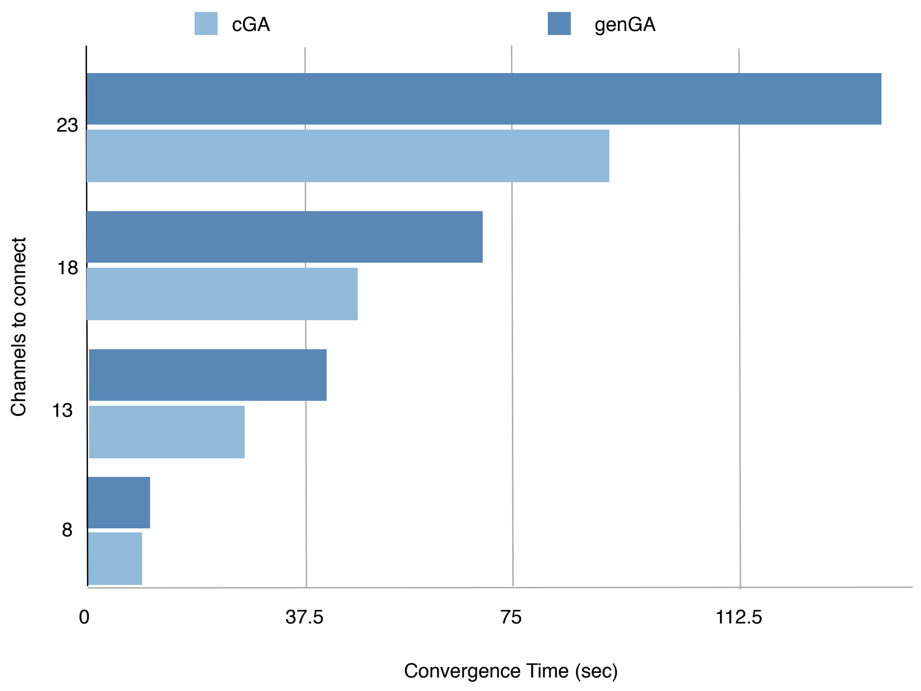


Figure 8.13: Comparison of the Average Convergence Time between genGA and cGA.

In more details, for the cases of 8 channels to connect, cGA converged after 9.41 seconds, whereas genGA converged after 11.112 seconds. When 13 channels are connected, cGA converged after 26.94 seconds compared to 42.104 seconds required by genGA. Lastly, for the cases of 18 and 23 channels to connect, genGA converged in average after 68.774 and 138.126 seconds compared to only 46.81 and 91.85 seconds required by cGA respectively. The highest difference in the convergence time between the two algorithms is 46.41 seconds and occurs for the case of 23 channels to connect.

8.5 Summary

This chapter investigates approaches for applying single-solution and population-based metaheuristics to the payload switch matrix configuration optimisation problem. Metaheuristics are applied for the first time to the considered problem. The focus is on minimising the length of the longest path considering the initial configuration problem case. To this scope, a local search (LS) method has been proposed where the problem is tackled as a permutation one. Furthermore, a generational genetic algorithm (genGA) and a cellular genetic algorithm (cGA) have been compared. For these evolutionary methods, each individual is represented as the set of positions of all the switches of the payload switch matrix.

It was experimentally demonstrated that the cGA provided the most promising results in terms of hit rate (percentage of successfully solved instances) and solution quality. Furthermore, the average convergence time of cGA was lower in all cases compared to genGA. The poorest performance is obtained by LS. After comparing the solutions of cGA to the optimal ones obtained by the ILP based exact method, a degradation in the quality of the solutions is remarked. Consequently, the following chapter focuses on the implementation of hybrid algorithms. These approaches integrate the proposed metaheuristics and the ILP based exact method. The purpose is at first to improve the quality of the solutions obtained by the metaheuristics and additionally to reduce the computational time required by the ILP based exact method.

Chapter 9

Hybridisation

Contents

Hybrid metaheuristics, that have been widely studied during the last years, combine different optimisation techniques, such as metaheuristics with other metaheuristics and/or with exact methods, in order to improve the overall performance of the optimisation algorithms. The best results for many optimisation problems are obtained by hybrid methods [63].

This chapter proposes hybrid algorithms for the payload switch matrix configuration optimisation problem. At first, an overview and a taxonomy of hybrid methods is presented in Section 9.1. The three proposed hybrid algorithms are detailed in Sections 9.2.1, 9.2.2 and 9.2.3 respectively. Their effectiveness is indicated in Section 9.3 where the experimental results are presented and analysed.

9.1 Hybrid Metaheuristics

A state of the art in hybrid metaheuristics, in a wide range of application domains such as data mining or routing problems, is provided in [65]. A taxonomy of hybrid metaheuristics has been introduced in [63] in order to provide a common classification mechanism for these techniques. Figure 9.1 provides an illustration of this taxonomy, which is divided in hierarchical and flat classification.

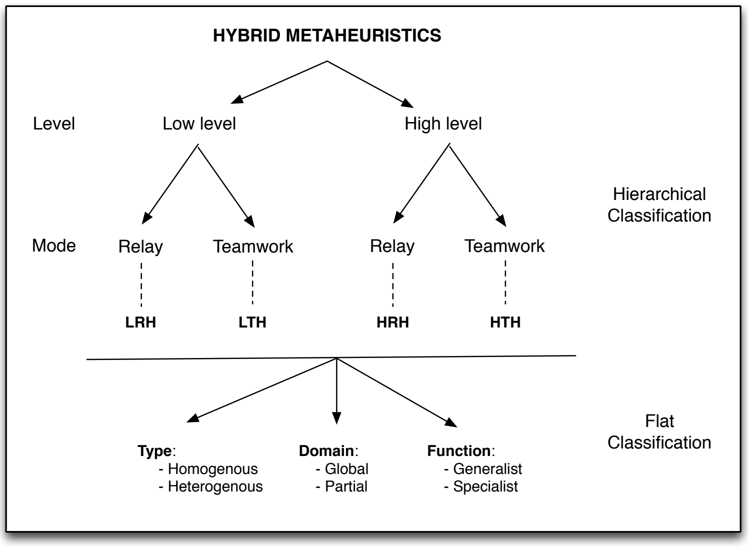


Figure 9.1: Classification of hybrid metaheuristics [63].

Concerning the hierarchical classification, at first a distinction is done between low-level and high-level hybridisations. The low-level hybrids occur when a given function of a metaheuristic is replaced by another metaheuristic. As an example, local search can be used for the mutation operator of a genetic algorithm. On the other hand, in the high-level hybrid algorithms, there is no influence to the internal functions of a metaheuristic. Besides, one can distinguish between relay and teamwork metaheuristics. In the first case, the metaheuristics are applied one after another whereas in teamwork hybridisation the components of the hybrid collaborate in parallel, each one performing a search in a solution space [65].

According to this taxonomy, four main classes are derived, as illustrated in Figure 9.1, namely the LRH (Low-Level Relay Hybrid), the LTH (Low-Level Teamwork Hybrid), the HRH (High-Level Relay Hybrid) and the HTH (High-Level Teamwork Hybrid) class. There exist many proposed hybrid algorithms in the literature that belong to each of the above classes. For example, the authors in [67] apply a hill climbing as a mutation operator in a genetic algorithm or in [54] a local search is applied within a crossover function. Both works make part of the LTH class. In [47] genetic algorithms follow the application of simulated annealing to improve the obtained solutions. One example of High-Level Teamwork Hybrid is the island model, where the population is partitioned into small subpopulations and a genetic algorithm evolves each subpopulation [30].

Concerning the flat classification, in the homogeneous hybrids the same metaheuristics are applied whereas in the heterogeneous hybrids different metaheuristics are used. Another distinction is done between the global and the partial hybrids. In the first case, all the algorithms explore the same search space, whereas in partial hybrids the methods perform search in subspaces. Finally, the specialist hybrids combine algorithms that solve different problems, for example, using a metaheuristic to optimise the parameters of another metaheuristic, whereas in the general hybrids the same optimisation problem is solved by the components of the hybrid [65].

Especially for the cases where hybrid algorithms combine metaheuristics with exact methods, many approaches have been proposed for each of the above mentioned classes. Several examples can be found in [65, 58]. For the HRH hybrids, as mentioned in [65], classical approaches where information is provided to the exact methods by the metaheuristics, are the following:

- Upper bound provided by the metaheuristics to the exact method.
- Partitioning of decision variables. The metaheuristics are used to fix the values of a subset of the decision variables and the exact method optimises the problem over the rest set of decision variables.
- Domain reduction. The metaheuristics are used for reducing the domain of values that the decision variables can have.

9.2 Solving the Payload Switch Matrix Configuration Problem Using Hybrid Algorithms

The motivation to implement hybrid metaheuristics for the satellite payload switch matrix configuration optimisation problem is to overcome the limitations related at first to the solution quality when metaheuristics are individually applied to the problem and also to the low hit rate when the exact method is used. To this scope, three hybridisation schemes are proposed. The schemes integrate the local search (LS), the cellular genetic algorithm (cGA) and the ILP based exact method. An illustration of these schemes with their components is provided in Figure 9.2. The local search and the cellular genetic algorithm have been presented in the Section 8.2.1 and Section 8.3.2 respectively. The ILP model used is the one presented in 6.3.1 for minimising the length of the longest channel path. In the following three subsections the three hybrids are presented in details.

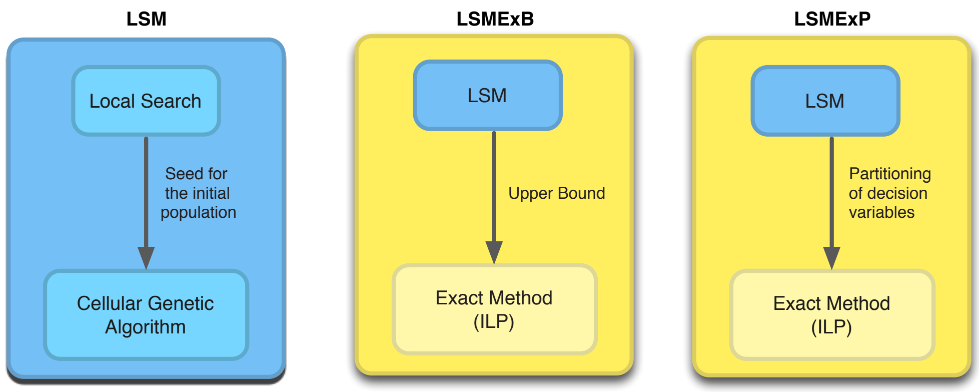

LSM
Local Search
Seed for the initial population
Cellular Genetic Algorithm
LSMExB
LSM
Upper Bound
Exact Method (ILP)
LSMExP
LSM
Partitioning of decision variables
Exact Method (ILP)

Figure 9.2: The three hybrid schemes and their components.

9.2.1 Local Search and Metaheuristics (LSM)

In the first hybrid, referred to as LSM, instead of generating randomly the whole initial population of the cGA, a seed individual, that is provided by the local search, is inserted in the initial population. The pseudocode of LSM is presented in Algorithm 7. At first the problem is solved using the local search (line 2). From the final solution of the local search, the paths and the switch positions are known and thus this solution is converted to the corresponding representation used by the cGA (line 3) and inserted in the initial population (line 5). The remaining individuals are created uniformly at random. Finally, the cGA is applied and the best individual is returned.

Algorithm 7 Pseudocode of LSM

```
1: Input: n channels to connect
2: Exec_LS(n)
3: c ← Get_Indi(LS_final_sol)
4: GenerateInitialPopulation(cga.popSize - 1)
5: Add(cga.pop, c)
6: Exec_cGA()
7: Return: best individual
```

The LS and cGA are applied in sequence. The global problem is solved by both methods. This hybrid is classified as High Level Relay Hybrid (HRH)(heterogenous, global, general).

9.2.2 LSM and Exact Bound (LSMExB)

In this hybrid, referred to as LSMExB, the solution provided by the LSM is considered as an upper bound for the exact method. This collaborative scheme was applied aiming to improve the hit rate of the exact method. The pseudocode of this method is shown in Algorithm 8. At first, the LSM method is executed (line 2). Based on the obtained solution, the upper bound of the objective function is set in the ILP model (line 4). Finally the exact method is called (line 5).

Algorithm 8 Pseudocode of LSMExB

```
1: Input: n channels to connect
2: Exec_LSM(n)
3: obj_bound ← Get_Bound(LSM_final_sol)
4: ilp_model ← Set_Bound(obj_bound)
5: Exec_Exact(ilp_model)
6: Return: final solution
```

The global problem is solved by both methods. The LSM and the ILP based exact method are applied in sequence. This hybrid belongs to the class of High Level Relay Hybrid (HRH)(heterogenous, global, general).

9.2.3 LSM and Exact Partitioning (LSMExP)

In this hybridisation scheme, denoted as LSMExP, the size of the original problem solved by the exact method is reduced by partitioning the decision variables. As indicated in [64], the decision variables can be partitioned in two sets X and Y. Based on the solution generated by the metaheuristic (LSM) the variables of the set X will be fixed and the exact method will be called to optimise the problem over the set Y. The decision variables that are selected to be fixed are the positions of the switches used in a certain number of channel paths obtained from the final solution of LSM. Therefore, at first the problem is solved using metaheuristics and based on some generated paths the positions of the switches are set in the ILP model. The exact method is then called. Two different approaches of this hybrid algorithm are implemented and compared. Given n channels to connect, in the first approach, denoted as LSMExP_Rnd, a random set of $k \leq n$ paths obtained by LSM is selected. The switch positions used by this set of paths are fixed. In the second approach, denoted as LSMExP_Sh, the set of the k shortest paths obtained by LSM is always selected and the switch positions used by this set of paths are fixed.

The pseudocode of LSMExP is provided in Algorithm 9. At first the LSM hybrid is called. Based on the solution obtained by LSM a list of k (random or shortest) paths is selected (line 3). The switch positions used by these paths, are fixed in the ILP model (line 4) and finally the exact method is called. This hybrid is classified as High Level Relay Hybrid (HRH)(heterogenous, partial, general).

Algorithm 9 Pseudocode of LSMExP

```
1: Input: n channels to connect
2: Exec_LSM(n)
3: p_list ← Get_Paths(LSM_final_sol, k)
4: ilp_model ← Fix_Switch_Pos(p_list)
5: Exec_Exact(ilp_model)
6: Return: final solution
```

9.3 Experimental Results

The effectiveness of the hybrids compared to the individual application of their components is indicated in this section. At first, information about the experimental setup is provided and then the numerical results are detailed.

9.3.1 Experimental Setup

An instantiation of the experimental framework is provided in Figure 9.3. The initial configuration case is considered with the aim to minimise the length of the longest channel path.

All the experiments were carried out using the HPC facility of the University of Luxembourg, each algorithm run being executed on a single CPU core of an Intel Xeon L5640 at 2.26GHz. The operating system was Debian GNU/Linux 6.0.9 and kernel 3.2.0-0.bpo.4-amd64. No core affinity was set. The metaheuristic algorithms were implemented using the

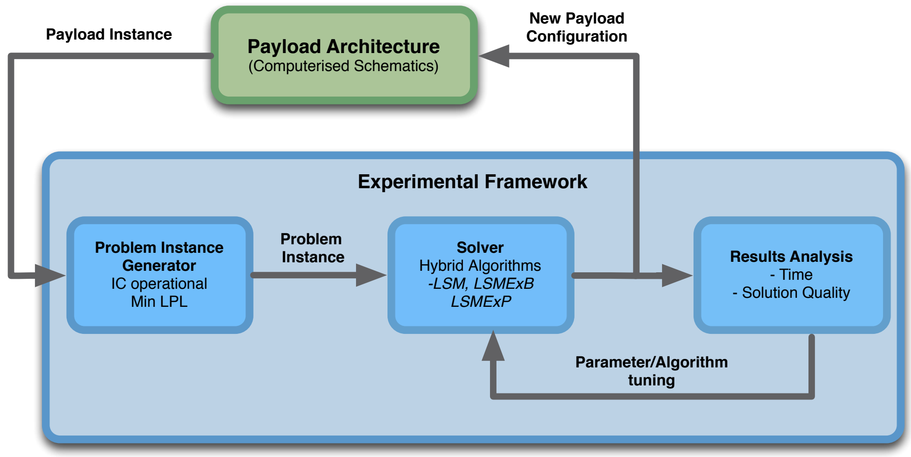


Figure 9.3: Instantiation of the experimental framework for minimising LPL in the initial configuration operational case using hybrid techniques.

ParadisEO framework v.1.3 [27] . For the exact method IBM ILOG CPLEX 12.0.0 solver was used [2]. The parameters of the local search and cGA are provided in Table 9.1.

	Term. condition	10 min
LS	Algorithm	Hill Climbing
	Selection Strategy	First Improvment
	Neighbor operator	Exchange
cGA	Population	49, 7×7
	Selection	Binary tournament (BT), Current indiv. + BT
	Neighborhood	L5
	Crossover	DPX, p_c=0.8
	Mutation	bit flip, $p_m = \frac{1}{chrom\_length}$
	Replacement strategy	Replace if better
	Elitism	1 individual
ILP	Model	ILP
	Solver	IBM ILOG CPLEX 12.0.0

Table 9.1: Algorithm Parameters

The selection strategy of the local search is the first improvement strategy and the neighborhood is generated using the exchange operator. For the cellular genetic algorithm, we used default parameters, i.e. two point crossover with probability p_c=0.8 and bit-flip mutation with $p_m = \frac{1}{chrom\_length}$. One parent is the current individual for cGA and the other parent is selected using the binary tournament strategy. The neighborhood used is L5.

The experiments were conducted on a switch matrix with 50 switches of type R and 23 amplifiers (thus the maximum number of channels to connect is 23), as shown in Table 9.2 where information about the payload instances tackled is provided. The 30 different sets of channels of size 8, 13, 18 and 23 to connect are the same as the ones considered in Chapter 6.

Numb. of Switches	50
Switch Type	R type
Numb. of Amplifiers	23
Channels to Connect	8,13,18,23
	30 random sets per size

Table 9.2: Payload Instances

For the case of LSMExP_Rnd, given a size k of paths according to which the decision variables will be fixed, 30 different sets of paths of this size are selected uniformly at random, whereas for LSMExP_Sh the decision variables are fixed based on a single set of k shortest paths.

9.3.2 Numerical Results

This section analyses the performances of the proposed schemes related to the hit rate (percentage of successfully solved instances), the fitness value and the required computational time. In the next subsection, the LS, cGA and LSM methods are compared. In Section 9.3.2.2 the performances of the two hybrid algorithms LSMExP_Rnd and LSMExP_Sh are analysed. In all the conducted experiments the total termination condition was set to ten minutes as the operational phase is considered. When the experimental results concern the comparison of different methods, the best values are highlighted with a grey background.

9.3.2.1 Comparison of the Hybrid Algorithms

Concerning the comparison of the 3 metaheuristics, LS, cGA and LSM, the hit rate, which in this case denotes the percentage of instances where valid solutions were found, is displayed in Table 9.3. A valid solution is a solution where the paths have been constructed for all the required channels to connect. As demonstrated from these results, the first hybrid LSM performs better. More precisely, the hit rate of LSM is 100% when connecting 8, 13 and 18 channels and 82.333% for the cases of 23 channels to connect. For cGA the hit rate is also 100% when 8, 13 and 18 channels are connected but 80.888% for the cases of 23 channels to connect. For the LS the hit rate is 90%, 42.222%, 23.333% and 4.888% for 8, 13, 18 and 23 channels to connect respectively.

Channels	LS	cGA	LSM
8	90	100	100
13	42.222	100	100
18	23.333	100	100
23	4.888	80.888	82.333

Table 9.3: Hit Rate(%) Comparison of Hybrid Metaheuristics.

In terms of fitness, the hybrid LSM outperforms cGA as well in all cases. The average and standard deviation of the fitness values of the three metaheuristics are presented in Table 9.4.

Channels	LS	cGA	LSM	Conf.
8	$0.10346_{\pm 0.3}$	$0.00195_{\pm 0.0006}$	$0.00194_{\pm 0.0005}$	−
13	$0.69998_{\pm 0.69}$	$0.00317_{\pm 0.0006}$	$0.00304_{\pm 0.0005}$	−
18	$1.18435_{\pm 0.88}$	$0.00367_{\pm 0.0007}$	$0.00365_{\pm 0.0007}$	−
23	$2.84213_{\pm 1.33}$	$0.19879_{\pm 0.4}$	$0.18183_{\pm 0.381}$	▲

Table 9.4: Fitness Comparison of Hybrid Metaheuristics.

More precisely, LSM provides fitness values of 0.00194 and 0.00304 for the cases of 8 and 13 channels to connect, compared to 0.00195 and 0.00317 obtained by cGA. When 18 channels are connected the difference in fitness is small as LSM has average 0.00365 compared to 0.00367 obtained by cGA. For the sets of 23 channels to connect, LSM has fitness 0.18183 compared to 0.19879 for cGA. It can also be observed that LSM has smaller standard deviation compared to cGA in all cases except for the case 18 channels where the standard deviation is equal. The poorest performance is obtained in all cases by the LS alone. Statistical confidence in the results is assessed using the Wilcoxon test [73]. In the last column of Table 9.4, it is depicted by (▲) if LSM is statistically better compared to cGA and by (−) if no significant difference was found. LSM proposes significantly better results (with 95% confidence) compared to cGA for the case of 23 channels to connect. LSM and cGA perform better with statistical confidence compared to LS itself in all cases.

In order to evaluate the quality of the solutions provided by these approximate algorithms, their performance is compared on the instances solved by the exact method where the optimal value of the length of the longest path is known. Considering the termination condition of 120 hours, the optimal values in switch crossings is compared to the performance of cGA and LSM in Table 9.5. The better performance of LSM is indicated in most of the cases. For the case of 8 channels to connect, the average fitness of the exact method is 0.00185, the average fitness of LSM is 0.00186, whereas for cGA it is 0.00187. For 13 channels to connect, the average fitness of the exact method is 0.00263, whereas LSM has fitness 0.00278 and cGA 0.00284. When 18 channels are connected, for the instances that were solved exactly, cGA had better average fitness compared to the LSM. i.e, 0.00350 compared to 0.00356. As has been seen in Table 9.4 the performance of these two algorithms was similar for this case. Lastly, for the cases of 23 channels to connect, that is the case where LSM performed better with statistical confidence compared to cGA, on the instances where the optimal solution is found, LSM had average fitness 0.15412 compared to 0.19370 obtained by cGA.

Another essential aspect to compare between cGA and LSM is the convergence time, that denotes the average time needed by each of these methods to reach the best solution found. As can be seen in Table 9.6, LSM converges faster than cGA in all cases. The convergence time of LSM for the cases of 8 and 13 channels to connect is 5.26 seconds and 26.94 seconds respectively compared to 9.41 seconds and 27.98 seconds for cGA. For the case of 18 channels, LSM converged in average after 44.11 seconds whereas cGA converged in average after 46.81 seconds. For the case of 23 channels LSM converged in average after 79.48 seconds and cGA in average after 91.85 seconds.

Channels	Exact	cGA	LSM
8	0.00185	0.00187	0.00186
13	0.00263	0.00284	0.00278
18	0.00294	0.00350	0.00356
23	0.00394	0.19370	0.15412

Table 9.5: Fitness Comparison on the commonly solved instances.

Channels	cGA	LSM
8	9.41	5.26
13	27.98	26.94
18	46.81	44.11
23	91.85	79.48

Table 9.6: Average Convergence Time(sec) between cGA and LSM.

In Table 9.7 the comparison results between the ILP based exact method and the LSLMExB and LSLMExP are presented. For these two hybrid methods, LSM runs for the time provided in Table 9.6 and then the ILP based exact method is called for the remaining time, so that the total termination condition is not violated. LSLMExB slightly improved the hit rate of the exact method for the case of 13 channels to connect as the hit rate is 16.666% compared to 13.333% achieved by the ILP based exact method. Before analysing the two different approaches of LSMExP, namely the LSMExP_Rnd and LSMExP_Sh, that are related to the number of paths to fix and the techniques according to which they are selected, for this initial evaluation of the method it was chosen to set for each channel size, the positions of the switches used by 4, 7, 10 and 12 randomly selected paths respectively. The LSMExP hybrid provided better results as it achieved 89.222% hit rate for the maximum case of connecting 23 channels. For 18 channels to connect, LSMExP solved 89.444% of instances compared to 6.666% of LSMExB. With LSMExP, the process is significantly accelerated since the exact method is optimising over a subset of the decision variables. In Table 9.8, the average fitness is compared between LSM, LSMExB and LSMExP for the commonly solved instances. It is observed that LSMExP provided the optimal solution for these instances (same fitness with LSMExB) and LSM provided solutions of less good quality.

9.3.2.2 Comparison of LSMExP_Rnd to LSMExP_Sh

The comparison results between LSMExP_Rnd and LSMExP_Sh in terms of hit rate, fitness, computational time and percentage of infeasible cases are here presented. The focus is on the cases of 18 and 23 channels to connect where the hit rate of the ILP based exact method is very small, i.e. 6.666% and 0% respectively. Based on the convergence time displayed in Table 9.6, for the case of 18 channels, LSM runs for 44.11 seconds followed by the exact method for 555.89 seconds. For the cases of 23 channels to connect, LSM runs for 79.48 seconds followed by the exact for 520.52 seconds.

Channels	Exact		LSMexB		LSMExP	
	fitness	hit-rate	fitness	hit-rate	fitness	hit-rate
8	$0.00181_{\pm 0.0004}$	90	$0.00181_{\pm 0.0004}$	90	$0.00191_{\pm 0.0005}$	97.888
13	$0.00200_{\pm 0}$	13.333	$0.00260_{\pm 0.0005}$	16.666	$0.00300_{\pm 0.0005}$	93
18	$0.00200_{\pm 0}$	6.666	$0.00200_{\pm 0}$	6.666	$0.00363_{\pm 0.0008}$	89.444
23	–	–	–	–	$0.00487_{\pm 0.0009}$	89.222

Table 9.7: Fitness and HitRate(%)

Channels	LSMExB	LSM	LSMExP
8	$0.00181_{\pm 0.483}$	$0.00182_{\pm 0.0003}$	$0.00181_{\pm 0.474}$
13	$0.00260_{\pm 0.547}$	$0.00261_{\pm 0.515}$	$0.00260_{\pm 0.491}$
18	$0.00200_{\pm 0}$	$0.00200_{\pm 0}$	$0.00200_{\pm 0}$

Table 9.8: Fitness comparison on the commonly solved instances

9.3.2.2.1 Connecting 18 Channels.

The results related to the hit rate are displayed in Table 9.9. In the first column k represents the number of paths that is chosen to be fixed (starting from 2 fixed paths to 16 with a step of 2). The second column indicates the hit rate for LSMExP_Rnd and the last column provides the hit rate for LSMExP_Sh. A first remark is that with this approach a considerable percentage of instances is solved within the acceptable time. Even for the case of $k = 2$, LSMExP_Rnd managed to solve 27.666% of the instances. When k increases to 10, the hit rate of LSMExP_Rnd is 89.444% and when $k = 16$, it reaches 96.666%. The hit rate for LSMExP_Sh is lower compared to LSMExP_Rnd. This is explained from the fact that fixing random (and thus also longer) paths implies that more decision variables are fixed and therefore the search space is smaller. More precisely, when $k = 4$, LSMExP_Sh solved 26.666% of the instances, for 10 fixed paths the hit rate increases to 53.333% and for $k = 16$ it reaches 99.666%.

Table 9.10 provides the average and standard deviation of the fitness values for both methods. LSMExP_Sh has always smaller fitness therefore this approach provides solutions of better quality. A path of bad quality will not be fixed using LSMExP_Sh and shortest paths is more probable to be the optimal ones. For instance, when $k = 6$, LSMExP_Sh has fitness 0.00291 compared to 0.00350 for LSMExP_Rnd and when $k = 10$ LSMExP_Sh has average fitness 0.00306 compared to 0.00363. The fitness value increases with the number of fixed paths and the difference in fitness between LSMExP_Rnd and LSMExP_Sh becomes lower. For $k = 2$ the difference in fitness is 0.00075 and when $k = 16$ it is 0.00014.

For the instances of 18 channels to connect that were solved exactly by the ILP based exact method, LSMExP_Rnd and LSMExP_Sh have the same fitness value as the optimal ones. This can be seen also from Table 9.8 where LSM has found the optimal values for these instances. The better performance of LSMExP_Sh in terms of fitness was not detected as statistically significant.

The computational time required by the two approaches is provided in Table 9.11. As

k	LSMExP_Rnd	LSMExP_Sh
2	27.666	13.333
4	55.444	26.666
6	72.666	40
8	83	43.333
10	89.444	53.333
12	96.666	70
14	99.555	86.666
16	99.666	99.666

Table 9.9: Hit Rate(%) comparison between LSMExP_Rnd and LSMExP_Sh when connecting 18 channels.

k	LSMExP_Rnd	LSMExP_Sh
2	$0.00325_{\pm 0.0009}$	$0.00250_{\pm 0.0005}$
4	$0.00343_{\pm 0.0008}$	$0.00275_{\pm 0.0004}$
6	$0.00350_{\pm 0.0008}$	$0.00291_{\pm 0.0005}$
8	$0.00358_{\pm 0.0008}$	$0.00292_{\pm 0.0004}$
10	$0.00363_{\pm 0.0008}$	$0.00306_{\pm 0.0005}$
12	$0.00368_{\pm 0.0008}$	$0.00328_{\pm 0.0006}$
14	$0.00371_{\pm 0.0008}$	$0.00342_{\pm 0.0007}$
16	$0.00372_{\pm 0.0008}$	$0.00358_{\pm 0.0007}$

Table 9.10: Fitness Comparison between LSMExP_Rnd and LSMExP_Sh when connecting 18 channels.

expected LSMExP_Rnd performs faster. Starting from 2 fixed paths, LSMExP_Rnd requires 62.122 seconds and only 1.856 seconds for 16 fixed paths. For LSMExP_Sh the computational time is higher, with an average of 148.942 seconds for 2 shortest paths fixed and 25.568 seconds for 16 shortest paths fixed.

9.3.2.2.2 Connecting 23 Channels.

The hit rate is displayed in Table 9.12. In the first column, k denotes the number of paths that is chosen to be fixed (starting from 4 fixed paths up to 20 with a step of 2). The second column indicates the hit rate for LSMExP_Rnd and the last column provides the hit rate for LSMExP_Sh. It can be noticed that with these hybrids a considerable percentage of instances is solved within the time limit whereas the initial hit rate of the ILP based exact method was 0%. As an example, when $k = 4$, LSMExP_Rnd managed to solve 43.777%. When k increases to 10, the hit rate reaches to 80.555% and goes up to 95.555% when $k = 16$. For the last case of $k = 20$ the hit rate for LSMExP_Rnd reaches 94.888%.

The hit rate for LSMExP_Sh is smaller compared to LSMExP_Rnd in 7 out of 9 cases.

k	LSMExP_Rnd	LSMExP_Sh
2	$62.122_{\pm 115.043}$	$148.942_{\pm 174.215}$
4	$52.286_{\pm 105.657}$	$220.912_{\pm 218.007}$
6	$46.615_{\pm 102.033}$	$125.265_{\pm 100.174}$
8	$26.921_{\pm 72.648}$	$76.255_{\pm 114.729}$
10	$12.097_{\pm 45.454}$	$79.647_{\pm 99.284}$
12	$8.694_{\pm 35.598}$	$56.628_{\pm 84.960}$
14	$3.041_{\pm 15.055}$	$59.456_{\pm 78.171}$
16	$1.856_{\pm 18.059}$	$25.568_{\pm 59.531}$

Table 9.11: Time(sec) comparison between LSMExP_Rnd and LSMExP_Sh when connecting 18 channels.

k	LSMExP_Rnd	LSMExP_Sh
4	43.777	3.333
6	58.444	20
8	73.222	33.333
10	80.555	50
12	89.222	66.666
14	94	60
16	95.555	86.666
18	95.555	96.666
20	94.888	100

Table 9.12: Hit Rate Comparison between LSMExP_Rnd and LSMExP_Sh when connecting 23 channels.

More precisely, when $k = 4$, LSMExP_Sh solved only 3.333% of the instances whereas for $k = 10$ the hit rate of LSMExP_Sh increases to 50% and for $k = 16$ the hit rate reaches 86.666%. However, it has to be mentioned that for the cases of fixing 18 and 20 paths the hit rate for LSMExP_Sh is higher compared to LSMExP_Rnd, namely 96.666% and 100%, compared to 95.555% and 94.888% for LSMExP_Rnd respectively. This is explained by the fact that the rate of infeasible instances increases significantly for LSMExP_Rnd when 18 and 20 paths are fixed whereas for LSMExP_Sh the rate of infeasible instances remains 0% in all cases. Increasing the number of paths to fix, can also produce an infeasible problem. This is explained as fixing the switch positions used by a channel path may permanently block the connection of another channel in the switch matrix. The comparison of the rates of infeasible cases for LSMExP_Rnd and LSMExP_Sh are shown in Table 9.13. As can be seen, the percentage of infeasible cases increases with the number of fixed paths for LSMExP_Rnd. It is 0.222% when $k = 4$ and reaches 4.333% and 5.111% when 18 and 20 paths are fixed respectively. The corresponding rate for LSMExP_Sh is 0% in all cases, denoting that when the shortest paths are selected to be fixed, none problem instance is turned to infeasible. We can thus claim that fixing shortest paths is the best approach in terms of solution quality.

k	LSMExP_Rnd	LSMExP_Sh
4	0.222	0
6	0.222	0
8	0.555	0
10	1.222	0
12	2	0
14	2.222	0
16	3.888	0
18	4.333	0
20	5.111	0

Table 9.13: Percentage of infeasible cases between LSMExP_Rnd and LSMExP_Sh when connecting 23 channels.

The average and standard deviation of the fitness values for both approaches, are presented in Table 9.14. The average fitness of LSMExP_Sh is always smaller compared to LSMExP_Rnd which denotes that solutions of better quality are obtained with this method. For example for 6 fixed paths, LSMExP_Sh has fitness 0.00380 compared to 0.00459 for LSMExP_Rnd and after fixing 10 paths LSMExP_Sh has average fitness 0.00413 compared to and 0.00479. The same holds also for the other cases, for example when $k = 18$ LSMExP_Sh has average fitness 0.00489 compared to 0.00501 by LSMExP_Rnd. The better performance of LSMExP_Sh in terms of fitness was not detected as statistically significant.

k	LSMExP_Rnd	LSMExP_Sh
4	$0.00458_{\pm 0.0008}$	$0.00300_{\pm 0}$
6	$0.00459_{\pm 0.0008}$	$0.00380_{\pm 0.0004}$
8	$0.00467_{\pm 0.0008}$	$0.00410_{\pm 0.0003}$
10	$0.00479_{\pm 0.0009}$	$0.00413_{\pm 0.0003}$
12	$0.00487_{\pm 0.0009}$	$0.00440_{\pm 0.0006}$
14	$0.00495_{\pm 0.0009}$	$0.00461_{\pm 0.0007}$
16	$0.00500_{\pm 0.0009}$	$0.00480_{\pm 0.0008}$
18	$0.00501_{\pm 0.0010}$	$0.00489_{\pm 0.0009}$
20	$0.00502_{\pm 0.0010}$	$0.00500_{\pm 0.0009}$

Table 9.14: Fitness Comparison between LSMExP_Rnd and LSMExP_Sh when connecting 23 channels.

The average and standard deviation of the time is displayed in Table 9.15. LSMExP_Rnd performs always faster as the exact is optimising over a smaller set of decision variables compared to LSMExP_Sh. For example, for 4 fixed paths, LSMExP_Rnd required in average 65.227 seconds compared to 273.05 seconds for LSMExP_Sh, for 10 fixed paths LSMExP_Rnd required in average 31.459 seconds compared to 223.646 seconds for LSMExP_Sh and finally for 20 fixed paths LSMExP_Rnd required in average 1.742 seconds compared to 12.710 seconds for LSMExP_Sh.

k	LSMExP_Rnd	LSMExP_Sh
4	$65.227_{\pm 115.516}$	$273.05_{\pm 0}$
6	$58.407_{\pm 109.853}$	$253.394_{\pm 209.690}$
8	$52.181_{\pm 105.946}$	$179.713_{\pm 139.411}$
10	$31.459_{\pm 78.151}$	$223.646_{\pm 171.073}$
12	$28.205_{\pm 77.452}$	$169.91_{\pm 141.590}$
14	$14.652_{\pm 48.137}$	$153.051_{\pm 152.900}$
16	$6.216_{\pm 30.846}$	$116.684_{\pm 130.922}$
18	$3.960_{\pm 22.845}$	$50.188_{\pm 70.424}$
20	$1.742_{\pm 10.954}$	$12.710_{\pm 15.953}$

Table 9.15: Time(sec) Comparison between LSMExP_Rnd and LSMExP_Sh when connecting 23 channels.

9.4 Summary

This chapter proposes hybrid optimisation algorithms for the satellite payload switch matrix configuration optimisation problem. These techniques integrate a local search (LS), a cellular genetic algorithm (cGA) and the ILP based exact method. According to the taxonomy of hybrid methods presented in [63], these approaches are classified as high level relay hybrids.

In the first hybrid, referred to as LSM, an individual that is obtained by the LS, is inserted in the initial population of the cGA and the genetic algorithm is executed. In the second hybrid, named as LSMExB, the bound of the objective function, obtained by LSM, is set to the ILP model and the exact method is called. Finally, in the last hybrid, referred to as LSMExP, some decision variables are fixed in the ILP model, based on the best solution found by LSM. These variables are the switch positions of either a set of shortest paths (LSMExP_Sh) or a set of randomly selected paths (LSMExP_Rnd) obtained by LSM. The exact method is then optimising over the rest set of decision variables.

The experimental results demonstrated the effectiveness of each hybrid compared to the individual application of each of their composing algorithms. LSM provided better results compared to cGA in all cases. Statistical confidence was remarked for the most difficult tackled case of 23 channels to connect on the payload with 50 switches and 23 amplifiers. Besides, LSMExB improved the hit rate compared to the ILP based exact method. Especially the hybrid LSMExP provided significantly better results in terms of hit rate as well as better results in terms of fitness compared to LSM. Concerning the comparison of LSMExP_Sh to LSMExP_Rnd, which is related to the techniques according to which the decision variables are fixed, it was shown that given k paths to fix, LSMExP_Sh provides solutions of better quality but with higher computational time compared to LSMExP_Rnd. Lastly, using the LSMExP_Sh method, for the most difficult cases of connecting 23 channels to a payload with 50 switches and 23 amplifiers while minimising the length of the longest channel path, a hit rate of 100% was achieved, i.e. all tackled instances were solved.

Part IV

Conclusion and Perspectives

Chapter 10

Conclusion and Perspectives

Contents

10.1 Summary

The current communication satellites require flexibility in order to answer the increasing operational demands of a highly competitive and rapidly evolving market. Their payloads integrate large and complex switch matrices to enable the flexible routing of the channels and ensure recovery capabilities in case of failures. As a consequence though, the management of the payload, which was previously done manually by the engineers, requires now efficient optimisation techniques.

The optimisation of communication payload switch matrices configuration and reconfiguration has been studied in this thesis. A classification of the problem in three related problem cases, namely the *initial configuration*, the *reconfiguration* and the *restoration* case, has been proposed. When a set of channels has to be connected in order to allow the provision of the dedicated services, several operational objectives have to be reached. Minimising the length of the longest channel path and the number of switch changes, as well as minimising the required power to saturate the amplifiers and the number of channel interruptions, are of interest for the engineers.

Different techniques are applied in this thesis in order to efficiently solve this challenging problem. A first optimisation model has been developed in order to apply single and multi-objective exact methods. Furthermore, single-solution and population-based metaheuristics, as well as hybrid methods, have been investigated. Lastly, a generic payload optimisation framework has been proposed, that works in conjunction with the internal software tools used by payload engineers. The framework integrates the developed algorithms and provides solutions to the considered problem.

10.1.1 Optimisation with Exact Methods

A novel Integer Linear Programming (ILP) model has been developed in order to solve exactly the payload switch matrix configuration and reconfiguration problem. The model can be used for all the different problem cases, namely the initial configuration, reconfiguration and restoration problems. Several variants of the model allow the optimisation of the considered operational objectives. Furthermore, the model is easily extendable to additional operational objectives and constraints.

The experimental results on realistic payloads with 50 and 100 switches, demonstrated the validity and the efficiency of the method, for all the considered objectives. It was experimentally shown that not all instances could be solved exactly. Especially minimising the longest path length (LPL) was shown to be the most difficult problem to solve compared to the rest objectives. Problem instances that were efficiently solved when the number of switch changes was optimised, could not be solved in an exact manner when minimising LPL, even when considering the planning phase of the initial configuration case.

Furthermore, two exact mutli-objective algorithms have been compared, namely the ϵ-constraint and the adaptive ϵ-constraint method, aiming to minimise simultaneously the LPL and the number of switch changes. The adaptive ϵ-constraint method performed better compared to the ϵ-constraint method as it managed to solve more instances and it also performed faster on the commonly solved instances. Minimising the LPL individually, as one of the two extreme points, is the main factor of the limited performance of the ϵ-constraint method. This computation influences significantly the required time and this effect is more significant as the number of Pareto optimal solutions on the fronts is very low.

10.1.2 Optimisation with Metaheuristics

Metaheuristics have been applied for the first time to the considered problem targeting to tackle the problem instances that are not solvable in exact manner. The initial configuration problem case was considered with the aim to minimise the length of the longest channel path. Single-solution and population-based metaheuristics have been investigated and hybrid algorithms, that integrate the proposed methods, have been implemented.

More precisely, a local search (LS) method, based on simple hill-climbing, has been developed and a greedy method is designed and implemented in order to construct the channel paths and evaluate the quality of the solution. Furthermore, population-based metaheuristics have been compared, namely a generational genetic algorithm (genGA) and a cellular genetic algorithm (cGA). The obtained results demonstrated the efficiency of the methods and especially cGA provided better results in terms of solution quality and percentage of successfully solved instances. Nevertheless, after comparing the solutions of cGA to the optimal ones obtained by the ILP based exact method, a small degradation in the quality of the solutions is remarked.

Therefore, with the aim to further improve the quality of the solutions obtained by the metaheuristics, and to reduce the computational time required by the exact method, hybrid algorithms have been investigated that integrate the aforementioned techniques. The first hybrid, referred to as LSM, combines LS with the cGA. The second one, named as LSMExB, sets the bound of the objective based on the solution obtained by LSM and then the ILP based exact method is called. Lastly, the third hybrid, referred to as LSMExP, applies at first LSM and fixes some decision variables, based on random or shortest paths obtained by LSM. Then the ILP based exact method is called to optimise over the rest set of decision variables.

The experimental results demonstrated the effectiveness of the hybrids as they provided better results compared to the individual application of each of their composing algorithms. Especially the LSMExP hybrid performed better compared to the rest approaches. By applying this technique, all the problem instances of the larger sizes, where all the available amplifiers have to be used, were successfully solved.

10.1.3 The Experimental Framework

The proposed experimental framework that integrates the different optimisation algorithms (exact, metaheuristics, hybrid methods) allows a direct interaction with the computerised schematics (CS) tool. The CS tool is used by the engineers to describe the current payload configuration and to facilitate the payload operations that take place during the satellite lifetime. The main differences between the proposed experimental framework and the commercial tools that exist on the market for the payload reconfiguration are summarised in Table 10.1. In more details, the commercial tools either generate all feasible solutions or plan reconfigurations while minimising the number of necessary changes. Besides, they do not allow flexibility on the model and the algorithms used. The optimisation of new objectives or the application of multi-objective optimisation algorithms in order to generate the set of non-dominated solutions is not possible. Lastly, the interaction between these packages and the CS tool that is used by the payload engineers is not direct.

	Proposed Framework	Commercial Products
Flexibility	Flexible Solver/Model/Algorithms	Black Box Solver/Model/Algorithms
Multi-objective Optimisation	Yes	No
Solutions	Optimal or set of non-dominated	All feasible solutions/Limited optimisation
Interaction with CS	Direct (Input / Output: text file)	Not easy
Payload description	CS tool	New requested format
Additional features	No	Yes

Table 10.1: Differences between the proposed framework and the commercial packages.

10.2 Future Research Work

There are many open issues that can be addressed in the future. At first, a useful extension of the work can be the application of the ILP optimisation model to the front end part of the payload. In this case, the model can be easily adapted in order to consider the additional components such as multiplexers or splitters. This extension will allow the automated configuration and reconfiguration of the full payload as well as the direct calculation of the corresponding values such as losses or power.

Furthermore, the investigation of efficient hybrid multi-objective optimisation algorithms is also an interesting area for research in the considered problem. With these methods the engineers could obtain the set of non-dominated solutions considering all the operational objectives.

Besides, looking for robust solutions for the switch matrix configuration and reconfiguration problem is another important aspect. A robust solution can be defined in this context as a solution which, given any reconfiguration or restoration problem that may arise in the future, can be reconfigured with the minimum cost.

Last but not least, tackling larger payload sizes for the future communication payloads or considering fleets of satellites instead of the configuration of a unique satellite is another challenge. This could require also the investigation of parallel and distributed optimisation algorithms or co-evolutionary approaches.

Part V

Appendices

Appendix

The complete optimisation models for all the operational objectives are presented in this appendix.

Complete ILP model for minimising the longest path length

$Variables:$	
$z \in \mathbb{Z}$	
$pos_s \in \mathbb{Z}$	$\forall s \in S$
$flow_l \in \mathbb{Z}$	$\forall l \in L \cup \{l_o\}$
$b_{s,p} \in \{0;1\}$	$\forall s \in S, \forall p \in P$
$ampused_t \in \{0;1\}$	$\forall t \in T$
$flow_{l,c} \in \{0;1\}$	$\forall l \in L, \forall c \in C_{conn}$
$path_c \in \mathbb{Z}$	$\forall c \in C_{conn}$
$ObjectiveFunctions:$	
$Min\ z$	
$Constraints:$	
$flow_{l_i,i} = 1$	$\forall l_i \in CL_{conn}$
$flow_{l_i} = 0$	$\forall l_i \in \{\{CL_{in}\} - \{CL_{conn}\}\}$
$flow_{l1} + b_{s,p} * q - q \leq flow_{l2} \leq flow_{l1} - b_{s,p} * q + q$	$\forall s \in S$, $\forall p \in P$, $\forall (l_1, l_2) \in (L \cup \{l_0\})^2$, s.t. $m_{s,p,l_1,l_2} = 1$.
$\sum_{p \in P} b_{s,p} = 1$	$\forall s \in S$
$pos_s = \sum_{p \in P} p * b_{s,p}$	$\forall s \in S$
$ampused_t * q \geq flow_{tl_{in_t}}$	$\forall t \in T$
$\sum_{t \in T} ampused_t = q$	
$\sum_{c \in C_{conn}} flow_{l,c} \leq 1$	$\forall l \in L$
$flow_l = \sum_{c \in C_{conn}} c * flow_{l,c}$	$\forall l \in L$
$path_c = \sum_{l \in L} flow_{l,c}$	$\forall c \in C_{conn}$
$path_c \leq z$	$\forall c \in C_{conn}$
$flow_{l_0} = 0$	

Complete ILP model for minimising the number of switch changes

$Variables:$	
$pos_s \in \mathbb{Z}$	$\forall s \in S$
$change_s \in \{0;1\}$	$\forall s \in S$
$flow_l \in \mathbb{Z}$	$\forall l \in L \cup \{l_o\}$
$b_{s,p} \in \{0;1\}$	$\forall s \in S, \forall p \in P$
$ampused_t \in \{0;1\}$	$\forall t \in T$
$flow_{l,c} \in \{0;1\}$	$\forall l \in L, \forall c \in C_{conn}$
$ObjectiveFunctions:$	
$Min \sum_{s \in S} change_s$	
$Constraints:$	
$flow_{l_i,i} = 1$	$\forall l_i \in CL_{conn}$
$flow_{l_i} = 0$	$\forall l_i \in \{\{CL_{in}\} - \{CL_{conn}\}\}$
$flow_{l1} + b_{s,p} * q - q \leq flow_{l2} \leq flow_{l1} - b_{s,p} * q + q$	$\forall s \in S,\ \forall p \in P,\ \forall (l_1, l_2) \in (L \cup \{l_0\})^2$, s.t. $m_{s,p,l_1,l_2} = 1$.
$\sum_{p \in P} b_{s,p} = 1$	$\forall s \in S$
$pos_s = \sum_{p \in P} p * b_{s,p}$	$\forall s \in S$
$change_s * n \geq pos_s$	$\forall s \in S$
$change_s \leq pos_s$	$\forall s \in S$
$ampused_t * q \geq flow_{tl_{in_t}}$	$\forall t \in T$
$\sum_{t \in T} ampused_t = q$	
$\sum_{c \in C_{conn}} flow_{l,c} \leq 1$	$\forall l \in L$
$flow_l = \sum_{c \in C_{conn}} c * flow_{l,c}$	$\forall l \in L$
$flow_{l_0} = 0$	

Complete ILP model for minimising the number of channel interruptions

$Variables:$	
$pos_s \in \mathbb{Z}$	$\forall s \in S$
$change_s \in \{0;1\}$	$\forall s \in S$
$flow_l \in \mathbb{Z}$	$\forall l \in L \cup \{l_o\}$
$b_{s,p} \in \{0;1\}$	$\forall s \in S, \forall p \in P$
$ampused_t \in \{0;1\}$	$\forall t \in T$
$flow_{l,c} \in \{0;1\}$	$\forall l \in L, \forall c \in C_{conn}$
$interr_q \in \{0;1\}$	$\forall q \in Q_{init}$
$ObjectiveFunctions:$	
$Min \sum_{q \in Q_{init}} interr_q$	
$Constraints:$	
$flow_{l_i,i} = 1$	$\forall l_i \in CL_{conn}$
$flow_{l_i} = 0$	$\forall l_i \in \{\{CL_{in}\} - \{CL_{conn}\}\}$
$flow_{l1} + b_{s,p} * q - q \leq flow_{l2} \leq flow_{l1} - b_{s,p} * q + q$	$\forall s \in S$, $\forall p \in P$, $\forall (l_1, l_2) \in (L \cup \{l_0\})^2$, s.t. $m_{s,p,l_1,l_2} = 1$.
$\sum_{p \in P} b_{s,p} = 1$	$\forall s \in S$
$pos_s = \sum_{p \in P} p * b_{s,p}$	$\forall s \in S$
$change_s * n \geq pos_s$	$\forall s \in S$
$change_s \leq pos_s$	$\forall s \in S$
$flow_{tl_{in_t}} = flow_{tl_{out_t}}$	$\forall t \in T$
$ampused_t * q \geq flow_{tl_{in_t}}$	$\forall t \in T$
$\sum_{t \in T} ampused_t = q$	
$\sum_{c \in C_{conn}} flow_{l,c} \leq 1$	$\forall l \in L$
$flow_l = \sum_{c \in C_{conn}} c * flow_{l,c}$	$\forall l \in L$
$interr_q \geq change_s, \forall s \in S_q$	$\forall q \in Q_{init}$
$flow_{clin_c} = flow_{clout_c}$	$\forall c \in C$
$flow_{l_0} = 0$	

Complete ILP model for minimising the Input Power to Saturation (IPS)

$Parameters:$	
$att_{l,c}$	$\forall l \in L, \forall c \in C$
att_s	$\forall s \in S$
$IP_{t,c}$	$\forall t \in T, \forall c \in C$
$Variables:$	
$s \in \mathbb{R}$	
$pos_s \in \mathbb{Z}$	$\forall s \in S$
$flow_l \in \mathbb{Z}$	$\forall l \in L \cup \{l_o\}$
$b_{s,p} \in \{0;1\}$	$\forall s \in S, \forall p \in P$
$ampused_t \in \{0;1\}$	$\forall t \in T$
$twused_{t,c} \in \{0;1\}$	$\forall t \in T, \forall c \in C_{conn}$
$flow_{l,c} \in \{0;1\}$	$\forall l \in L, \forall c \in C_{conn}$
$used\_sw_{s,c} \in \{0;1\}$	$\forall s \in S, \forall c \in C_{conn}$
$ips\_path_c \in \mathbb{R}$	$\forall c \in C_{conn}$
$ObjectiveFunctions:$	
$Min \sum_{c \in C_{conn}} ips\_path_c$	(1st problem)
$Min\ s$	(2nd problem)
$Constraints:$	
$flow_{l_i,i} = 1$	$\forall l_i \in CL_{conn}$
$flow_{l_i} = 0$	$\forall l_i \in \{\{CL_{in}\} - \{CL_{conn}\}\}$
$flow_{l1} + b_{s,p} * q - q \leq flow_{l2} \leq flow_{l1} - b_{s,p} * q + q$	$\forall s \in S$, $\forall p \in P$, $\forall (l_1, l_2) \in (L \cup \{l_0\})^2$, s.t. $m_{s,p,l_1,l_2} = 1$.
$\sum_{p \in P} b_{s,p} = 1$	$\forall s \in S$
$pos_s = \sum_{p \in P} p * b_{s,p}$	$\forall s \in S$
$ampused_t * q \geq flow_{tl_{in_t}}$	$\forall t \in T$
$\sum_{t \in T} ampused_t = q$	
$4 * used\_sw_{s,c} \geq \sum_{l \in L_s} flow_{l,c}$	$\forall s \in S, \forall c \in C_{conn}$
$used\_sw_{s,c} \leq \sum_{l \in L_s} flow_{l,c}$	$\forall s \in S, \forall c \in C_{conn}$
$\sum_{c \in C_{conn}} flow_{l,c} \leq 1$	$\forall l \in L$
$flow_l = \sum_{c \in C_{conn}} c * flow_{l,c}$	$\forall l \in L$
$ips\_path_c \leq s$	$\forall c \in C_{conn}$
$twused_{t,c} \geq flow_{tlin_{t,c}}$	$\forall t \in T$
$flow_{l_0} = 0$	
$ampused_t = \sum_{c \in C_{conn}} twused_{t,c}$	$\forall t \in T$
$ips\_path_c = -\sum_{l \in L} att_{l,c} * flow_{l,c} - \sum_{s \in S} att_s * used\_sw_{s,c} + \sum_{t \in T} ips_{t,c} * twused_{t,c}$	$\forall c \in C_{conn}$

References

[1] Trecs transponder reconfiguration system. http://www.integ.com/TRECS.html. 36, 37

[2] Ibm ilog cplex. http://www.ilog.com/products/cplex/. 68, 90, 124

[3] 2014 state of the satellite industry report. *www.sia.org*, . 11, 17, 18

[4] Satellite technology and services. *www.sia.org*, . 26

[5] Karen I. Aardal, Stan P. M. Van Hoesel, Arie M. C. A. Koster, Carlo Mannino, and Antonio Sassano. Models and solution techniques for frequency assignment problems. pages 261–317, 2001. 34

[6] Emile Aarts and Jan K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997. 101, 102

[7] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. 57

[8] Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin, and M.R. Reddy. Applications of network optimization. In *Network Models, volume 7 of Handbooks in Operations Research and Management Science*, pages 1–83. North-Holland, 1995. 57

[9] E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*. Operations Research/Compuer Science Interfaces. Springer-Verlag Heidelberg, 2008. 108

[10] S. Alouf, E. Altman, J. Galtier, J. F Lalande, and C. Touati. Quasi-optimal bandwidth allocation for multi-spot MFTDMA satellites. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 560–571 vol. 1, March 2005. doi: 10.1109/INFCOM.2005.1497923. 34

[11] O. Amini, F. Giroire, S. Prennes, and F. Huc. Minimal selectors and fault tolerant networks. *Networks*, 55(4):326–340, 2010. ISSN 1097-0037. doi: 10.1002/net.20326. URL `http://dx.doi.org/10.1002/net.20326`. 35

[12] P. Angeletti, P. Gabellini, and N. Gatti. Study of a flexible payload for s-band mobile broadcasting to multiple shaped beams. In *Antennas and Propagation Society International Symposium, 2007 IEEE*, pages 3169–3172, June 2007. doi: 10.1109/APS.2007.4396209. 11, 27, 28

[13] Krzysztof Apt. *Principles of Constraint Programming.* Cambridge University Press, New York, NY, USA, 2003. ISBN 0521825830. 55

[14] Alexios Aravanis, Bhavani Shankar, Gregoire Danoy, Pantelis-Daniel Arapoglou, Panayotis Cottis, and Bjorn Ottersten. Power allocation in multibeam satellites. a hybrid-genetic algorithm approach. In *2nd ESA Workshop on Advanced Flexible Telecom Payloads*, pages 1–8. European Space Agency, 2012. 34

[15] A. A. Assad. Multicommodity network flows a survey. *Networks*, 8(1):37–91, 1978. ISSN 1097-0037. 57

[16] Cédric Balty, Jean-Didier Gayrard, and Patrick Agnieray. Communication satellites to enter a new age of flexibility. *Acta Astronautica*, 65(1-2):75 – 81, 2009. 33

[17] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Integer multicommodity flow problems. In *Integer Programming and Combinatorial Optimization*, volume 1084 of *Lecture Notes in Computer Science*, pages 58–71. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-61310-7. URL `http://dx.doi.org/10.1007/3-540-61310-2_5`. 57

[18] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.*, 48(2):318–326, March 2000. ISSN 0030-364X. doi: 10.1287/opre.48.2.318.12378. URL `http://dx.doi.org/10.1287/opre.48.2.318.12378`. 57

[19] M Basseur, E.G Talbi, A Nebro, and E Alba. Metaheuristics for multiobjective combinatorial optimization problems: Review and recent issues. *INRIA Report-ISSN*, pages 0249–6399, 2006. 83

[20] Bruno Beauquier and Eric Darrot. On arbitrary waksman networks and their vulnerability, 1999. 35

[21] Richard Ernest Bellman. *Dynamic Programming.* Dover Publications, Incorporated, 2003. ISBN 0486428095. 55

[22] J.C. Bermond, F. Havet, and C. Toth. Fault tolerant on-board networks with priorities. *Networks*, 47(1):9–25, 2006. ISSN 1097-0037. doi: 10.1002/net.20094. URL `http://dx.doi.org/10.1002/net.20094`. 35

[23] D.P. Bertsekas. *Network optimization: continuous and discrete methods.* Optimization and neural computation series. Athena Scientific, 1998. ISBN 9788865290279. 57

[24] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Inf. Sci.*, 237:82–117, July 2013. 101

[25] Lorenzo Brunetta, Michele Conforti, and Matteo Fischetti. A polyhedral approach to an integer multicommodity flow problem. *Discrete Appl. Math.*, 101. 57

[26] Lorenzo Brunetta, Michele Conforti, and Matteo Fischetti. A polyhedral approach to an integer multicommodity flow problem. *Discrete Applied Mathematics*, 101(1-3):13 – 36, 2000. ISSN 0166-218X. 57

[27] S. Cahon, N. Melab, and E.-G. Talbi. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, May 2004. ISSN 1381-1231. 112, 124

[28] J.P. Chaumon, J.C. Gil, T.W. Beech, and G. Garcia. Smartrings: advanced tool for communications satellite payload reconfiguration. In *Aerospace Conference, 2006 IEEE*, page 11 pp., 2006. 36

[29] Marie-Christine Costa, Lucas Létocart, and Frédéric Roupin. Minimal Multicut and Maximal Integer Multiflow: A Survey. *European Journal of Operational Research*, 162: 55–69, 2005. 57

[30] C. Cotta, E. G. Talbi, and E. Alba. Parallel hybrid metaheuristics. In *Parallel Metaheuristics, a New Class of Algorithms*, pages 347–370. John Wiley, 2005. 120

[31] Dilene Cruickshank. Trecs (transponder reconfiguration system). Integral Systems, Inc. 11, 37

[32] C. Dhaenens, J. Lemesre, and E.G. Talbi. K-ppm: A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1):45 – 53, 2010. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2008.12.034. URL `http://www.sciencedirect.com/science/article/pii/S0377221708010709`. 86

[33] Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992. 102

[34] B.R. Elbert. *Introduction to Satellite Communication*. Artech House space technology and applications library. Artech House, 2008. ISBN 9781596932111. URL `http://books.google.lu/books?id=_vqOgUwtWfIC`. 26, 30

[35] I. Gamvros. *Introduction to satellite communication*. PhD thesis, University of Maryland, 2006. 17

[36] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition, 2010. 101

[37] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549, may 1986. 101

[38] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675. 107

[39] S. Gulgonul, E. Koklukaya, I. Erturk, and A. Y. Tesneli. Communication satellite payload redundancy reconfiguration. In *Satellite Telecommunications (ESTEL), 2012 IEEE First AESS European Conference on*, pages 1 –4, oct. 2012. 11, 38

[40] John H. Holland. Outline for a logical theory of adaptive systems. *J. ACM*, 9(3):297–314, July 1962. 101

[41] Kaj Holmberg. Optimization models for routing in switching networks of clos type with many stages. *Advanced Modelling and Optimization*, 2008. 57

[42] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4. IEEE, nov 1995. 102

[43] Kata Kiatmanaroj, Christian Artigues, Laurent Houssin, and Frederic Messine. Hybrid discrete-continuous optimization for the frequency assignment problem in satellite communication system. 2012. 34

[44] M. Laumanns, L. Thiele, and E. Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932 – 942, 2006. ISSN 0377-2217. doi: DOI: 10.1016/j.ejor.2004.08.029. URL `http://www.sciencedirect.com/science/article/pii/S0377221704005715`. 86

[45] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966. 55

[46] J. Lemesre, C. Dhaenens, and E.G. Talbi. Parallel partitioning method (ppm): A new exact method to solve bi-objective problems. *Computers & Operations Research*, 34(8): 2450 – 2462, 2007. ISSN 0305-0548. URL `http://www.sciencedirect.com/science/article/pii/S0305054805003059`. 85, 86

[47] F. T. Lin, C. Y. Kao, and C. C. Hsu. Incorporating genetic algorithms into simulated annealing. In *Proc. of the Fourth Int. Symp. on AI*, 1991. 120

[48] Jeffrey T. Linderoth and Andrea Lodi. *MILP Software*. John Wiley & Sons, Inc., 2010. 56

[49] Gérard Maral and Michel Bousquet. *Satellite communications systems: systems, techniques and technology*. John Wiley & Sons, 2011. 11, 25, 26, 27

[50] B. Meindl and M. Templ. Analysis of commercial and free and open source solvers for linear optimization problems. *Technical report, Institut f. Statistik u. Wahrscheinlichkeitstheorie*, Technical University of Vienna, 2012. 56

[51] Michel Minoux. Multicommodity network flow models and algorithms in telecommunications. In Mauricio G. C. Resende and Panos M. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 163–184. Springer US, 2006. ISBN 978-0-387-30165-5. 57

[52] Thomas Morel, Gonzalo Garcia, Mike Palsson, and Juan Carlos Gil. Integrating and optimizing design, reconfiguration and management of payloads. In *SpaceOps*, 2010. 11, 36

[53] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988. 55, 56

[54] Una-May O'Reilly and Franz Oppacher. Hybridized crossover-based search techniques for program discovery, 1995. 120

[55] AsumanE. Ozdaglar and DimitriP. Bertsekas. Optimal solution of integer multicommodity flow problems with application in optical networks. In C.A. Floudas and Panos Pardalos, editors, *Frontiers in Global Optimization*, volume 74 of *Nonconvex Optimization and Its Applications*, pages 411–435. Springer US, 2004. ISBN 978-1-4613-7961-4. doi: 10.1007/978-1-4613-0251-3_23. URL `http://dx.doi.org/10.1007/978-1-4613-0251-3_23`. 57

[56] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3):149 – 165, 2010. URL `http://www.sciencedirect.com/science/article/pii/S1572528610000125`. 85

[57] Feng Qi, Li Guangxia, Feng Shaodong, and Gao Qian. Optimum power allocation based on traffic demand for multi-beam satellite communication systems. In *Communication Technology (ICCT), 2011 IEEE 13th International Conference on*, pages 873–876, Sept 2011. 34

[58] Gunther R. Raidl and Jakob Puchinger. Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In Christian Blum, Maria-Jose Blesa Aguilera, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, volume 114 of *Studies in Computational Intelligence*, pages 31–62. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78294-0. doi: 10.1007/978-3-540-78295-7_2. URL `http://dx.doi.org/10.1007/978-3-540-78295-7_2`. 121

[59] Andrea Raith and Matthias Ehrgott. A two-phase algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 36(6):1945 – 1954, 2009. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2008.06.008. URL `http://www.sciencedirect.com/science/article/pii/S0305054808001172`. 85

[60] L. Simone and E. Pensa. Analysis and design of redundant networks for satellite payloads. *Applied microwave and wireless*, 2001. 28, 39

[61] S. Sinha, D. Bansal, and K.J. Rangra. Rf mems compact t-type switch design for switch matrix applications in space telecommunication. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference*, pages 130–135, March 2012. 30

[62] Bita Tadayon and J.Cole Smith. Algorithms for an integer multicommodity network flow problem with node reliability considerations. *Journal of Optimization Theory and Applications*, 161(2):506–532, 2014. 57

[63] E.-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564, September 2002. ISSN 1381-1231. 12, 119, 120, 132

[64] El-Ghazali Talbi. *Metaheuristics - From Design to Implementation*. Wiley, 2009. ISBN 978-0-470-27858-1. 55, 103, 111, 123

[65] El-Ghazali Talbi. *Hybrid Metaheuristics*. Springer, 2013. 119, 120, 121

[66] Jin-Fei Tang, Liang Ma, Wei Li, Wen-Lin Zhou, and Jie Zhou. On the solution of large scale channel assignment problem in broadband satellite communication. In *Image and Signal Processing (CISP), 2010 3rd International Congress on*, volume 9, pages 4239–4243, Oct 2010. doi: 10.1109/CISP.2010.5646894. 34

[67] Nico L.J. Ulder, EmileH.L. Aarts, Hans-Jürgen Bandelt, Peter J.M. Laarhoven, and Erwin Pesch. Genetic local search algorithms for the traveling salesman problem. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 109–116. Springer Berlin Heidelberg, 1991. 120

[68] E.L Ulungu and J.Tegham. The two-phase method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20:14 – 165, 1995. 85

[69] Y. Haimes V. Chankong. Multiobjective decision making theory and methodology. *Elsevier*, 1983. URL `http://www.sciencedirect.com/science/article/pii/S0377221704005715`. 86

[70] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos. Management of an Academic HPC Cluster: The UL Experience. In *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*, Bologna, Italy, July 2014. IEEE. 68, 91, 112

[71] Daniel Wagner, Günther Raidl, Ulrich Pferschy, Petra Mutzel, and Peter Bachhiesl. A multi-commodity flow approach for the design of the last mile in real-world fiber optic networks. In Karl-Heinz Waldmann and Ulrike M. Stocker, editors, *Operations Research Proceedings 2006*, volume 2006 of *Operations Research Proceedings*, pages 197–202. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-69995-8. 57

[72] I-L. Wangt. Shortest paths and multicommodity network flows. *PhD thesis, Georgia Inst. Tech.*, 2003. 57

[73] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1 (6):80–83, 1945. ISSN 00994987. doi: 10.2307/3001968. URL `http://dx.doi.org/10.2307/3001968`. 114, 126

[74] L. A. Wolsey. *Integer programming*. Wiley-Interscience, New York, NY, USA, 1998. 56

Publications

[ACII12] Stathakis, A., Danoy, G., Bouvry, P., Morelli, G. *Satellite payload reconfiguration optimisation: An ILP model.* Lecture Notes in Computer Science, vol. 7197, ACIIDS 2012 pp. 311320. Springer (2012).

[GEC13] Stathakis, A., Danoy, G., Schleich, J., Bouvry, P., Morelli, G. *Minimising longest path length in communication satellite payloads via metaheuristics.* In: Proceedings of the fifteenth annual conference on Genetic and evolutionary computation conference, GECCO 2013.

[MET12] Stathakis, A., Danoy, G., Talbi, E.G., Bouvry, P. *A local search algorithm for telecommunication satellite payload configuration.* In: International Conference on Metaheuristics and Nature Inspired Computing (2012), META 2012.

[ROA12] A. Stathakis, G. Danoy, T. Veneziano, P. Bouvry and G. Morelli *Bi-objective Optimisation of Satellite Payload Configuration.* 13e Congrés Annuel de la Société Franaise de Recherche Oprationnelle et d' Aide a la Décision,(ROADEF), Angers, France, 2012.

[ESA12] A. Stathakis, G. Danoy, T. Veneziano, J. Schleich, P. Bouvry, and G. Morelli *Optimising satellite payload reconfiguration: An ILP approach for minimising channel interruptions.* In Proceedings of 2nd ESA Workshop on Advanced Flexible Telecom Payloads, Noordwijk, The Netherlands, 2012.

[EVO14] A. Stathakis, G. Danoy, E.-G. Talbi, P. Bouvry and G. Morelli *Hybridisation Schemes for Communication Satellite Payload Configuration Optimisation.* In Proceedings of Evolutionary Computation Conference (EvoStar), Spain, 2014.

[MET14] E. Kieffer, A. Stathakis, G. Danoy, E.-G. Talbi and P. Bouvry *Bi-objective Optimization of Satellite Payload Power Configuration.* In: International Conference on Metaheuristics and Nature Inspired Computing (2014), META 2014, (to appear).

[EngOpt14] A. Stathakis, G. Danoy, , P. Bouvry, E.-G. Talbi and G. Morelli *Optimising Communication Satellites Payload Configuration with Exact Approaches,.* Engineering Optimization Journal, (submitted).

[ALIO14] E. Kieffer, A. Stathakis, G. Danoy, P. Bouvry and G. Morell *Bi-objective Exact Optimization of Satellite Payload Power Configuration*, In: VIII ALIO/EURO Workshop on Applied Combinatorial Optimization, (submitted).

[SSCI14] E. Kieffer, A. Stathakis, G. Danoy, P. Bouvry, E.-G. Talbi and G. Morell *Multi-Objective Evolutionary Approach for the Satellite Payload Power Optimization Problem.* In: IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2014), (submitted).