TÉCNICO LISBOA

Development of Genetic Algorithm in a Branch-and-Bound Framework to Solve European Air Traffic Flow Management Problems with Conflict Cost

André Filipe Lança Serrano

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors:
Prof. Susana Margarida da Silva Vieira
Prof. Volker Gollnick

Examination Committee

Chairperson: Prof. Carlos Baptista Cardeira
Supervisor: Prof.Susana Margarida da Silva Vieira
Members of the Committee: Prof. José Rui de Matos Figueira
Prof. José Raul Carreira Azinheira

November 2018

“Blow by blow a giant is knocked down.” — David and Goliath

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my three supervisors, Prof. Susana Vieira, Prof. Volker Gollnick and Dipl.-Ing Jan Berling, for their unconditional support and cooperation throughout the thesis period. Their guidance and knowledge was absolutely determinant for the success of the developed work.

Besides my supervisors, I'm extremely grateful to Dr. Alexendar Lau, for sharing with me his deep knowledge of Network Flow Environment and for providing crucial data for the development of this work, and Mr. Ali Cinar, for all the IT support. My sincere thanks also goes to Ms. Malgorzata Safari, Ms. Birgit Sogorski, and Ms. Maice Hagemann, for their help, during the period I was in Hamburg, regarding administrative issues, and to my two colleagues, Christopher Schwanen and Sean Dericks, who never denied me their help, during the course of this work. I also would like to say a big thank you to Eng. Camilo for setting all the computers to run all the computational tests in the last phase of this journey.

Finally, I would like to express my profound gratitude to my family and friends that, each in their own way, were crucial during the course of my studies, believed in me and supported me in every way they could. This accomplishment would not have been possible without all of them. Thank you.

Resumo

O congestionamento dos aeroportos e do espaço aéreo é um problema recorrente em todo o mundo, resultando frequentemente em atrasos substanciais de voos, redireccionamentos e até cancelamentos, causando custos muito elevados para as companhias aéreas e operadores de aeronaves todos os dias. Especial atenção vai para o espaço aéreo Europeu o qual é um dos espaços aéreos mais congestionado do mundo. Preve-se que o tráfego neste espaço vai aumentar consideravelmente nos próximos anos. Nesta tese desenvolveu-se um novo Algoritmo Genético, designado de *atfmGA*, para resolver problemas binários de grande escala da Gestão do Tráfego Aéreo Europeu com probabilidades de conflicto, integrado na framework de Branch and Bound SCIP. Este novo algoritmo é baseado em conhecimento específico do problema e considera a alocação de atrasos nos voos com o objectivo económico de reduzir os custos de atrasos como também a redução de custos de conflito. O *atfmGA* é capaz de encontrar soluções admissíveis rapidamente e novos *upper bounds* para o problema considerando mais do que dois milhões de conflictos.

Palavras-chave: Gestão do Tráfego Aéreo Europeu, Alocação de Atrasos, Programação Binária Inteira, Branch and Bound, Algoritmo Genético

Abstract

Airport and airspace congestion is an inherent problem worldwide, frequently resulting in substantial flight delays, re-routings and even cancellations causing very high expenses for airlines and aircraft operators every day. Special attention goes to the European airspace which is one of the most congested airspaces in the world. Its traffic is predicted to increase considerably over the next years. In this thesis, a new Genetic Algorithm, namely *atfmGA*, for solving large-scale European Air Traffic Flow Management binary problems with conflict probabilities, integrated in the optimisation framework SCIP, was developed. For this purpose, the designed GA, based on problem-specific knowledge, considers the slot allocation with the economic objective of reducing the delay cost as well the conflict cost. The *atfmGA* is able to find quickly feasible solutions and to provide new upper bounds for the problem considering more than two millions conflict probabilities.

Keywords: Air Taffic Flow Management, Slot Allocation, Binary Integer Programming, Branch and Bound, Genetic Algorithm

Contents

List of Tables

List of Figures

Nomenclature

Air Traffic Flow Management Optimisation

c	Sector capacity
D	Total number of departure slots
d	Departure slot
F	Total number of flights
f	Flights
H	Total number of arrival airport
h	Arrival airport
i	Airport arrival capacity
J	Total number of departure airport
j	Departure airport
k	Conflict cost
o	Airport departure capacity
P_c	Conflict probability
S	Total number of sectors
s	Sector
T	Total number of sectors time-slots
t	Sector time-slot
t_h	Arrival time interval
t_j	Departure time interval
w_{fd}	Delay cost coefficient for each flight in a departure slot
x_{fd}	Decision variable for each flight in a departure slot

y Conflict surrogate variable

Z Cost of the objective function

Genetic Algorithm

γ Size of the tournament selection

μ Number of individuals in a given population

f_i Fitness of the individual

κ Crossover point

p_c Crossover probability

P_i Individual

p_m Mutation probability

Θ Population

φ Gene

d_k Allele

Glossary

1-P	1-Point crossover
2-P	2-Point crossover
ACO	Ant Colony Optimisation
ADP	ATFCM Daily Plan
AFV	Average Fitness Value
ANS	Air Navigation Services
ASM	Air Space Management
ATC	Air Traffic Control
ATFCM	Air Traffic Flow and Capacity Management. European ATFM has evolved into the new concept of ATFCM emphasising the need to balance the management of limited capacity with the demand of increasing traffic.
ATFM	Air Traffic Flow Management
ATM	Air Traffic Management
ATS	Air Traffic Service
BFit	Best Fitness
BF	Basic Feasible
BIP	Binary Integer Programming
BR	Bias Resetting mutation
CASA	Computer-Assisted-Slot-Allocation
CDM	Collaborative Decision Making
CFMU	Central Flow Management Unit
CIP	Constraint Integer Programming
CM	Creep Mutation
CODA	EUROCONTROL Central Office for Delay Analysis
CPF	Corner-Point Feasible
CTOT	Calculated Take-Off Time
DCB	Demand-Capacity-Balancing

DLR	German Aerospace Centre
EATMN	European Air Traffic Management Network
EA	Evolutionary Algorithm
ECAC	European Civil Aviation Conference
EC	Evolutionary Computation
EOBT	Estimated Off-Block Time
ESRA08	European Statistical Reference Area
ETO	Estimated Time Over
FMP	Flow Management Position
FPFS	First-Planed-First-Come
FR	Feasibility Rate
GA	Genetic Algorithm
GDP	Ground Delay Program
IATA	International Air Transport Association
IFR	Instrument Flight Rules
ILT	Institute of Air Transportation Systems
LB	Lower Bound
LP	Linear Programming
MBFit	Median of the Best Fitness
MEX	Matlab Executable
MINLP	Mixed Integer Non Linear Programming
MIP	Mixed Integer Programming
NFE	Network Flow Environment
NMOC	Network Manager Operation Centre
NM	Network Manager
NSGA-II	Non-dominated Sorting Genetic Algorithm
PDF	Probability Distribution Function
PR	Performance Rate
RR	Random Resetting mutation
SA	Simulated Annealing
SCIP	Solving Constraint Integer Programming
SESAR	Single European Sky ATFM Research
SES	Single European Sky
SGA	Simple Genetic Algorithm
STD	Standard Deviation
TUHH	Technical University of Hamburg
UB	Upper Bound
UX	Uniform Crossover

Chapter 1

Introduction

1.1 Motivation for Solving the European Air Traffic Flow Management

Airport and airspace congestion is an inherent problem worldwide, frequently resulting in substantial flight delays, re-routings and even cancellations causing very high expenses for airlines and aircraft operators every day. Special attention goes to the European airspace which is one of the most congested airspaces in the world. Its traffic is predicted to increase considerably over the next years [1]. The increasing of the travel air traffic flow demand in Europe is illustrated in figure 1.1. It shows an increasing trend of IFR movements or just simply, number of flights in the European Statistical Reference Area (ESRA08). Until the year of 2040, four main scenarios with different growth rates are represented. For all different economical scenarios using different input assumptions[1], a growth in the number of flights is observed.

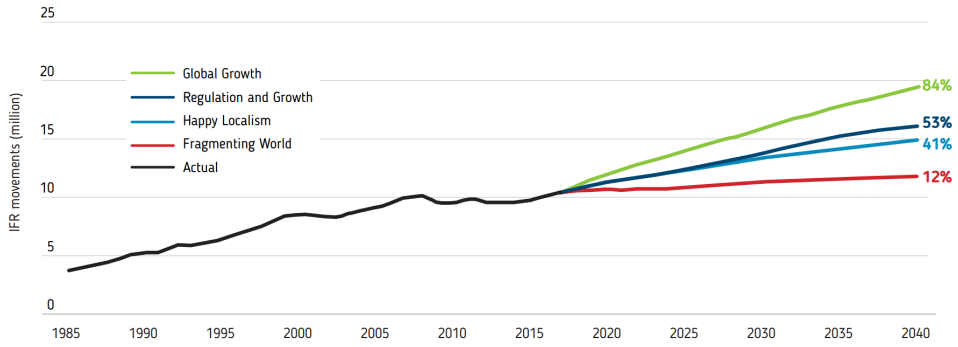


Figure 1.1: Increasing Air Traffic Flow in Europe for different scenarios in the long-run, source [2]

Consequently, a great number of conflicts between aircraft emerge in the en-route area. To avoid such conflicts and minimise delays, an aircraft position forecast should be used and an Air Traffic Flow Management (ATFM) must be provided improving the efficiency and safety of air transportation.

[1] Such as economic growth, fuel prices, load factors etc.

One way to measure the efficiency of air transportation is to analyse flights' delays. In the figure 1.2 it is notable an increasing of departure delay per flight over the past 7 years in Europe and consequently a decreasing of its efficiency.

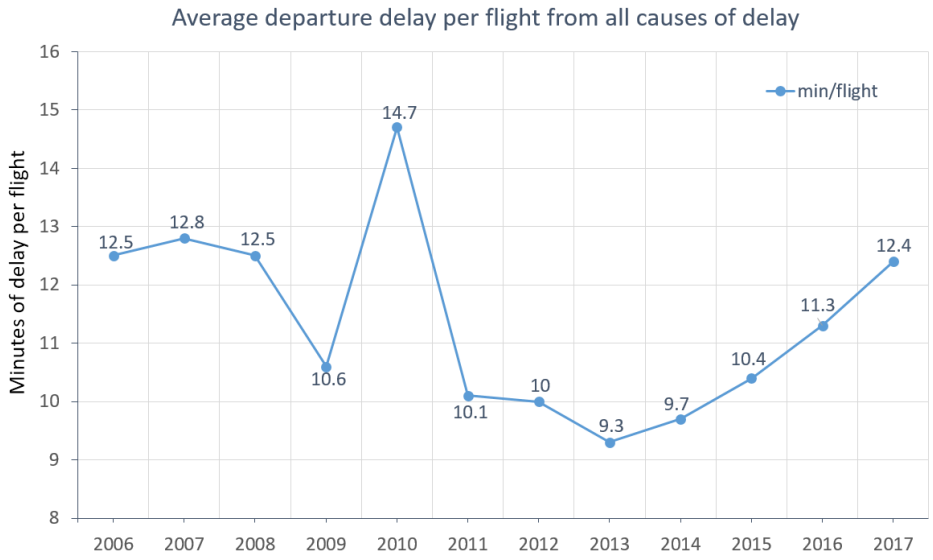


Figure 1.2: Average departure delay per flight in Europe between 2006 and 2017, data' source: CODA Digest reports from 2006 to 2017

Sources of error in trajectories spatial and temporal prediction, span a wide range of factors such as meteorological conditions, human behaviour, take-off time uncertainties, etc. inducing inaccurate forecasts. The focus is on deviations of departure time which are the largest temporal uncertainties on the ground. These deviations can be categorised in causes of departure delay facilitating their identification. The figure 1.3 depicts the causes of departure delay differentiated by EUROCONTROL.

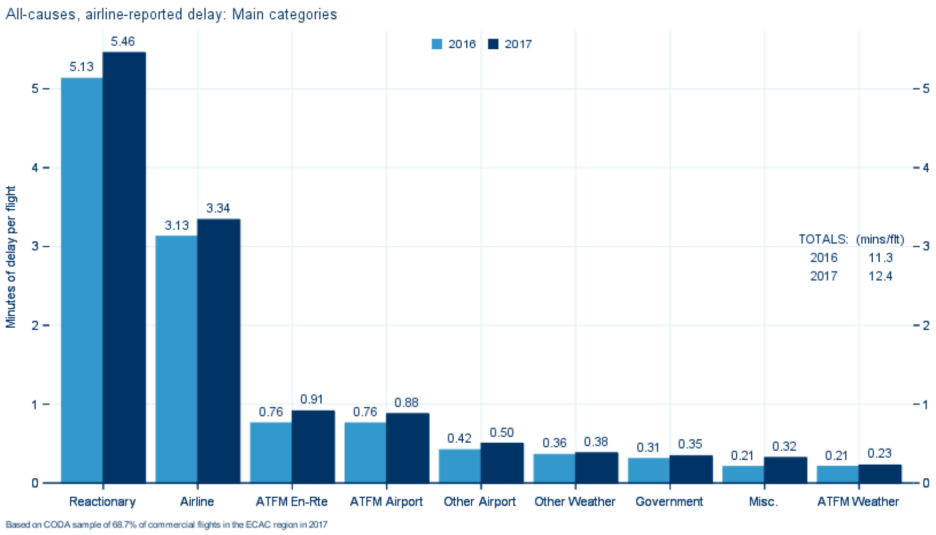


Figure 1.3: All causes of delay in Europe in 2016 and 2017, source [3]

The target is on causes related with ATFM delays (ATFM En-route plus ATFM Airport) which accounts for about 14% of all causes of departure delay. Distributions of departure times' deviation in Europe, show that nearly half of all fights deviate at least five minutes [4]. The figure 1.4 depicts the mentioned

distribution for the years 2016 and 2017. It's notable that the fraction of flights delayed raised from 2016 to 2017.

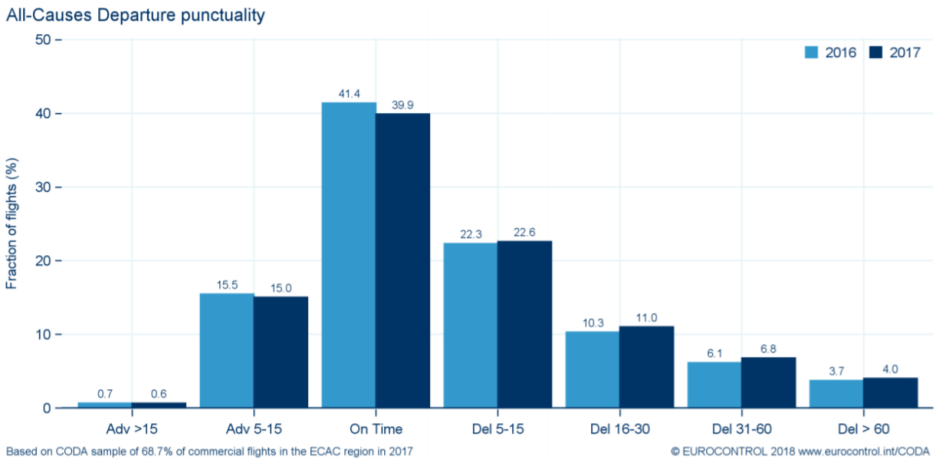


Figure 1.4: Distributions of departure times' deviation in Europe in 2016 and 2017

Safety is one of the key performance of the air transportation system [5]. To guarantee a safe flight for each aircraft, air traffic controllers (ATCs) ensure a separation minima standard for all aircraft. When two aircraft are in a situation that may lead to a violation of separation minima, a conflict occurs [6]. On a daily basis, these conflicts are solved by ATCs responsible for specific sectors. Hence, the emerging en-route conflicts increase controller workload and thus tighten sector capacity in order to controllers be able to solve all conflicts. To improve safety, the Single European Sky ATFM Research (SESAR) program seeks to reduce the number of conflicts [7]. Because conflicts are dependent of time, improved separation of aircraft in time reduces conflict potential. For strategic deconfliction, the Network Manager (NM) could re-allocate departure time-slots while satisfying both sector and airport capacity constraints.

Taking into account the inherent uncertainties of the departure times, it is possible to compute the probability associated to potential conflicts, namely, conflict probabilities [8]. Conflict probabilities are computed for all the possible departure time combinations. Moreover, many flights have conflicts with multiple other flights. One of the aims of this work is a strategically prevention of probabilistic conflicts in any future point in space and time, by allocating alternative departure time-slots to planned flights, namely strategic deconfliction, which can reduce conflict probabilities but also should not increase flight take-off's delays significantly. As a basis for the deconfliction, actual datasets of planned trajectories along with the respective conflict probabilities, sector bounds and airport features are aggregated to model the European ATFM. Consequently, this work develops an ATFM approach that combines strategic probabilistic deconfliction with adherence to system capacities. To achieve both objectives, alternative departure timeslots are allocated. Reduce conflict probabilities by departure time-slot allocation increases delays. Due to the interconnectedness of probabilistic conflicts between flights and their departure timeslots, a large scale ATFM assignment problem is to be solved in order to find the best departure time-slots combination.

1.2 Optimisation of the European Air Traffic Flow Management

For these type of problems, a planning (programming) of activities is to be done to obtain an optimal solution, thus there is the necessity for allocating resources (departure time-slots) to activities (flights) by choosing the levels of those activities. For this, there is the need to use decision variables associated to their respective costs, see [9]. Since the nature of the decision is *yes-or-no*, a Binary Integer Programming (BIP) problem arises with binary decision variables which means that they are restricted to integer values, namely 0 or 1. Binary problems with n variables have an exponential large number of possible solutions of 2^n. For that reason, it is not possible to check all solutions. Moreover, this BIP problem is NP-Hard (Non deterministic Polynomial acceptable) which means, it cannot be solved in polynomial runtime but only in exponential runtime. Thus, computer programs cannot handle them efficiently taking a long time to run hence there are no such available algorithms that can find optimal solutions for this problem.

ATFM with deconfliction is to be computed by using BIP allocating several departure time-slots. This optimisation problem has quadratic conflict cost associated to each conflicted flight pair and linear capacity constraints associated to sector and aerodrome capacity which bound the feasible region. In order to solve the problem, there is the need to linearise the quadratic conflict-cost-term [5] because quadratic functions are non-linear and for that reason they are hard to solve. Unfortunately the non-linear programming problems can be *nonconvex* which means that it is not possible to ensure for this problem that a *local minimum* is also a *global minimum* [9]. Thus, the quadratic costs are replaced by linear conflict cost constraints and surrogate variables.

The European ATFM is modelled by a tactical ATFM model named Network Flow Environment (NFE). It was implemented at the Institute for Air Transport Systems (ILT) of the German Aerospace Centre (DLR) of the Technical University of Hamburg-Harburg (TUHH). NFE is based on real traffic data and involves a realistic approximation of European sector and airport capacity. The ability to allocate departure time-slots to flights throughout the European ATFM network is represented in the model by binary decision variables by satisfying the objective of optimal delay minimization. These variables are constrained by sector and airport capacity. The sectors and airports capacity-afflicted by the respective planned route of the flights, are determined by the mapping of the real flight plans to the network elements capacities.

The ATFM problem in Europe, for one day, typically deals with around 30.000 flights associated to 10 different departure time-slots resulting, as product between flight movements and departure time-slots, in 300.000 decision variables and millions of possible conflicts. The great number of conflicts is due to the fact that for each pair of flights there are several points from their trajectories which have conflict probability [5]. This illustrates the interconnectedness mentioned earlier which requires a great computational effort for finding solutions. There is an extremely large number of possible solutions, around two to the power of 30.000 plus millions of possible conflicts.

To solve such a large BIP problem an algorithmic approach must be used. One option, is to use Branch-and-Bound methodology. Firstly, this methodology uses LP relaxations to ignore the integer

constraints that require variables to have integer values. Secondly, solves the problem as a linear programming problem by using the Simplex Method and hopefully get an integer solution [10]. Since it has fewer constraints, its optimal solution provides a lower bound (LB). Because these hopes are almost always unfulfilled, a backup strategy is necessary. The easiest strategy is to use the simple heuristic *rounding* which rounds each resulting solution value to its nearest integer value providing an upper bound (UB). However, there is no guarantee that a relaxed solution is necessarily feasible after it is rounded. The goal is to do several iterations to try to improve the bounds.

To enhance the Branch-and-Bound process, there is the possibility to incorporate general solution methods, such as metaheuristics which are capable of dealing with problems that are too large and complicated to be solved by exact algorithms. Among the available metaheuristics, Genetic Algorithm (GA) is the one proposed to be used. This metaheuristic is inspired by the process of natural selection. Genetic algorithms are commonly used to generate high-quality solutions to optimisation and search problems by relying on bio-inspired operators such as mutation, crossover and selection [9]. The advantages of using this algorithm are the ability to escape from a *local minimum* and, using the Branch-and-Bound information, converge to good solutions performing a robust and fast search of a feasible region [9]. The disadvantage when using GA alone is that there is no way to find the global lower bound without trying every branches (all possible solutions) which requires much more computational effort. Fortunately, when combining GA in the framework, the lower bounds are computed using SCIP's Simplex Method, namely SoPlex, thus making possible to compare with the UB.

1.3 Related Work - Genetic Algorithms in Air Traffic Flow Management

In 1990s, the Operational Research community has started to study ATFM problems using different optimisation approaches on many variants of this problem. Only in 1994, Daniel et al. first shown that GA can be used in ATFM problems.

Due to the complexity of this problem, the majority of the approaches are not able solve the whole problem. Hence, it is usually solved partially for simplified instances. For example, the works [12–14] only solve the problem for the French airspace. The problem's mathematical model also varies from study to study. Some models don't contemplate sectors' capacity constraints [15] and others don't contemplate airports' capacity constraints [12].

Some ATFM mathematical models using GA present different objective functions. They can be classified as mono-objective [13, 16, 17], bi-objective [15, 16, 18] and multi-objective [19]. Different GA schemes have been used in previous works depending on ATFM problem' structure. For problems with more than one objective function, the Non-dominated Sorting Genetic Algorithm (NSGA-II) is used [18, 20]. In [15] a competitive co-evolutionary GA is applied.

For problems with just one objective function, such as in [13], the authors use two different GA approaches namely, Simple Genetic Algorithm (SGA) and new-recombinators Genetic Algorithm. In the

work [16], the authors also use two different GA approaches. The first is an hybrid GA which has a flavour of Simulated Annealing (SA) in the crossover operator in order to speed up the convergence and the second is close to the former but with an enhanced problem-specific crossover operator. This operator receives additional information regarding the most congested sectors and then focus on recombinations in order to smooth the peaks of sectors' capacity congestion by assigning to flights new departure slots. The latter mentioned work was the first attempted to solve the whole ATFM problem considering sector and airport capacity. In this work, an ATFM network with 15 sectors and 24 airports was used to evaluate the performance of implemented GAs. 3000 flight plans spread over 6 hours were generated randomly. Two different combinatorial models are tested, one that allocates new routes and departure slots, and another one that only allocates departure slots.

Other previous work [17] solved the whole ATFM problem with sector and airport capacity constraints using the BIP model presented by Bertsimas and Patterson. This model uses binary integer variables and its objective function is minimize the total delays cost of the overall system. The optimisation was performed by two metaheuristics, a parallel GA and a Simulated Annealing (SA), and Integer Programming (IP). To evaluate the performance of these three approaches, a network model of the National Air Space (NAS) of United States of America (USA) and a data set with 974 sectors, 905 airports and thousands of flights of one full day. Results have shown that IP is generally more efficient and provides better results than the two metaheuristics. Surprisingly, it was used a GA scheme without the mutation operator which was pointed out to explain the fall short of expectations on GA convergence.

1.4 Objectives and Contribution

1.4.1 Objectives

The goal of this thesis is to find good feasible solutions in an acceptable time-frame reducing delays and conflict probabilities costs of the European Air Traffic Flow Management problem making use of an enhanced Branch-and-Bound framework, incorporating Genetic Algorithms into the solving process.

1.4.2 Contributions

One of the contributions of this thesis is the development and incorporation of a stochastic metaheuristic scheme, namely Genetic Algorithm, to enhance the SCIP solving process of the European ATFM problem with conflict cost. A new GA, namely *atfmGA*, was developed from another framework code structure and fully re-engineered to match the problem's features.

The other contribution, attains the study and comparison of the performances of SCIP and GA.

To make use of GA in the Branch-and-Bound framework it is necessary to do the following steps. Firstly, select the most suitable GA framework and include it in SCIP. Secondly, adapt it to the problem' structure to leverage GA's capabilities. Therefore, it is necessary to customise a SCIP's modules to allow a good *interplay* between GA and Branch-and-Bound's framework.

1.5 Thesis Outline

The manuscript is organised as follows:

Chapter 2 - Air Traffic Management in Europe.

In this chapter 2 an historical background and an overview of the ATFM organisation in Europe is presented. A brief description of the algorithm used to aid the planning of the ATFCM is given. Furthermore, a short analysis of delays in Europe is carried out.

Chapter 3 - Modelling Work-flow and Data Set

The chapter 3 provides a detailed description of the ATFM network model and associated data. Mathematical formulations of the ATFM without and with strategic conflicts are exposed.

Chapter 4 - Solving Binary Integer Programming by Branch-and-Bound

An introduction of mathematical optimisation theory can be found in chapter 4. Then follows a brief description of SCIP with special focus on the primal heuristics incorporation in its solving process. The used interface is described along with data handling from Matlab to SCIP and vice-versa.

Chapter 5 - Genetic Algorithm

The implementation of the GA is described in this core chapter 5. Related works using GA in ATFM problems are referred. Then, it follows an introduction to GA terminology and concepts. A research on the available GA frameworks was done before the actual GA development. The remaining part of this chapter is devoted to GA implementation and its integration in SCIP.

Chapter 6 - Results

In chapter 6 the obtained results are presented. The performances of optimisation tools described in chapter 4 and 5 are then compared.

Chapter 7 - Conclusions

Last but not least, concluding remarks and future work are presented in chapter 7.

Chapter 2

Air Traffic Flow Management in Europe

Before this chapter gets under way, it's important to clarify some basic terminology. Based on the figure 2.1, ATFM is a component of one of the Air Navigation Services[1] (ANS) called Air Traffic Management (ATM). The service of ATM comprises all the services related to air navigation, which are:

- Air Space Management (ASM);
- Air Traffic Service(s) (ATS);
- Air Traffic Flow Management (ATFM).

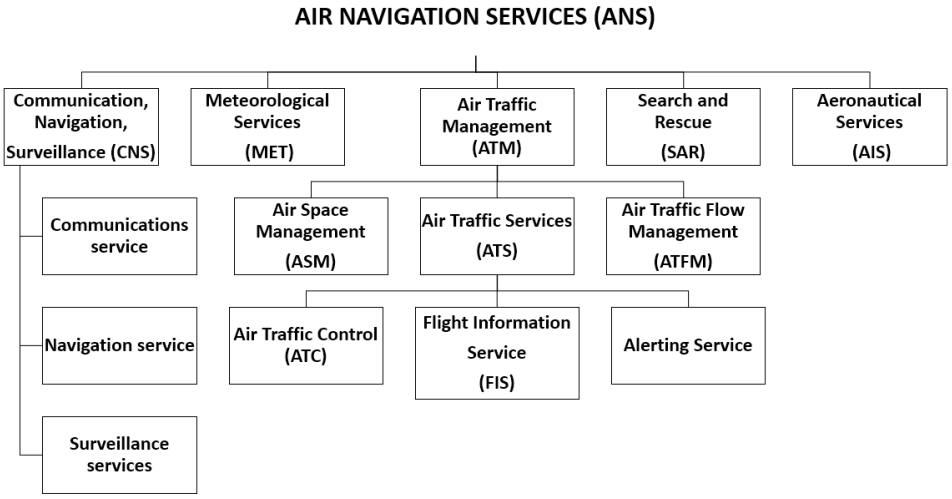


Figure 2.1: Diagram of Air Navigation Services(ANS), adapted from source [22]

The function of ASM is to plan and publish the management of airspace, branched into air routes, civil and military control routes and areas reserve for airports, while at the same time ensuring the safety

[1] Also known as Air Navigation Services Providers

and fluidity of traffic. Together, ASM and ATFM support the use of the available airspace effectively, including airport capacity, by minimising waiting times.

The focus of this thesis is on ATFM, more specifically, the European ATFM network. This service is established with the objective of contributing to a safe, orderly and expedited flow of air traffic by ensuring that the existing network elements (ATC sector[2] and airport) capacity are utilised to the maximum extend possible and that the traffic volume is compatible with capacities declared by the appropriate ATS authority with timely, accurate information for planning and execution of an economical air transport, as close as possible to foreseen flight intention and without discrimination [23].

2.1 Historical Background

In Europe, during the late 1960s, appeared a necessity to organise and co-ordinate the air traffic regarding demand, time and geographical distribution. Until mid 1990s, aircraft position forecast, ATFM and ASM used to be the responsibility of the Air Traffic Control (ATC) of each ATC sector in an attempt to regulate traffic flows and to match demand with capacity. However, it was soon realised that flow control on a regional basis gave rise to problems. As each state tried to protect its airspace without understanding the impact on neighbours and the rest of the network, the airspace gradually grew more restricted, giving rise to yet more delay. Due to increasing amount of traffic and the untangling of flights between sectors combined with a restricted airspace, the ATCs were no longer able to cope, at economically acceptable levels, with peaks in demand resulting in an unmanageable situation.

Consequently, the ATM in Europe had reached its worst performance ever during 1980s. In 1986, 12% of flights were delayed for more than 15 minutes on average and in 1989, 25% of all flights were delayed for more than 15 minutes [24]. Politicians and their constituents were haunted by television images of people stranded at airports while they waited for their flights.

It began to appear that the only solution was to carry out flow management centrally so as to make the best possible use of all the available airspace capacity. The ministers of transport of the European Civil Aviation Conference (ECAC) member states met in Frankfurt in October 1988 to plan the commencement of a Europe-wide centralised ATFM service [25]. The management of this project was entrusted to EUROCONTROL[3], the European Organization for the Safety of Air Navigation, which set up the Central Flow Management Unit (CFMU). From 1989, the CFMU gradually took over responsibility for providing efficient and safe flow management and by 1996, it had taken over responsibility for the range of ATFM services in Europe previously handled by the regional flow management units.

The forecasts of the increasing air traffic levels and the additional costs every year was a big concern of the EU. In order for the European airspace accommodate the increasing air traffic flows, whilst cutting costs and improving its performance, the Single European Sky (SES) was established, [26]. This way, the airspace became organized into functional blocks according to traffic flows instead of national borders, following common rules and procedures at European level. The European Commission estab-

[2]Or just simply sector is the smallest area of airspace under specific control.

[3]EUROCONTROL is an intergovernmental organisation with 41 members that delivers the ATM performance required for the following years

lished the centralised function Network Manager (NM), as part of the regulation (EU) Nº 677/2011, and nominated EUROCONTROL for this function in July 2011, see [27].

2.2 Organization of Air Traffic Flow Management

At the heart of EUROCONTROL lies the Network Manager Operations Centre (NMOC). NMOC is responsible for the following operational functions:

- **Airspace Data Management**: creates, verifies and updates the airspace data system (Environment);
- **Flight Plan Processing**: receives flight plans filed by aircraft operators, validating and if necessary correcting those flight plans (automatically or manually) then redistributing the flight plans to the aircraft operators and the overflown airspace control centres;
- **Air Traffic Flow and Capacity Management**[4] **(ATFCM)**: Aims to optimise ATC capacity in order to meet the air traffic demand as effectively as possible, as well as guaranteeing safety and efficiency in every phase of its activities.

2.2.1 Air Traffic Flow Management Phases

As the figure 2.2 depicts, the ATFCM activities are assigned to four time-related execution phase:

- **Strategic phase**: takes place at least 6 months prior the day of operations[5] and ends approximately 7 days before. This phase includes flight plan processing, coordination activities and preplanning through a Collaborative Decision Making (CDM) process to predict what capacity ANSPs will need to provide in each of their air traffic control centres avoiding imbalances between capacity and demand for events taking place in the future (large-scale military exercises, major sports events, etc.) which causes traffic loads and consequently congesting network elements. Bottlenecks of traffic flows within the European Air Traffic Management Network (EATMN) are identified [28] and then large scale ATFM plans on the orientation of traffic flows are designed;
- **Pre-tactical phase**: starts 6 days prior the day of operation, studies the demand for the D-Day, compares it with the predicted available capacity on that day, and makes any necessary adjustments to the plan that was developed during the Strategic phase by implementing a range of ATFCM measures (*e.g.* re-routing scenarios to flights, manage sector configuration, etc) optimising the efficiency and balance demand and capacity through an effective organization of resources. The work methodology is based on a CDM process between NM and their stakeholders [29]. The output is the ATFCM Daily Plan (ADP) published in preparation for the next phase [22];

[4] The terms ATFM and ATFCM are roughly identical in most contexts such as in this one.

[5] Also known as D-day, is the last day of the ATFM routine. In the D-day, the traffic is managed through slot allocation and re-routing.

- **Tactical phase**: is operated on the day of operation. In this phase, traffic rates and capacities are regularly updated to the ADP. Especially in the case of conflicting network impact, capacity profiles dynamically oscillate according to traffic complexity patterns. The necessity to adjust the original plan ADP may result from disturbances such as staffing problems, significant meteorological phenomena, crises, special events, etc. The provision of accurate information is of vital importance in this phase, since it permits short-term forecasts, including the impact of any event and maximises the existing capacity without jeopardising safety [28, 29];

- **Ad-hoc phase or Post Operational Analysis**: is the final step in the ATFCM planning and management process, in which activities operated collaboratively by controllers and pilots, are enforced to balance traffic flows within affected airspaces and congested airports. During this phase, an analytical process is carried out that measures, investigates and reports on operational processes and activities throughout all domains and external units relevant to an ATFCM service for feedback purposes [28, 29].

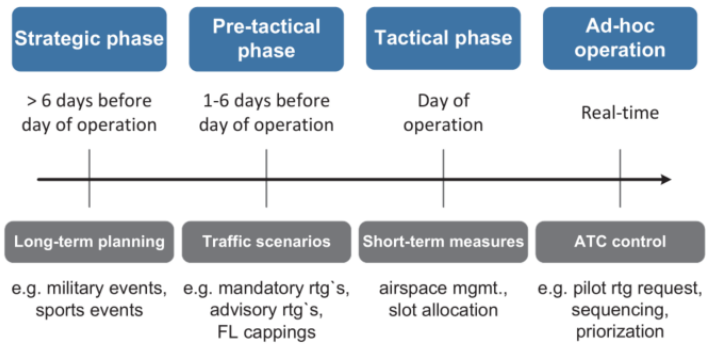


Figure 2.2: Time line and functions of the ATFCM phases, source [28]

The focus of this thesis is on the tactical phase. The central development goal of NMOC in this phase is to apply a Demand-Capacity-Balancing (DCB) process, which includes dynamic airspace management and pre-flight departure slot allocation [28]. The NMOC manages the flows of 43 states which participate in ECAC to ensure that user demand (*i.e.* from the airlines) does not overload the capacities offered by the airspace infrastructure (*i.e.* En-route and Terminal Manoeuvring Areas).

2.2.2 Network Manager Operations Centre and its Stakeholders

The NMOC is composed by independent systems each one with different functions assigned, communicating internally between them and externally with EUROCONTROL' stakeholders. The figure 2.3 illustrates the NMOC system' structure.

The Air Operators (AOs) represent all responsible for a flight under IFR (except the military ones). They shall file their Flight Plans (FPL) through the Integrated Initial Flight Plan Processing System (IFPS) and file Planed Flight Data (PFD) to the Pre-Tactical system. The ATS Reporting Offices (AROs) shall in some cases play the role of interface between NMOC and AOs.

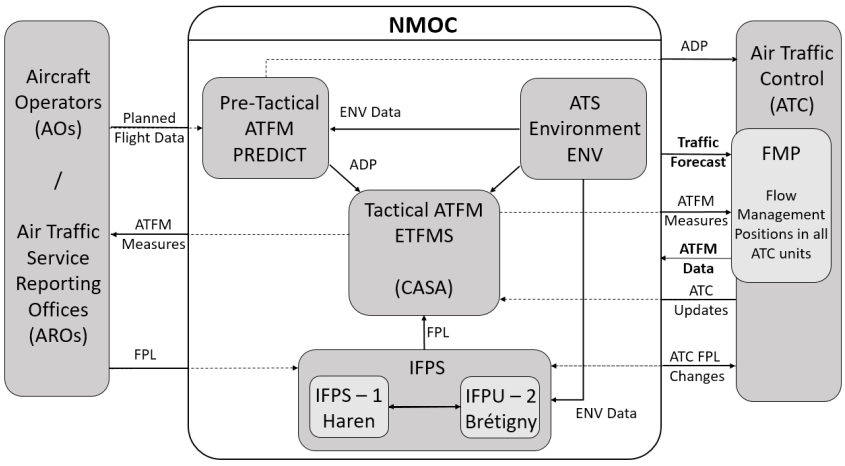


Figure 2.3: Network Manager Operations Centre system' structure overview, adapted from source [23]

The Air Traffic Control (ATC) is one of the existing Air Traffic Services (ATS), see figure 2.1, responsible for the safety of all aircraft on the ground or in the air. The Flow Management Position (FMP) is one of the EUROCONTROL stakeholders which ensures the necessary interface between local ATFCM partners (*i.e.* ATCs, AOs and airports) and NMOC on matters concerning the provision of the air traffic flow and capacity management service. The FMP shall provide the NMOC with relevant ATFM data to enable it to carry out its responsibilities in all phases of ATFCM operations, see [30]. FMPs and AOs are informed of the ATFM measures taken by NMOC.

The IFPS shall collect the FPL and check against the airspace structure in the system [31]. To adhere to sector and airport capacities, any inconsistencies need to be resolved before the FPL can be accepted. A copy of every accepted FPL with or without changes is then sent to the NMOC's Tactical system and to all the ATC units in Europe affected by that particular flight. In order to guarantee this service to all European ATC units, the NMOC has two IFPS Units working in parallel, one in Haren, Belgium and another in Brétigny, France acting as contingency sites for each other.

The ATS Environment (ENV) is the common airspace data repository feeding operational systems and enabling aeronautical data services. All EUROCONTROL member states provide the NMOC with their airspace data, which is then used to create a 3D model of the airspace structure. The ENV data contains both static data such as sector boundaries and air routes (*e.g.* the maximum capacity for each airport and sector), as well as dynamic data such as default ATC capacities (*e.g.* the number of runways available, availability of air traffic controllers, etc) and air-route availability based on military airspace usage [32].

The Pre-Tactical system is responsible for the pre-tactical phase refining the details of the original forecast over time. It uses a tool as support named PREDICT which provides a fairly accurate overview of the traffic loading on the day of operations, sector configurations and a Tactical-like environment in which ATFCM measures can be simulated off-line enabling to study the operations overall effect [30]. This tool receives as input PFDs and ENV data and produces as output the ATFCM Daly Plan (ADP) published one day before the day of operations. The ADP is a set of ATFCM measures that will be in force in European airspace on the following day.

The Enhanced Tactical Flow Management System (ETFMS) is located in the very heart of the interaction between the ETFMS system, AOs and ATC. It is responsible for the tactical phase which is overruled by short-term slot allocation, in this context ATFM slot[6]. ATFM slots comprise tactically Calculated Take-Off Times (CTOTs) for which a flight departs on time within 5 minutes before and 10 minutes after the CTOT [29]. The slot allocation is implemented for departures from ATFCM area or from ATFCM Adjacent area when entering ATFCM area. A map in figure 2.4 illustrates the areas' boundaries.

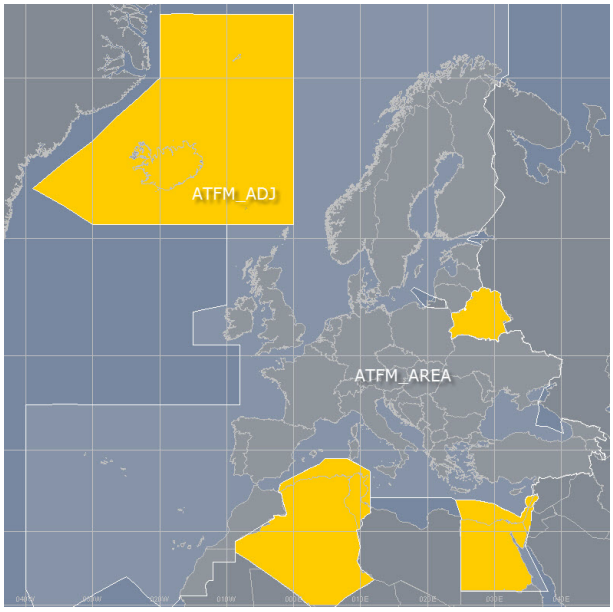


Figure 2.4: The map of ATFCM Areas, source [33]

The Estimated Off-Block Time (EOBT) reveals the true time the AO expects the aircraft to be ready to depart. Whether the aircraft is permitted to depart at this time depends on the effect of any flow restrictions placed on the airports and the airspace through which the flight's route is planned [22]. One of the usual ATFM measures is to apply a regulation[7], *i.e.* to limit the maximum rate of aircraft entering either a regulated volume of airspace or airport over a specific period of time due to flow restrictions. A flight affected by an ATFM regulation is assigned to a new take-off time, *i.e.* an ATFM slot, in an automated process using Computer-Assisted-Slot-Allocation (CASA) system under the principle First-Planed-First-Served (FPFS)[8]. Consequently, regulated flights which are delayed on the ground for longer time, suffer an ATFM delay. Thereby, CASA enables ATC to inflict ATFM delays to aircraft in order to adhere the airborne capacity.

The ETFMS has two main functions.

- The calculation of the traffic demand in every airspace sector within the NM area of operations, see figure 2.5, using the FPL information received from the IFPS.

- Using the CASA heuristic algorithm, the complex process of calculation, allocation and distribution

[6] Also known as a 'departure slot', 'CTOT', or just 'slot'. ATFM slots shall not be mistaken with airport departure and arrival slots. Airport slots constitute planned time frames of 15 minutes length, negotiated within a slot conference on the basis of an airports capacity benchmark value [28].
[7] Also known as ATFM regulation
[8] Adapted from the rule First-Come-First-Served (FCFS)

of slot lists. These lists have ATFM slots applied to the set of flights being restricted according to DCB requirements. This process is shortly described below.

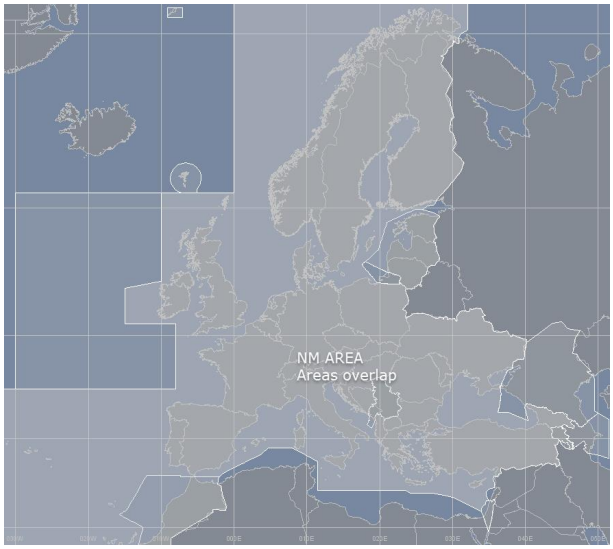


Figure 2.5: The map of NM Area of Operations, source [34]

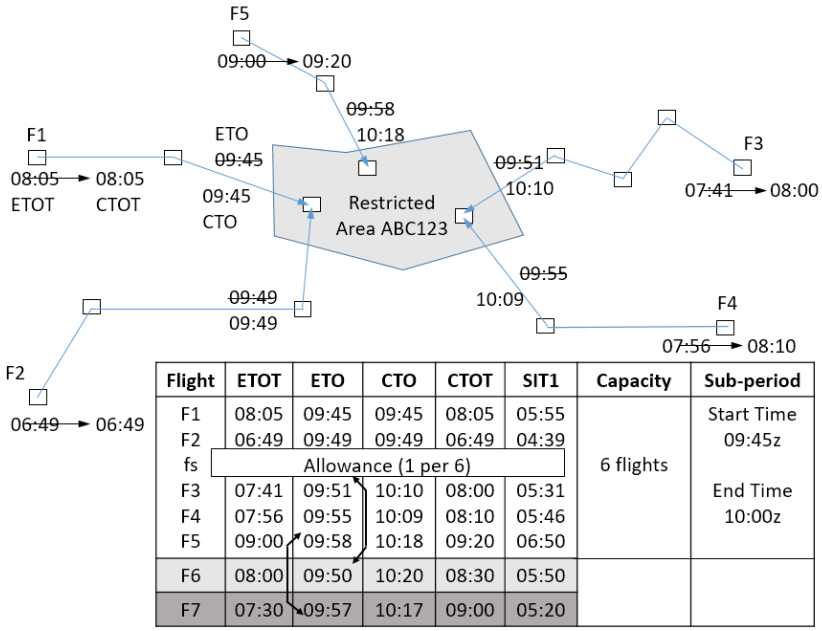
2.2.3 CASA - Computer Assisted Slot Allocation

An initial Slot Allocation List (SAL), is built and managed for each resource (*i.e.*, an en-route sector or an airport). This list consists in an empty slot allocation list with a number of slots function on the rate of acceptance (*e.g.* flight/hour) assigned over a sub-period [29]. In the example illustrated in the figure 2.6, a SAL was prepared for the fictitious Restricted Area ABC123. With a flow rate of 24 flights per 1 hour, the list has 6 empty slots for the upcoming flights separated from one another by 2,5 minutes over the sub-period 09:45z to 10:00z.

Then, CASA is fed with flight data (*i.e.* FPLs from IFPS), assigns an Estimated Take-Off Time (ETOT) for each flight based on the EBOT plus the taxi-time[9] at the departure aerodrome. This way, it is possible to calculate and assign for each flight the Estimated Time Over (ETO) for the point of entry at each entering sector in the planned route [22]. Subsequently, CASA pre-allocates the time-slots to flights in the respective slot lists in accordance with the principle FPFS[10] as close to their ETO as possible, *i.e.* it sequences them in the order they would have arrived at the airspace in the absence of any restriction. In the given example, in the figure 2.6, 5 initial flights (F1, F2, F3, F4 and F5) were assign to 5 different time-slots accordingly to their ETO. When CASA receives new flight data, the time-slot is pre-allocated as close to the requested ETO the restricted location as it is available:

- if that time-slot is free, it is assigned to the new flight which thus suffers no delay. In the same example a new flight F6 with an ETO in between the ETOs of the flights F2 and F3, is assigned to the third slot because there was one free slot;

[9] Time from the start of motion of an aircraft, under engine power, until the cessation of motion at the completion of a flight, minus flight time.
[10] This principle refers to the flight's ETO based on the FPL.



Flight	ETOT	ETO	CTO	CTOT	SIT1	Capacity	Sub-period
F1	08:05	09:45	09:45	08:05	05:55		Start Time
F2	06:49	09:49	09:49	06:49	04:39		09:45z
fs	Allowance (1 per 6)					6 flights	
F3	07:41	09:51	10:10	08:00	05:31		End Time
F4	07:56	09:55	10:09	08:10	05:46		10:00z
F5	09:00	09:58	10:18	09:20	06:50		
F6	08:00	09:50	10:20	08:30	05:50		
F7	07:30	09:57	10:17	09:00	05:20		

Figure 2.6: Scenario for a fictitious Restrited Area at a given period of time, adapted from source [23]

- if that time-slot is already pre-allocated to a flight which is planned to enter the restricted location after the new flight, then the latter takes the slot. Another new flight, F7, is assigned to the last slot of the list because the latter enters the congested location first than fight F5 which is out of this SAL.

Airports and sectors have declared capacities, and although flow managers will try to match capacity with demand, when the number of aircraft in a slot list exceeds capacity, a regulation is activated. The flight F5 was not allocated in this time-slot thus activating a regulation, so it will get the next available one, *i.e.* an ATFM slot allocated is allocated to F5, and hence suffers a ATFM delay equal to the difference in slot times. From this, the respective CTOT is determined. This process often leads to a chain reaction of slot changes as new flights enter the time-slot list. To guarantee an optimal tactical reaction on ATFM restrictions, at the earliest two hours before the EOBT of each flight, named Slot Issue Time 1 (SIT1), see figure 2.7, the ATFM slot needs to be allocated to the flight and a message containing the respective time-slot allocation information is sent to the AOs and ATC [29]. To do so, at the earliest 3 hours before EOBT of each flight subject to ATFCM, its FPL must be sent to the IFPS to be filed [29]. If the flight doesn't comply with these timings, a range of mechanisms will act to compensate for the respective flaw.

The figure 2.7 depicts an exemplary slot allocation process of the flight F5 from the previous example planned to enter a regulated sector volume at an ETO at 09:58z. The flight is now declared as ATFM-restricted with a CTO at 10:28z, suffering an ATFM delay of 30 minutes. EOBT and ETOT times are going to be shifted accordingly. To comply with this slot, the flight must take off within a -5 to +10 minute slot window, that is, between 10:23z and 10:38z. Since a single flight might enter more than one regulated entity, the delay caused by the most penalising one (the one which causes the highest delay) along its planned 4D trajectory, is forced in all the others, rather than the ETO [29]. The actual delay also

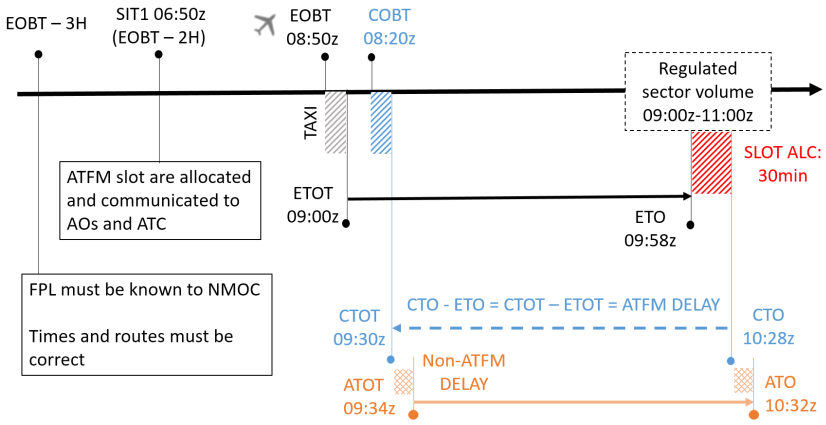


Figure 2.7: CASA-based ATFCM slot allocation scheme, adapted from source [28]

reflects a proportion of unpredictable delay originated by airport operations, airline scheduling, weather, etc.

2.3 Delays in Europe

The use of ATFM regulations by delaying aircraft on the ground to prevent more traffic than the coordinated capacity is a daily reality for AOs. This ATFM procedure is as known as Ground Delay Program (GDP) [35]. Its principle of tactically corresponding demand with capacity by inflicting delay on the ground is the essence of ATFCM flow control. It is cheaper, both in fuel and environmental cost-wise, safer and easier to delay a departure than to allow the aircraft to become airborne and then impose it a delay either en-route or on the approach to the destination airport, by speed control, by holding or by re-clearing the aircraft to non-optimal flight levels [36]. Thus, ground holding is more cost effective and more environmental friendly than airborne delay management.

If the capacity of the airspace through which a flight is planned is reduced, or the number of flights planned through that airspace in a given time period exceeds the standard capacity of that portion of airspace, it is then necessary to delay aircraft on departure to loosen the demand.

But what is delay? Accordingly to EUROCONTROL in [37] ”delay is the time lapse which occurs when a planned event does not happen at the planned time”. Depending on personal preferences and history, measuring delay can be done in different perspectives such as whether the lateness is measured on departure or arrival. As mentioned before, the focus of this work is on deviations of departure times on the ground, *i.e.* the delay is measured on departure.

Delays result for a large number of reasons such for a lack of resource, mechanical issues, planned processes, etc. In order to provide means of comparison and to enable ATM partners to them communicate issues associated with delayed departures, a standard set of delay definitions was introduced by the International Air Transport Association (IATA). The figure 2.8 depicts the causes of departure delay analysed by EUROCONTROL.

It is notable in the above figure that, by far, the most meaningful cause of delay is ”reactionary[11]”. This

[11] Also known as secondary or rotation delay

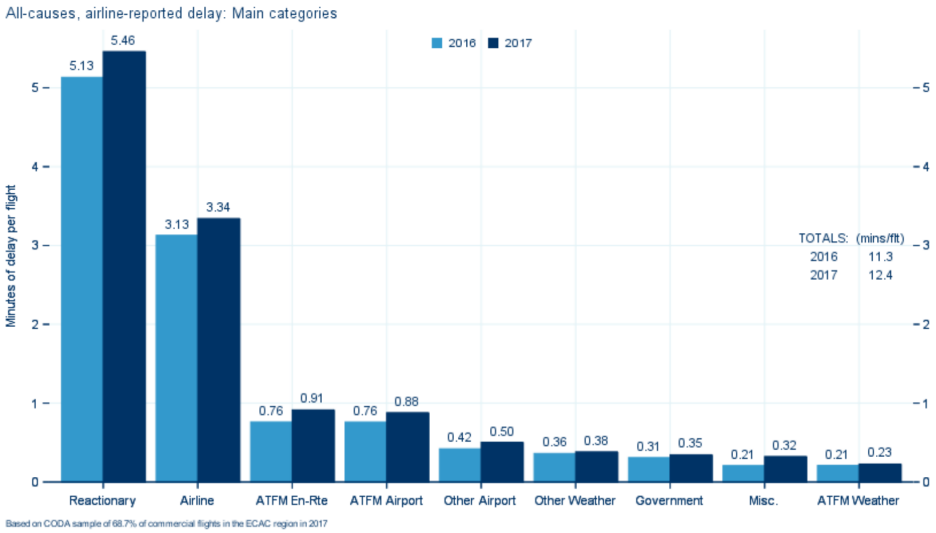


Figure 2.8: All causes of delay in Europe with the respective average delay per flight in 2016 and 2017, source [3]

cause refers to all delays which may be directly attributed to an initial delay [22]. In 2017, this accounted for about 44% of the total amount of delays or around 5,5 minutes per flight. The other causes can be categorised in the same group as known as ”primary delays” which is defined as delay that affects the initiation of the flight [37]. These delays are unaffected by any earlier or accumulated delay unlike reactionary delays. The figure 2.9 illustrates the primary causes of delay analysed by EUROCONTROL.

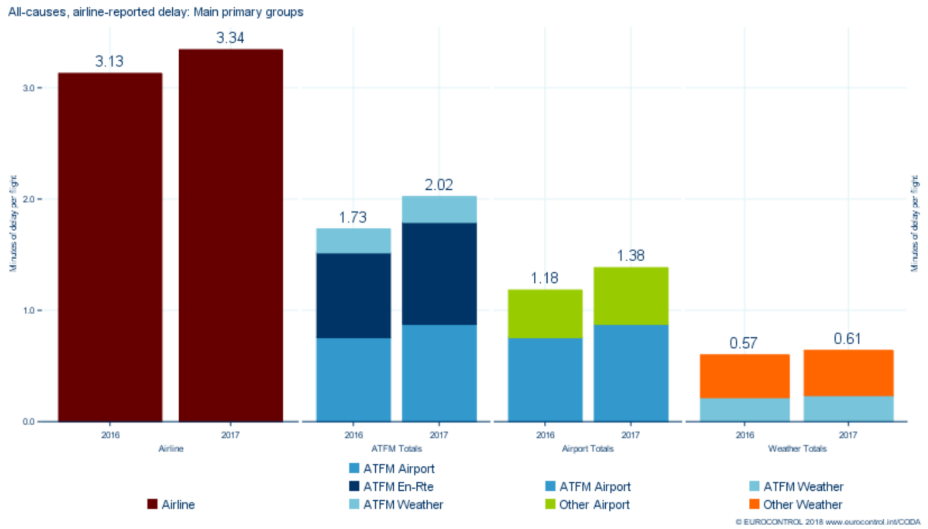


Figure 2.9: Primary delay causes in Europe with the respective average delay per flight in 2016 and 2017, source [3]

2.3.1 ATFM Capacity

The main contributor to aviation delay in Europe is a lack of ATC capacity. To evaluate ATC capacity, causes related with flow management delays must analysed such as:

- ATFM en-route: delay caused by regulations based on traffic volume to protect en-route ATC

sectors from overload;

- ATFM airport: delay caused by regulations based on traffic volume on destination or departure airports to protect them from overload.

By analysing the data in figures 2.8 and 2.9, in total these two causes account for about 14% of all causes of departure delay and 26% of all primary causes in 2017. Other causes related with en-route and airport staffing and disruptions of ATC capacity accounts for 16%, [38]. This means general capacity issues in Europe accounted for 42% of delays during the year. The influence of weather on en-route and airport operations collectively total 32.7% of delays [38], which means, the bulk of airspace interruptions are clearly derived from controllable factors.

2.3.2 Air Traffic Influence on ATFM Delay

The figure 2.10 depicts the evolution of ATFM en-route and airport delays over the past years.

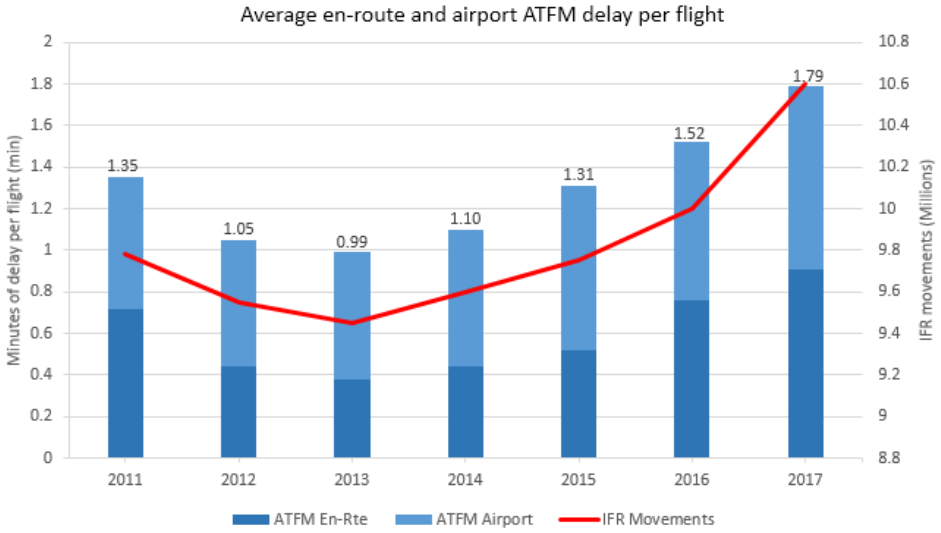


Figure 2.10: Average ATFM en-route and airport delays per flight versus IFR movements, data' source: Performance Review Reports from 2011 to 2017

It is noteworthy that the increasing of ATFM en-route and airport delays goes along with growth of IFR movements, *i.e.* more traffic and demand leads to more delays. This phenomena is clearly notable if we look to the evolution of the en-route ATFM delay throughout the months' years in figure 2.11.

A seasonal pattern peaking of number of flights in summer is clearly visible. Particularly the months of July, August and September of 2017 had the highest monthly traffic ever recorded, each amounting more than one million flights. The busiest period for the European airline industry is undoubtedly in the summer, specifically for the month of Jul-2017, there were on average 33 721 flights per day [38], the highest of the year. However, during the highest demand period, en-route ATFM delay also reached its yearly peak at 1.98 minutes per movement.

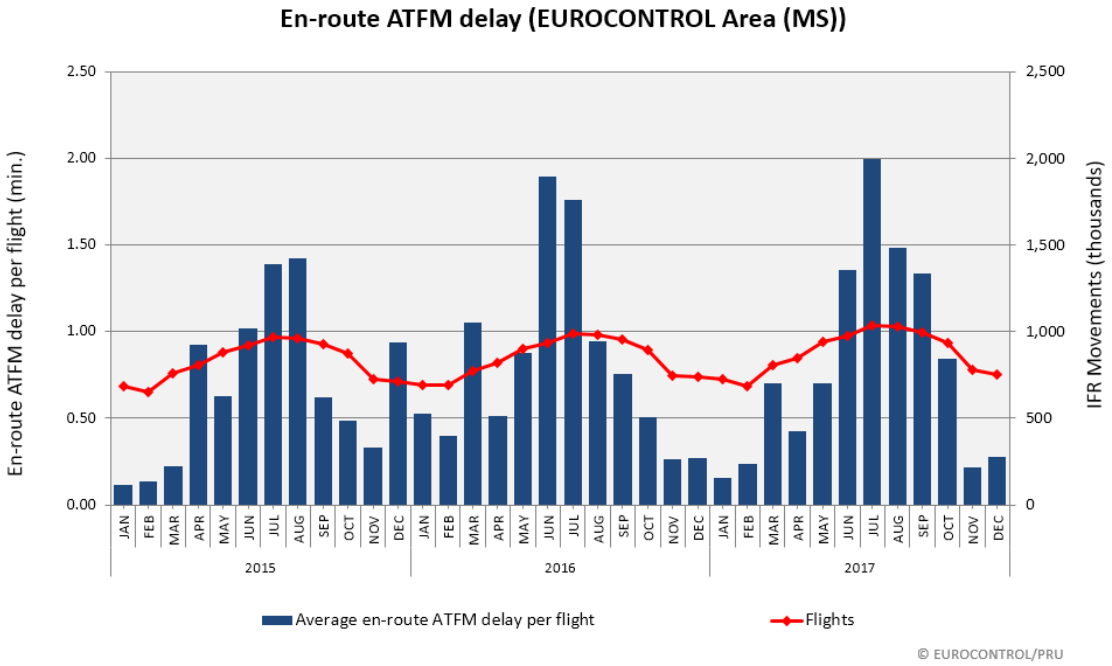


Figure 2.11: Monthly evolution of ATFM en-route delay per flight from 2015 to 2017 versus IFR movements, data' source: Performance Review Reports from 2015 to 2017

2.3.3 Cost of ATFM Delay

Many implications emerge when introducing delays to air traffic causing a number of detrimental impacts such as on customers perception, cost, efficiency and environment. As this work deals with air traffic flow control, costs incurred by ATFM are particular relevant. The network average cost of ATFM delay, per minute was updated in 2014 and is set to the value of EUR 100 [39], *i.e.* the cost of an extra minute delay for an average flight is estimated for all aircraft operating in Europe at EUR 100. Using the data from figure 2.10 it is possible to compute the total costs associated with ATFM en-route and airport delays presented in the table 2.1.

Year	ATFM Delays (M min.)			Estimated cost of ATFM delays (EUR 2014 Prices) in millions of euros		
	En-Route	Airport	Total	En-route	Airport	Total
2014	4.2	6.3	10.6	422.4	633.6	1056.0
2015	5.1	7.7	12.8	507.0	770.3	1277.3
2016	7.6	7.6	15.2	760 .0	760.0	1520.0
2017	9.7	9.3	19.0	964.6	932.8	1897.4

Table 2.1: Estimated cost of ATFM departure delays

The estimated costs due to ATFM en-route and airport delay amounted to EUR 932.8 millions in the last year. The delays and costs increased drastically in the past 5 years. This is largely due to the increasing number of flights since then.

Chapter 3

Modelling Work-flow and Data Set

In this chapter, the network model of EATFM is presented in the section 3.1. In the following sections, the mathematical optimisation model is depicted. Firstly, in section 3.2, the ATFM problem with only delay costs is presented. Secondly, in section 3.4 the ATFM problem with strategic deconfliction is also presented.

3.1 The Network Flow Environment

The Network Flow Environment (NFE) is a tactical ATFM model software suite of the whole European ATFM network for pre-flight, re-routing and pre-slot allocation [40]. NFE allocates ATFM departure slots, *i.e.* CTOTs, in a similar way as it is actually applied by CASA's algorithm in NMOC for the tactical Demand-Capacity Balancing (DCB) in Europe. When flights are planned to enter highly congested network elements along their individual estimated trajectory and there is no free slots, then departure slots derive in pre-departure ATFM delays. To handle the European ATFM network, the modelling approach comprises of network elements, *i.e.* airports and ATC sectors.

NFE consists in two functional blocks illustrated as horizontal process flows of sub-modules in figure 3.1:

- Data Preparation and Processing;
- Demand-Capacity-Balancing (DCB and slot allocation).

3.1.1 Data Preparation and Processing

In this section, data is extracted from several different sources. The Eurocontrol Demand Data Repository (DDR2) and the European AIS database (EAD) serve as data sources of environmental data types (airspaces, navigational data, ATS route data, capacity and regulation data). The traffic data used in this work contains estimated flight plans provided by the DDR2.

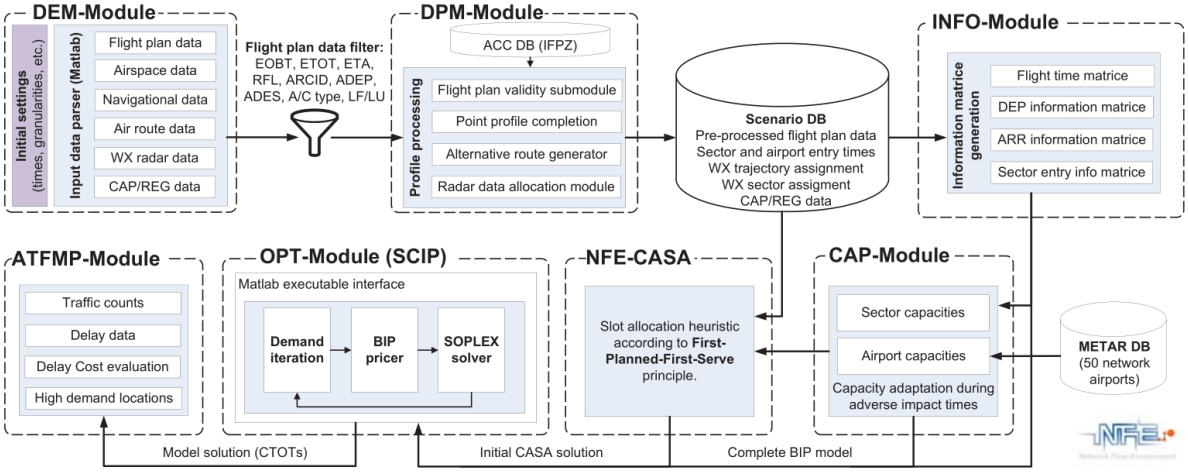


Figure 3.1: Scheme of the Network Flow Environment model, source [40]

Data Extraction

The Data Extraction Module (DEM) stands at the beginning of the NFE data preparation and processing section. This module is responsible for the extraction of all data types according to initial settings, like day of choice and geographical area of interest. The Initial Flight Plan Processing Zone (IFPZ) delimited by the most outer white line in the figure 2.4, represents the default setting, containing a wider area including Europe and some of its neighbouring states.

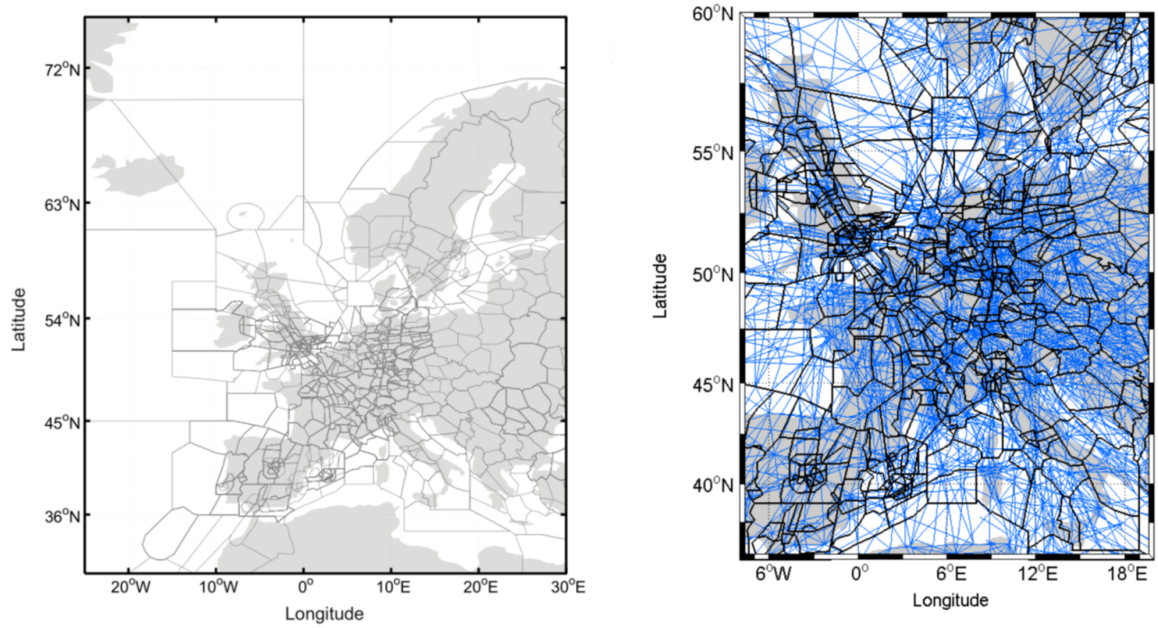
Most of the airspace model provided is depicted in figure 3.2.a. It contains 617 sector volumes, representing approximately 30 000 traffic flows, *i.e.* flights, of the EATMN. The traffic flows within the airspace model can be visualised in figure 3.2.b. The model contains realistic sectors with capacities and closed boundaries defined.

The used flight plan data contains elapsed flight time, air route indicator, navigational point information including origin and destination airports, flight level indicator and ETOs. NFE applies estimated flight plan profiles according to tactical ATFCM operations to generate demand profiles for each network element.

Data Processing

The Data Processing Module (DPM) is called after the flight plan data filter for the generation of specific traffic scenarios without demand uncertainties. This module is capable of generating realistic demand time-constrained scenarios, for which a subset of flight plans are extracted, *e.g.* with an EOBT within a given period of time. Thereafter, the DPM completes every point profile by adding:

- additional points for every minute;
- geographical coordinates to each point according to AIRAC-conform navigation data;
- the sector profile;
- adverse impact location (*e.g.* weather convective events) information.



(a) Airspace static sector model, source [40]

(b) Airspace model with traffic flows, source [28]

Figure 3.2: NFE airspace model

To determine sector entry times and conflicts accurately, point profiles are generated with a time-step of 1 minute, although the specified ATFM slot time-step of the model is 15 minutes.

NFE also applies tactical pre-flight re-routing according to initiated routing scenarios during tactical NM. Flights planned to enter highly congested sectors or being assigned to high ATFM departure delay are proposed to be re-routed. Nevertheless, the present study is not integrating pre-flight re-routing as tactical ATFM measure.

3.1.2 Demand-Capacity-Balancing

Capacity Module

The Capacity Module (CAP) generates nominal capacities for airports and ATC sectors. These capacities are valid for time periods according to DCB time-step and thus, determining the length of a computed ATFM slot. This value is set to 15 minutes by default. ATC sector capacities are provided by the DDR2 database. NFE generates quickly nominal sector capacity vectors according to sector design guidelines adjusting its capacity values during adverse impacts. The airports runways capacities are generated by means of a process simulation model which receives individual airport data concerning the number of runways, aircraft mix and airport weather. This model covers high demand network of airports for which service capacities are generated according to individual service values and respective flow-delay-functionalities [40]. For this work, historic capacities profiles are used in order to provide a most realistic network input.

Mathematical Optimisation Model

The allocation of departure slots according to an overall system delay minimisation is performed by a binary-integer optimisation module, which is able to handle large scale ATFM problems in an acceptable amount of computation time. The slot allocation problem is implemented in MATLAB and applies pre-compiled libraries of the SCIP (Solving Constraint Integer Programs) 3.2.1 software framework together with the SoPlex linear programming solver. SCIP is interfaced within NFE's computational work-flow via an adaptation of the OPTimization Interface (OPTI) [28].

3.2 Air Traffic Flow Management Mathematical Formulation Problem

Considering a set of tasks that need to be processed, and a set of agents that are able to process these tasks. A limited amount of a single resource is available to each of the agents, and each task requires a particular amount of this resource when it is processed by a particular agent. The resource consumption may depend on which agent processes a given task. The Generalised Assignment Problem is then the problem of assigning each task (time-slot) to exactly one agent (flight), so that the total cost of processing all tasks is minimised and no agent exceeds its resource capacity.

The model is based on aircraft's spatial-temporal movements, *i.e.* flights, with different starting time intervals. Thereby, flights are assigned to discrete starting time intervals whereby departure time-slots are modelled as decision variables. When a flight departures at specific departure time-slot the corresponding value of the variable is one, otherwise is zero. These variables are related to a singular flight, its respective entry point in a sector and its respective departure and arrival airports whereby a local discretisation lies due to the clear separation and delimitation of the network elements. Due to the fact that the capacity is dependent of time intervals, the time is discretised, *i.e.* an ATFM slot is specified according to the selected temporal granularity.

Each decision variable is associated with a fixed route that defines which network elements are capacity-afflicted by a flight. The routes specify how much time a flight takes to the respective sectors and airports. Therefore, jointly with the starting time intervals, a true assignment of decision-making variables on network elements' timeslots is accomplished.

Each airport has individually limited take-off and landing capacity. The control of air traffic flow is performed with time-slots allocation, hence holding the aircraft on the ground as long as it is needed in order to avoid expected capacity bottlenecks due to flight's route. The amount of delay inflicted on flight's take-off requires clear costs for each decision variable and thus, ultimately, for the total cost. The latter is composed of the costs of all decision variables. In the following, the individual components of the linear model are presented.

Decision Variable

Following the work of Bertsimas and Stock Patterson (1998) [21], the same binary decision variable formulation is used. Each variable $x_{f,d}$ for each flight f in a departure slot d of 15 minutes takes the form,

$$x_{f,d} = \begin{cases} 1, & \text{if flight } f \text{ obtains ATFM slot } d \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Delay Cost Coefficient

The costs of start-slots d (also known as ground-holding costs) $w_{f,d}$ varies for each flight f and it is measured in dollars $.

Flight Times CTOT and CTO

After the allocation of an ATFM slot causing a delay d in a flight f, the latter had been given a new take-off time in the departure airport j, *i.e.* a new CTOT. It can be given by the sum of the Estimated Take Off-Time (ETOT) with ATFM delay inflicted:

$$CTOT_j = ETOT + d \tag{3.2}$$

In the en-route phase, the entry point of the flight f in the sector s, *i.e.* the new Calculated Time Over CTO_s is going to be shifted accordingly and it can be obtained by the sum of the new take-off time $CTOT_j$ with the ETO_s in the sector s:

$$CTOT_s(f,d) = CTOT_j + ETO_s = ETOT + d + ETO_s \tag{3.3}$$

Constrains

The problem is characterised by two types of constraints. On one hand, each flight takes-off once and on the other hand sector and airport capacities must be fulfilled for every time interval.

Start Condition

The departure constraint ensures that every flight f is assigned to exactly one departure d then we have the following set partitioning constraint:

$$\sum_{d \in D(f)} x_{f,d} = 1, \quad \forall f \tag{3.4}$$

Capacity Constraints

Sectors and airports have individual limited capacity. For a sector, this means that no more aircraft are allowed to enter at any instant of the fully congested time interval, until the next time-slot as its capacity allows. For the airports in turn, it means that in a time interval no more aircraft may take off than the take-off's capacity allows and may not land more planes than the landing's capacity allows. Airport capacity is build in NFE via individual runway capacity. The partial capacities of other airport elements, such as the terminal, are neglected.

Sector Capacity

The coefficient a assigns the flight f with delay d to the sector s in the particular time-slot t which corresponds to the Calculated Time Over (CTO) of the respective flight.

$$a_{(s,t),(f,d)} = \begin{cases} 1, & \text{if } CTO_s(f,d) = t \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

The sum of all sector s incoming flight's entries assigned to timeslot t is constrained by the respective capacity c resulting in the following *knapsack* constraint, *i.e.* the sum of decision variables (traffic demand) must be equal or smaller than the sector's capacity.

$$\sum_{f \in F} \sum_{d \in D(f)} a_{(s,t)(f,d)} \cdot x_{f,d} \leq c_s, \quad \forall s, t \tag{3.6}$$

Airport Capacity

For the arrival airport capacity:

The coefficient r assigns the flight f arrival with delay d to the departure airport h in the particular departure time-slot t_h which corresponds to the CTO_h of the respective flight.

$$r_{(h,t_h),(f,d)} = \begin{cases} 1, & \text{if } CTO_h(f,d) = t_h \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

The sum of all airport h incoming flight's departures assigned to time-slot t_h is constrained by the respective departure capacity o resulting in the following *knapsack* constraint, *i.e.* the sum of decision variables (traffic demand) must be equal or smaller than the destination airport's departure capacity.

$$\sum_{f \in F} \sum_{d \in D(f)} r_{(h,t_h)(f,d)} \cdot x_{f,d} \leq i_h, \quad \forall j, t \tag{3.8}$$

For the departure airport capacity:

The coefficient p assigns the flight f arrival with delay d to the destination airport j in the particular arrival time-slot t_j which corresponds to the CTO_j of the respective flight.

$$p_{(j,t_j),(f,d)} = \begin{cases} 1, & \text{if } CTO_j(f,d) = t_j \\ 0, & \text{otherwise.} \end{cases} \tag{3.9}$$

The sum of all airport j incoming flight's arrivals assigned to time-slot t_j is constrained by the respective arrival capacity o resulting in the following *knapsack* constraint, *i.e.* the sum of decision variables (traffic demand) must be equal or smaller than the destination airport's arrival capacity.

$$\sum_{f \in F} \sum_{d \in D(f)} p_{(j,t_j)(f,d)} \cdot x_{f,d} \leq o_j, \quad \forall j, t \tag{3.10}$$

The delay must not be negative, *i.e.* premature departure times are not assigned and thus,

$$d \geq 0 \quad \forall d \in D(f) \tag{3.11}$$

3.3 Binary Integer Programming Model

The ATFM problem is formulated as a Binary Problem with binary decision variables, a cost function and linear constraints allowing the application of methods of linear optimisation. Therefore, it was modelled within NFE a binary integer programming (BIP) optimisation which determines how to allocate departure slots according to an overall system delay minimisation.

The objective function of the total delay is to be minimised. Since each flight f is assigned exactly to one ATFM slot, *i.e.* departure slot d, the delay cost $\omega_{f,d}$ is afflicted for each flight only once. The linear objective function $Z(x)$ is the sum of the delay costs. It is declared by the equation 3.12.

$$Z(x) = min\left(\sum_{f} \sum_{d} \omega_{f,d} \cdot x_{f,d} \right) \tag{3.12}$$

In vector formulation,

$$Z(\mathbf{x}) = \min_{x} \mathbf{w}^\top \mathbf{x} \tag{3.13}$$

The linear problem with binary decision variables (see equation 3.1) contains the cost function (see equation 3.13), the starting condition (see equation 3.4) and the capacity restriction (see equations 3.6, 3.8 and 3.10). The cost function is allowed under the constraints.

$$\begin{aligned} \min_{x} \quad & \mathbf{w}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{G}\mathbf{x} = \mathbf{e} \\ & \mathbf{A}\mathbf{x} \leq \mathbf{c} \\ & \mathbf{R}\mathbf{x} \leq \mathbf{i} \\ & \mathbf{P}\mathbf{x} \leq \mathbf{o} \\ \text{and} \quad & \mathbf{x} \in [0,1] \end{aligned} \tag{3.14}$$

3.4 Air Traffic Flow Management with Strategic Deconfliction

The goal of ATC is to manage air traffic on a short-term horizon by monitoring the traffic and keeping aircraft separated within a separation minima. The latter is standardized as the distance of five nautical miles lateral and thousand feet vertical between all aircraft in the upper space [7] as the figure 3.3 depicts. A conflict between two airborne aircraft arises when these aircraft converge in space and time so that they may endanger the minimum separation like figure 3.4 illustrates. To resolve conflicts, *i.e.*, to prevent predicted losses of separation between aircraft, the ATCs provide pilots with instructions to perform airborne manoeuvres. The latter involve changes in the speed, heading or flight level consequently inducing costs due to fuel consumptions.

ATCs have workload capacity, *i.e.* the physical and mental work that controllers must undertake to safely conduct air traffic under their jurisdiction through en-route airspace [41], which impacts sectors capacity [22]. Depending on the number of converging aircraft within an ATC sector at a specific time and the current traffic situation, the mentioned instructions may rule actual ATC workload. The increasing number of flights in the en-route area increases the changes of losses of separation minima consequently the number of conflicts leading to ATCs work overloading and thus tighten the airspace capacity. ATC often applies a safety margin controller workload which could be adapted when conflict forecast is improved [22]. However the delays' costs induced by airborne manoeuvres are significantly lower than those induced by ground holding [1] strategic deconfliction by allocating ATFM delays can potentially reduce ATCs workload and ultimately increasing the sector's capacity.

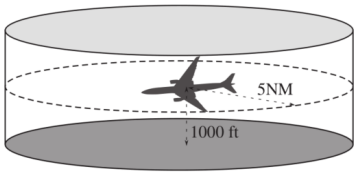
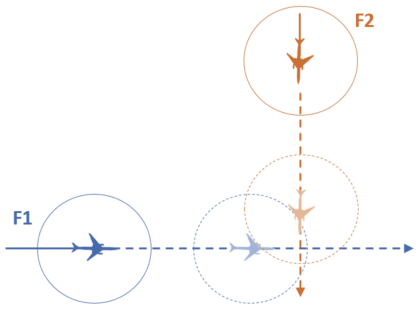


Figure 3.3: Aircraft' separation minima. Both vertical and horizontal separation. No other aircraft can be inside the cylinder at the same time. Source [1]

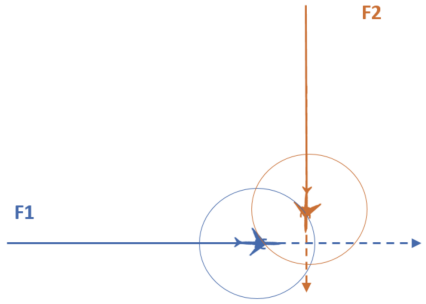
3.4.1 Strategic Deconfliction

The moment when aircraft infringe the separation minima, a loss of separation occurs. Hence, strategic conflict happens when two planned trajectories infringe the separation minima in any point in the future, see figure 3.4.b. To enable strategic deconfliction, every planned flight path point is checked against points of other flights for infringement of the separation minima [5].

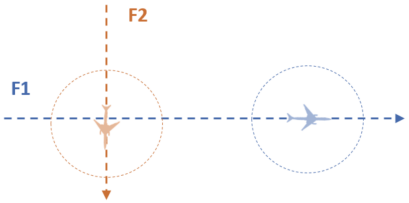
When a flight's departure is delayed, its trajectory shifts in time accordingly. Thereby, allocating new departure slots, can result in new conflicts between flight's trajectories that were separated in time beforehand. Delayed flights can only possibly be in conflict, if they are planned to arrive at the critical points with a time difference which is a multiple to the trajectory's time step. For instance, if flight F1 crosses a point 15 minutes after flight F2, there is no conflict, see 3.4.c. Conversely, if flight F2 is delayed 15 minutes, there is a strategic conflict, see 3.4.d.



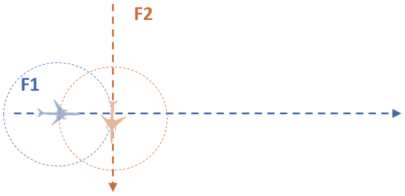
(a) Conflict between two aircraft converging to a violation of the minimum separation (indicated by surrounding circles)



(b) Loss of separation between two aircraft because of a violation of the separation minima at that point in time



(c) Two planned trajectories that are not in conflict. They are sufficiently separated because they cross with a time difference



(d) Two planned trajectories that are in conflict because there is a future point in time where they violate the separation minima

Figure 3.4: Strategic deconfliction, adapted from [5]

3.4.2 Conflict Probabilities

Precise prediction of conflicts, specially for large look-ahead times are hard to obtain due to inexact forecasts of aircraft positions. Besides ATFM delays, there are still many others factors such as meteorological conditions, human behaviour, mechanical issues, etc which can deteriorate flight's punctuality deviating the scheduled departure time. These departure times deviations are the largest temporal uncertainties on the ground. Distributions of departure times' deviation in Europe, show that nearly 40% of all fights deviate at least five minutes and delays greater than 15 minutes are on rising since last year, see the figure 3.5 which depicts the departure deviation distribution for the years 2016 and 2017.

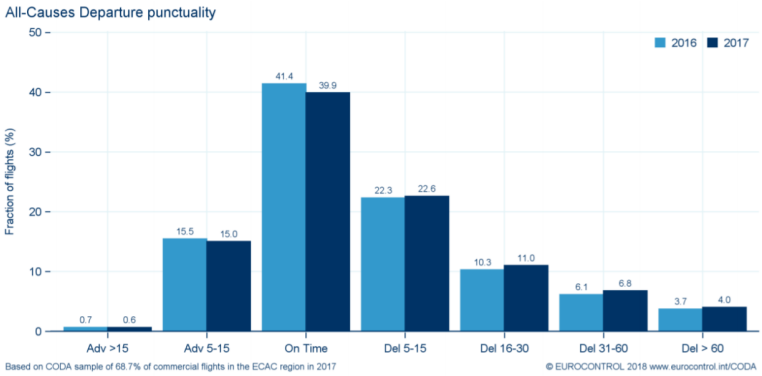


Figure 3.5: Distributions of departure times' deviation in Europe in 2016 and 2017, source [3]

Within just a period of time of 5 minutes, a commercial jet aircraft at typical cruise speed can cover a distance of more than 40 nautical miles, accounting for eight times the lateral en-route separation of 5 nautical miles. To measure uncertainties of departure times, a probability associated to potential

conflicts on the day of operation is computed, namely, conflict probabilities. Understanding the latter at this planning timing, allows assessing the potential for strategic trajectory deconfliction and flight plan pre-processing to increase flight safety [8]. Trajectory forecast before take-off is uncertain due to ground and airborne temporal uncertainties. These uncertainties are larger before the actual take-off than after. Consequently, the error of ground predictions are larger than the airborne ones as the figure 3.6 depicts.

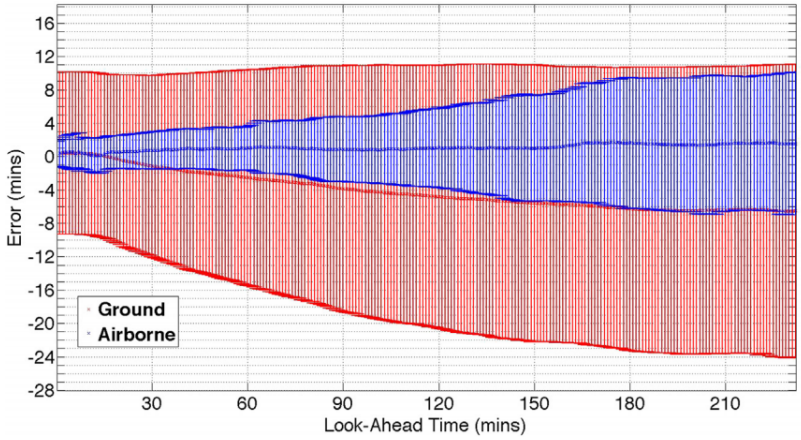


Figure 3.6: Ground (red) vs. airborne (blue) flight predictability, source [42]

Accurate take-off time prediction is of vital importance to be able to compute trajectory forecast with diminished errors [42]. Thereby, statistical departure time are chosen to represent all stochastic deviations from the planned trajectories without considering spatio-temporal airborne deviations. The latter are considered to be deterministic.

Uncertainty in time prediction of trajectories are represented by departure time uncertainty. Probabilities for deviations from planned departure time are inferred from respective statistics about planned and actual departure times. The probabilities for different departure times are given by the planned departure times and the probabilities for deviations from the planned departure time. Conflicts are spread in space, therefore potential conflicts between trajectories and the involved point pair are analysed geometrically. If the minimum separation is violated but there is a separation in time, there is still a conflict potential due to the stochastic character of the departure times. All potential conflicts with a maximum time difference of 120 minutes are taken into account. The conflict geometry in space and time can be described by the combination of all conflicted point pairs. The figure 3.7 illustrates two exemplary trajectory segments for two flights with the respective conflicted point pairs for every 10th conflict.

Point pairs from two flights are represented and the ones conflicted are linked with green lines showing the conflicted space and forming the conflict geometry. In this example, the peripheral points have fewer conflicts than the central points. The latter have more conflicts because the trajectories are crossing each other. The planned arrival time difference or each point point pair the planned arrival time difference is known from flight times and planned departure times. A spatio-temporal interpretation of potential conflicts yields exactly which trajectory points are in conflict with each others with their respective time difference. This is done by computing a conflicted time window, *i.e.* the conflicted time frames, which constitutes the set of all local time difference between the conflicted points from two flights' tra-

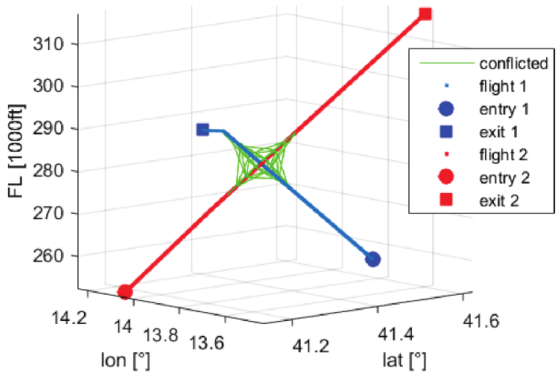


Figure 3.7: Points of two exemplary trajectory segments in a portion of a sector volume, source [8]

jectories. The conflicted time frames are combined with their departure time probabilities to compute strategic conflict probabilities.

In a nutshell, strategic conflict probabilities are the sum of the joint probabilities of conflicted departure time combinations within a time window.

3.4.3 Linearisation of the Quadratic Problem

ATFM with deconfliction is to be computed by using BIP allocating several departure timeslots. This optimisation problem has now quadratic conflict cost associated to each conflicted flight pair and the same linear capacity constraints associated to sector and aerodrome capacity.

Conflict cost

Every strategic conflict probability between two flights $f^{'}$ and $f^{''}$ with departure slots $d^{'}$ and $d^{''}$ has conflict cost $k_{f',d',f'',d''}$. This new coefficient represents the conflict delay cost for each pair of flight. It is given by the half of the probability of conflict costs of a pair $f^{'}$ and $f^{''}$ for every departure possible departure slot combination $d^{'}$ and $d^{''}$ as equation 3.15 shows.

$$k_{f',d',f'',d''} = \frac{1}{2} \cdot P_c(x_{f',d'}, x_{f'',d''}) \tag{3.15}$$

The conflict cost term depends on variables of two flights and therefore forms a quadratic objective. Hence, the objective function $z(x)$ now consists of both delay and conflict cost and takes now the form of the equation 3.16.

$$Z(x) = min\left(\sum_{f} \sum_{d} \omega_{f,d} \cdot x_{f,d} + \sum_{f'} \sum_{d'} \sum_{f''} \sum_{d''} k_{f',d',f'',d''} \cdot x_{f',d'} \cdot x_{f'',d''} \right) \tag{3.16}$$

Quadratic Term Linearisation

In order to solve the problem, there is the need to linearise the quadratic conflict-cost-term [8] because quadratic functions are non-linear and for that reason they are hard to solve. Unfortunately the non-linear

programming problems are *non-convex* which means that it is not possible to ensure for this problem that a *local minimum* is also a *global minimum* [9]. Thus, the quadratic costs are replaced by linear conflict cost constraints and surrogate variables.

To linearise the quadratic conflict cost, a surrogate variable has to replace the product $x_{f',d'} \cdot x_{f'',d''}$ and so equation 3.17 holds:

$$y_{f',d',f'',d''} = x_{f',d'} \cdot x_{f'',d''} \tag{3.17}$$

Surrogate Constraint

The linearisation of the conflict term shown in the equation 3.17, provides a new constraint, namely, conflict surrogate (3.18),

$$x_{f',d'} + x_{f'',d''} - y_{f',d',f'',d''} \leq 1 \tag{3.18}$$

Where the variables $x_{f',d'}$ and $x_{f'',d''}$ are the departure slots decision variables for a pair of conflicted flights and $y_{f',d',f'',d''}$ is the surrogate variable.

Thereby, the objective function takes the linear form:

$$Z(x) = min\left(\sum_{f} \sum_{d} \omega_{f,d} \cdot x_{f,d} + \sum_{f'} \sum_{d'} \sum_{f''} \sum_{d''} k_{f',d',f'',d''} \cdot y_{f',d',f'',d''} \right) \tag{3.19}$$

And the final binary optimisation problem is:

$$\begin{aligned} \min_{x} \quad & \mathbf{w}^{\top}\mathbf{x} + \mathbf{k}\mathbf{y} \\ \text{subject to} \quad & \mathbf{G}\mathbf{x} = \mathbf{e} \\ & \mathbf{A}\mathbf{x} \leq \mathbf{c} \\ & \mathbf{R}\mathbf{x} \leq \mathbf{i} \\ & \mathbf{P}\mathbf{x} \leq \mathbf{o} \\ & \mathbf{B}\mathbf{x} + \mathbf{M}\mathbf{y} \leq \mathbf{i} \quad \text{and} \quad \mathbf{x}, \mathbf{y} \in [0,1] \end{aligned} \tag{3.20}$$

3.5 Discussion

Within the software suite NFE, two formulations of the EATFM binary optimisation problem were modelled. The formulation 3.14 only observes delay costs in its objective function along with respective constraints however, the formulation 3.20 observes both delay costs and conflict costs in its objective funtion. Due to non-linearity nature of the conflict term in the objective function 3.16, the latter formulation was linearised given rise to another variable namely, surrogate variable in equation 3.17 and a another constraint namely, surrogate constraint in equation 3.18. These last two new elements introduce a great amount of information to the model. These formulations will serve as basis for the optimisation techniques used in this thesis exposed in the next two chapters.

Chapter 4

Solving Binary Integer Programming by Branch-and-Bound

This chapter aims to introduce optimisation theory and the framework SCIP. In the section 4.1, concepts such as Linear Programming, Simplex Method, Integer Programming, Branch-and-Bound and LP relaxation are presented. Thereafter, in section 4.2 the framework SCIP is presented as one optimisation framework to solve the ATFM problem.

Problem Dimension and Complexity

Binary problems with n variables have an exponential large number of possible solutions of 2^n. For that reason, it is not possible to check all solutions. Such problems are intractable[1], *i.e.* there is no existing polynomial-time algorithm that can solve them. Moreover, this BIP problem is proven to be NP-Hard [16, 43] (Non deterministic Polynomial acceptable) which means, it requires exponential runtime to be solved in optimality. Thus, computer programs cannot handle them efficiently taking a long time to run hence, there are some problems for which optimal solutions can't be found.

The ATFM problem in Europe, for one day, typically deals with around 30.000 flights associated with 10 different departure time-slots resulting, as product between flight movements and departure time-slots, in 300.000 decision variables and millions of possible conflicts. The great number of conflicts is due to the fact that for each pair of flights there are several points from their trajectories which have conflict probability [5]. There is an extremely large number of possible solutions, around two to the power of 30.000 plus millions of possible conflicts. This illustrates the interconnectedness mentioned earlier which requires a great computational effort for finding solutions.

[1] Or difficult

4.1 Mathematical Optimisation Theory

The development of mathematical optimisation has been ranked among the most important scientific advances of the mid-20th century. Thanks to the progress in this science field many thousands or millions of euros for many companies or businesses have been saved all around the world. Moreover, its use has been spreading rapidly in other sectors of society improving their efficiency.

4.1.1 Linear Programming

Linear Programming (LP) makes use of a mathematical model to describe the problem of concern. In a LP model, all its mathematical functions are required to be linear. Using these functions, a planning of activities in order to obtain an optimal result, *i.e.* a best result that reaches the specific goal according to the mathematical model, among all feasible alternatives is to be programmed.

The most frequent type of application of LP involves the general problem of allocating limited resources among competing activities. More precisely, this problem concerns selecting the level of certain activities that contest for scarce resources that are necessary to accomplish those activities. The decision making of activity levels dictates the quantity of each resource that will be consumed by each activity. The variety of situations to which this application is used is in fact diverse, stretching several science and business fields. However, all of these situations share a common ingredient that is to say, the need for allocating resources to activities by choosing the levels of those activities.

The components of LP along with their interpretation for the general problem of allocating resources to activities is described below.

Z = value of overall measure of performance.

x_j = level of activity j for $j = 1, 2, ..., n$.

c_j = increase in Z that would result from each unit increase in level of activity j.

b_i = amount of resource i which is available for allocation to activities for $i = 1, 2, ..., m$.

a_{ij} = amount of resource i consumed by each unit of activity j.

Due to the decision making nature of this type of problems, the levels of activities x_j are named the decision variables. The table 4.1 shows in a structured way, the data involved in a LP model. The values of c_j, b_i and a_{ij} are the input constants for the model.

Therefore, a linear optimisation problem basically consists of decision variables x_j, a objective function 4.1, functional constraints 4.2 and non-negativity constraints 4.3. The decision variables may take values within a defined range and thus determine the problem-specific decision options. They are linked by coefficients a_{ij} with the individual constraints. A constraint limits the admissible values b_i of the linked decision variables. In addition, the decision variables with their cost coefficients c_j form the objective function 4.1. The objective function value of a solution is the value of overall measure of performance to the linear problem which is determined by the sum of the decision value solution values multiplied by

Resource	Resource usage per unit of activity				Amount of resource available
	Activity				
	1	2	...	n	
1	a_{11}	a_{12}	...	a_{1n}	b_1
2	a_{21}	a_{22}	...	a_{2n}	b_2
.					.
.	...	...	...	...	.
.					.
m	a_{m1}	a_{m2}	...	a_{mn}	b_m
Contribution to Z per unit of activity	c_1	c_n	...	c_n	

Table 4.1: Data needed for a linear programming model involving the allocation of resources to activities, source [9]

their cost coefficients c_j. The linear problem in standard form consists then of linear constraints and a linear cost function.

(P.1)

$$\text{Maximise} \quad Z = z(x_1, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n \tag{4.1}$$

$$\text{subject to} \quad a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \leq b_1 \tag{4.2}$$

$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \leq b_2$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \leq b_m$$

$$\text{and} \quad x_1 \geq 0,\ x_2 \geq 0,\ ..., x_n \geq 0 \tag{4.3}$$

ATFM Linear Programming

The ATFM problem has a similar structure of the standard LP with some variations. For the present problem, the objective function is to be minimised 3.19 rather than maximised and there is one more type of constraint, the start constraint 3.4.

Now that the necessary LP terminology and notation is clarified, it is time to associate those to the ATFM problem's terms. The table 4.2 summarizes the mentioned association. The resources time-slots d are limited by the sector capacities c, airport o departure and i arrival capacities and they are to be allocated to all flights (F) activities flights f. Thereby, the corresponding level of activity is $x_{f,d}$ responsible for the allocation to time-slot d in flight f. To evaluate the quality of the solutions, the overall performance is measured in terms of delay costs and the conflict costs.

ATFM Problem	General Problem
Time-slots d	Resources
Sector capacities c, Airport o departure and i arrival capacities	m resources
Flights f	Activities
F	n activities
Allocation of time-slot d in flight f, $x_{f,d}$	level of activity j, x_j
Delay/conflict cost	Overall measure of performance Z

Table 4.2: ATFM problem's terminology for LP

Terminology of the solutions

The set of values for the decision variables $x_{f,d}$ is called a solution, regardless of whether it is a desirable or even an allowable choice. Different types of solutions are then differentiated:

- **Feasible solution:** is a solution for which all the constraints are satisfied;
- **Infeasible solution:** is a solution for which at least one constraint is violated;
- **Optimal solution:** is a feasible solution that has the most favourable value of the objective function.

The most favourable value is the largest value if the objective function is to be maximised, whereas it is the smallest value if the objective function is to be minimised. The latter is the case in the present problem. Therefore, the goal of linear optimisation is to find the best feasible solution, as measured by the value of the objective function.

Assumption of LP

All the assumptions of LP actually are implicit in the model formulation in 4.1. According to [9], they are:

- **Proportionality:** all the constants and decision variables are proportional in the same level;
- **Additivity:** every function in a LP model is the sum of the individual contributions of the respective activities.
- **Divisibility:** in a LP problem, decision variables are allowed to have any values, including non-integer values, that satisfy the functional and non-negativity constraints. So it is assumed that the activities can be run at fractional levels.
- **Certainty:** the values assigned to each parameter of a LP model are assumed to be a known constant.

4.1.2 Simplex Method

Linear optimisation problems such as the ATFM problem generally have many possible solutions as stated in the beginning of this chapter. Therefore, it is impractical to compute the objective function value of all solutions in order to select the best from the feasible ones. To improve the search for the best solution, an efficient way to find quickly the optimal solution, namely simplex algorithm, was developed in 1947 by George B. Dantzig.

The simplex method is an algebraic procedure with underlying concepts of geometry developed to operate on linear programs in the standard form like in P.1. The algorithm searches adjacent vertices, namely Corner-Point Feasible (CPF) solutions, of the feasible solution set, namely polyhedron, in sequence so that at each vertex the objective function improves or is unchanged [9]. It is only possible to apply the simplex method in a problem only and only if it complies with the following requirements:

- The problem must have feasible solutions;
- The problem must have a bounded feasible region;
- The problem with a bounded feasible region must allow that every feasible solution can be represented as a convex combination[2] of the CPF solutions.

Due to the convexity of the polyhedron, it is guaranteed that the connecting line lies between any point and the optimal point within the polyhedron. If there is a path between any CPFs solutions of the polyhedron and the optimal vertex, *i.e.* the optimal CPF solution, it follows that there exists an adjacent CPF solution with a better objective function value. For every LP problem that has feasible solutions and a bounded feasible region, three key properties of the polyhedron convexity, and therefore the CPF solutions, hold:

- **Property 1:** If there is exactly one optimal solution, then it must be CPF solution. If there are multiple optimal solutions, then at least two must be adjacent CPF solutions;
- **Property 2:** Since the optimal solution is a basic solution or is located on a vertex of the polyhedron there are only a finite number of CPF solutions;
- **Property 3:** If a CPF solution has no adjacent CPF solutions that are better, then there are no better CPF solutions anywhere in the feasible region. Therefore, accordingly to property 1, a CPF solution is guaranteed to be an optimal solution.

The simplex algorithm consists of three main steps:

1. Determining a feasible solution as a starting point, namely Basic Feasible (BF) solution. Whenever possible, the simplex method chooses the *origin* (in which all the decision variables are equal to zero) as the BF solution. This eliminates the need to use algebraic procedures to find a starting feasible solution, specially when there are too many variables;

[2] A weighted average of two or more solutions (vectors) where the weights are nonnegative and sum to 1 [9]

2. Locating an adjacent base corner with an improved objective function value. The simplex method moves along the edges of the feasible region and only checks the adjacent CPF solutions. The directions of these movements are determined by the rate of the solution improvement, *i.e.* the direction of the movement in which the solution has the highest rate of improvement, is the one chosen to search for a better CPF solution;

3. Finding the optimal solution by iteratively repeating the second step. The way that the simplex method finds the optimal solution is by applying optimality tests. The latter makes use of the third polyhedron geometric properties to find others CPF solutions. This test, dictates that if a CPF solution has no adjacent CPF solutions that are better, then it must be an optimal solution. The simplex therefore terminates if no adjacent CPFs with improved objective function value are found.

Using a computer, the simplex finds an optimal solution tremendously fast due to the fact that it only focus on CPF solutions which represent a very small minority of the feasible region. The optimality test reduces even more the search time because it doesn't require to examine all the CPFs thanks to their adjacency property. The simplex algorithm of Dantzig, creates the simplex tableau in the solution process, which is based on the simplex coefficient matrix. The tableau is fully updated in each iteration of the algorithm. In the so called revised simplex algorithm, only the base matrix is updated, which is much smaller than the entire simplex tableau. Furthermore, the revised simplex facilitates the use of sparse matrices. This avoids many arithmetic operations with null elements when updating the tableau.

4.1.3 Integer Programming

One key limitation that prevents many more applications is the assumption of divisibility, which requires that non-integer values be permissible for decision variables. In many practical problems, the decision variables actually make sense only if they have integer values. For example, it is often necessary to assign people, machines, and vehicles to activities in integer quantities. If requiring integer values is the only way in which a problem deviates from a linear programming formulation, then it is an Integer Programming (IP) problem.

The mathematical model for integer programming is the linear programming model in P.1 with the one additional restriction that the variables must have integer values.

Binary Integer Programming

However, there are wide range of problems which have to face *yes-or-no* decisions, thereby a slightly different model must be used namely Binary Integer Programming (BIP). For these type of problems, a planning of activities is to be done to obtain an optimal solution, thus there is the necessity for allocating resources (time-slots) to activities (flights) by choosing the levels of those activities. For this, there is the need to use decision variables associated to their respective costs, see [9]. Since the nature of the decision is *yes-or-no*, a BIP problem arises with binary decision variables which means that they are restricted to integer values, namely, 0 or 1. Thus, the ith *yes-or-no* decision would be represented by x_i such that:

$$x_i = \begin{cases} 1, & \text{if decision } i \text{ is yes} \\ 0, & \text{if decision } i \text{ is no} \end{cases} \tag{4.4}$$

4.1.4 Linear Programming Relaxation

Because LP problems are easier to solve than IP problems, algorithms to solve the latter incorporate the Simplex Method. This is called LP relaxation which is the LP obtained by deleting from the current IP problem the constraints that require variable to have integer values [9].

There are three main determining factors of computational difficult for solving an IP problem:

- Number of integer variables. For a BIP problem with n variables have 2^n solutions to be considered and thus an exponential growth of the difficulty of the problem arises;
- Number of constraints. In some cases, increasing the number of constraints can possibly decrease the computational time;

To solve a BIP, three main steps could be done:

1. Apply LP relaxation. The integer variables are now non-integer;
2. Apply Simplex Method;
3. Rounding the non-integer values to integers in the resulting solution.

However, there are two hazards which deserve a considerable awareness when solving a BIP problem. Firstly, the optimal solution from the relaxed problem, *i.e.* the LP problem, is not necessarily feasible after it is rounded. It is difficult to perceive in which way the rounding should be done to retain feasibility. In some situations changing the value of some decision variables after rounding is necessary in order to retain feasibility. However, in this problem there are too many variables and constraints thus it's extremely difficult to apply such a procedure. Secondly, there is no guarantee that the rounded solution will be the optimal integer solution. In fact, it may even be far from optimal in terms of the value of the objective function.

Because of these two hazards, a better approach for dealing with BIP problems that are too large to be solved is to use one of the available heuristic algorithms. Although these algorithms are somewhat efficient for large problems, they are not guarantee to find an optimal solution.

4.1.5 Branch-and-Bound

To solve such a large BIP problem an algorithmic approach must be used. One option, is to use Branch-and-Bound methodology. Firstly, this methodology uses LP relaxations to ignore the integer constraints that require variables to have integer values. Secondly, solves the problem as a linear programming problem by using the Simplex Method. Since it has fewer constraints, its optimal solution provides a lower bound (LB). The easiest strategy is to use the simple heuristic rounding which rounds each

resulting solution value to its nearest integer value providing an upper bound (UB). However, there is no guarantee that a relaxed solution is necessarily feasible after it is rounded. The goal is to do several iterations to try to improve the bounds. Take this simplified example for instance:

$$\begin{aligned} &\text{minimise} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{Ax} \leq \mathbf{B} \\ &\text{and} && \mathbf{x} \text{ is a vector of binary variables } x_i, \text{ for } i = 1, 2, ..., n. \end{aligned} \tag{4.5}$$

The objective function $f(\mathbf{x})$ is linear and it is bounded by the constraint $\mathbf{Ax} \leq \mathbf{B}$. Applying the Branch-and-Bound technique just for one iteration, different bounds are obtained as shown in the Figure 4.1.

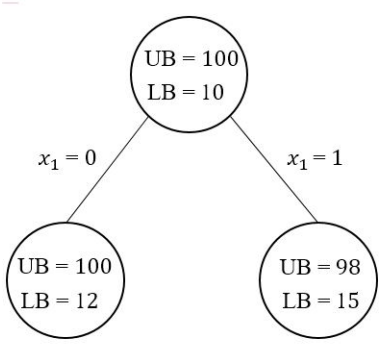


Figure 4.1: Example of one iteration using Branch-and-Bound with hypothetical values

Hypothetically, the first incumbent for the whole problem is $Z^* = 100$ which represents the UB of the "worst" solution. The LB is the optimal solution for the relaxation of the whole BIP problem therefore the optimal solution which observes integrability can not be better. In the iteration shown in 4.1, the variable x_1 is fixed to 0 on the left node and to 1 on the right node, and subsequently two branches are obtained with different bounds. The branch on the right which x_1 is fixed to 1, improved the UB from 100 to 98. The branch on the left which x_1 is fixed to 0, improved the LB from 10 to 12. From the respective branches, we update the incumbent to $Z^* = 98$ and the global lower bound to $LB = 12$. The difference between the UB and LB namely, optimality gap, was reduced like shown in the Figure 4.2.

To reduce even more the optimality gap (*i.e.* decrease the UB and increase the LB), it is necessary to do several iterations until the UB and the LB meet each other at the optimal solution. Therefore, the main goal is to find good feasible solutions in between the optimality gap in an acceptable time-frame.

LP relaxation weak lower bounds

The linearisation of the conflict term exposed in the previous chapter on page 32 equation (3.17), provides a new constraint, namely, conflict surrogate (4.6),

$$x_1 + x_2 - y \leq 1 \tag{4.6}$$

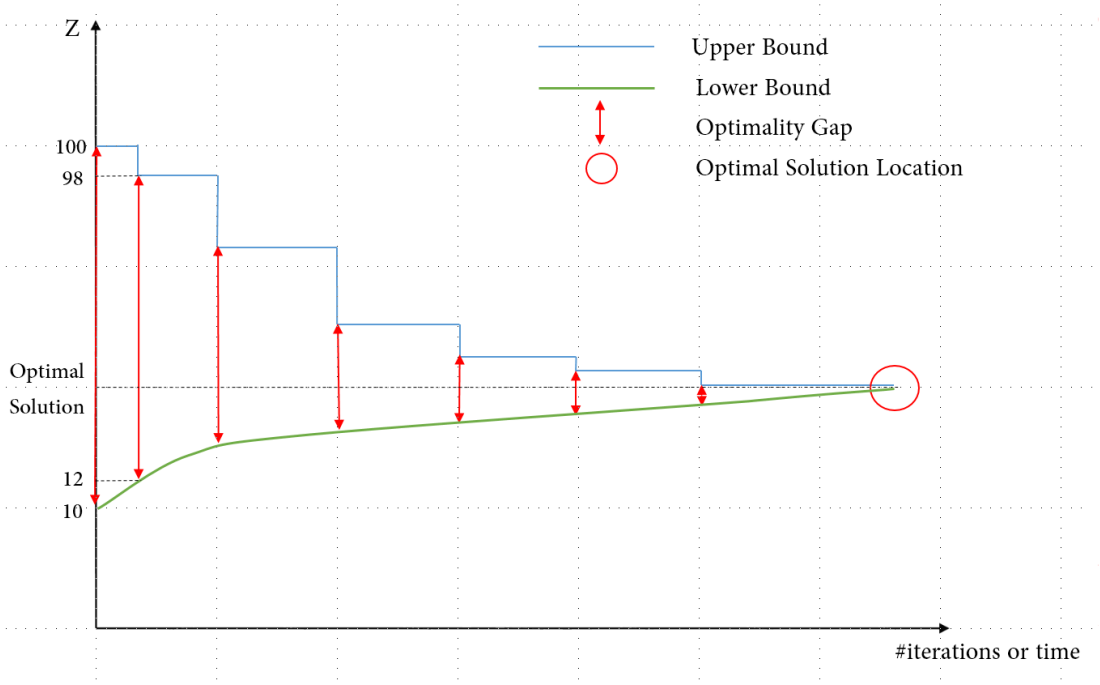


Figure 4.2: Conceptual visualisation of the lower bound and upper bound. The optimality gap is reducing along time.

Where the variables x_1 and x_2 are the decision variables, for two different flights and y is the surrogate variable, *i.e.* the conflict variable, which represents the quadratic conflict term of the objective function. This constraint is handled by the optimisation framework to be used which follows the logical conjunction operation. The truth table of this operation is adapted to the ATFM problem, which is to be minimised, as it follows in the table 4.3.

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

Table 4.3: Conflict surrogate bitwise operation

Accordingly to the above truth table, the surrogate variable takes the value one if and only if both of decision variables value is one. In another words, there is a conflict if and only if two flights allocated to their respective departure slots have a probability of conflict. Using the LP relaxation, the inequality (4.6) can take the following result:

$$x_1 = \frac{1}{2} \text{ and } x_2 = \frac{1}{2} \\ \text{then } y = 0 \tag{4.7}$$

This linearisation provides weak lower bounds. This means that conflict cost of relaxed conflicts might

not be taken into account. Therefore, one option is to use cutting planes in the available framework to improve the lower bounds [9].

4.2 SCIP - Solving Constraint Integer Programming

Even though this BIP problem is bounded, a very large number of solutions can be found in the feasible region, thus it is imperative to use a smart and structured search procedure so that only a tiny fraction of the feasible solutions actually need to be examined. Among the available search procedures, a Branch-and-Bound framework is the one proposed to be used. These frameworks are widely used in IP problems, but they do not guarantee to find an optimal solution. However, they do tend to be significantly more effective than the rounding approach in improving the UB and the LB.

The Branch-and-Bound framework to be used is SCIP (Solving Constraint Integer Programs) developed by the Zuse Institute Berlin (ZIB). In this framework, Constraint Programming is incorporated in the BIP problem to provide a compact model and specialised constraints handlers methodologies for this complex problem. With more than 500 000 lines of code, SCIP is implemented as C callable library which mimics object oriented programming. It is used for Constraint Integer Programming (CIP), mixed integer programming (MIP) and mixed integer non-linear programming (MINLP). SCIP is freely available in source code for academic and non-commercial purposes. Among the non-commercial solvers for MIP, this framework is currently one of the fastest, see figure 4.3.

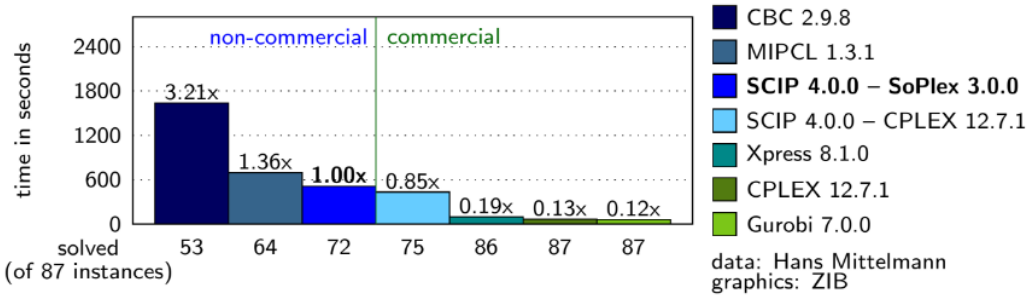


Figure 4.3: MIP solver benchmark (1 thread): Shifted geometric mean of results of Hans Mittelmann homepage http://plato.asu.edu/ftp/milpc.html on 14/Apr/2017. Unresolved or failed instances are accounted for with the time-frame limit of 2 hours. Source: http://scip.zib.de/ (Posted on 25/Sep/2017).

It allows a total control of the solution process and the access of detailed information down to the core of the solver. For a more detailed description of SCIP, please refer to the comprehensive documentation on the homepage [44]. SCIP has a modular structure, so most of the functionality is provided by C++ wrapper classes for user *plug-ins*.

4.2.1 Program Flow of SCIP

The solution process of a problem solved with SCIP, involves various stages until it finds the optimal solution. The stage in which SCIP is located is always clear and the different modules can only be called up in appropriate stages. The solution process of SCIP for a linear problem is shown in figure 4.4. The individual stages of the solution process are described below.

Problem Specification: Before the actual solution process the user has to specify his problem. A SCIP-LP object is created to which constraints and variables are added. The variables are linked by means of coefficients with the constraints.

Transform problem: After calling the `SCIPsolve(scip)` solution function with the `scip` object as input, firstly a copy of the problem that created the transformed problem is created. From now on, all operations are performed on the transformed problem. The purpose is that the original problem is not changed and the validity of the solution is therefore always verifiable on the original problem.

Pre-solving: At the beginning of the solution, this stage is started to simplify the problem. There is a variety of methods available. For example, unnecessary variables and constraints are removed from the problem.

Primal Heuristics: In this stage, methods based on experience on looking for a valid solution are used. Although it is possible to find a solution very quickly, finding an acceptable solution is not guaranteed. In the course of the solution process, primal heuristics can always be called.

LP Solving: SCIP sends the `scip` object in the next step to the solver SoPlex. The latter solves the linear problem using the revised Simplex Method and writes the optimal solution and the dual variables in the `scip` object.

Optimal Solution: SCIP stops its solution process when the optimal solution was found or a termination condition which defines when to stop the solution process was satisfied.

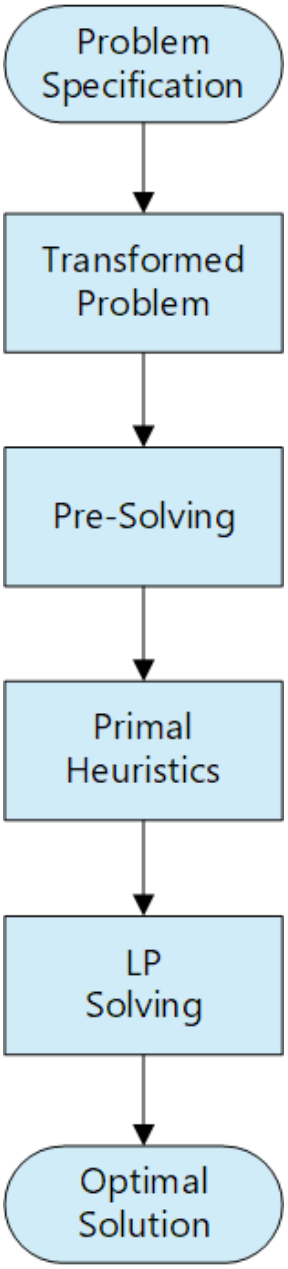


Figure 4.4: Flowchart of SCIP' solving process

4.2.2 Primal Heuristic in SCIP

The core of SCIP, is a framework that provides the infrastructure to implement adjustable search algorithms incorporated in the Branch-and-Bound tree search. SCIP is highly customised and allows the user to have total control of the solution process. Moreover, SCIP includes a large library of default algorithms to control the search which are part of external *plug-ins* which interacts with the framework through a very detailed interface. The goal of those *plug-ins* is to allow the user to enrich the SCIP solution process. SCIP 3.2.1 offers more than one hundred available *plug-ins* ready to be used which can be combined to enhance the solution process. The main advantage of using this *plug-in* approach is that facilitates the implementation of SCIP self-contained solver components.

Among all of them, the primal heuristic user *plug-in* is of interest. In SCIP 3.2.1 there are 23 primal heuristics implemented at the user's disposal. This type of *plug-ins* are designed to find feasible solutions in the transformed problem. They can be viewed as a module of SCIP which is to be integrated in the SCIP infrastructure. In order to allow a good interplay between its solution process and the module itself, primal heuristics have a set of properties settings which defines when modules are called in the Branch-and-Bound tree search. Therefore, the user has total control of when to call its module by choosing a combination of properties. The most relevant properties are described below

- **Priority (`priority`):** Prioritises the calling of the activated primal heuristics;
- **Frequency (`freq`):** Defines the ”jump” from a depth level to another one;
- **Frequency offset (`freqofs`):** Defines the depth level of the branching tree at which the primal heuristic is called for the first time;
- **Maximum depth (`maxdepth`):** Defines the maximum depth at which the primal heuristic is called;
- **Timing (`timing`):** Defines the entry point of the primal heuristic in the solving process.

To better understand how these properties influence the solving process, three hypothetical primal heuristics with the respective properties set are exemplified in the table 4.4.

	Primal heuristics settings		
Properties	Heuristic 1	Heuristic 2	Heuristic 3
`priority`	100	200	50
`freq`	0	1	8
`freqofs`	0	2	2
`maxdepth`	0	10	”no limit”
`timing`	”before node”	”after node”	”during loop”

Table 4.4: Hypothetical heuristics properties

The respective primal heuristics execution calls in the solving process can be visualised for the same branch-and-bound tree search of an hypothetical problem illustrated in the figure 4.5. In the figure, it

is possible to visualise some of the different levels k of the tree search. The latter, is represented by branches that connects the nodes. Each node fixes a set of variables and solves a different sub-problem of the original one. Each sub-problem is solved by the LP solving loop represented in the left side of the figure. In the first level, $k = 0$, namely the root node, the first LP relaxation is solved. The above heuristics execution calls are identified by colour shapes in the same figure.

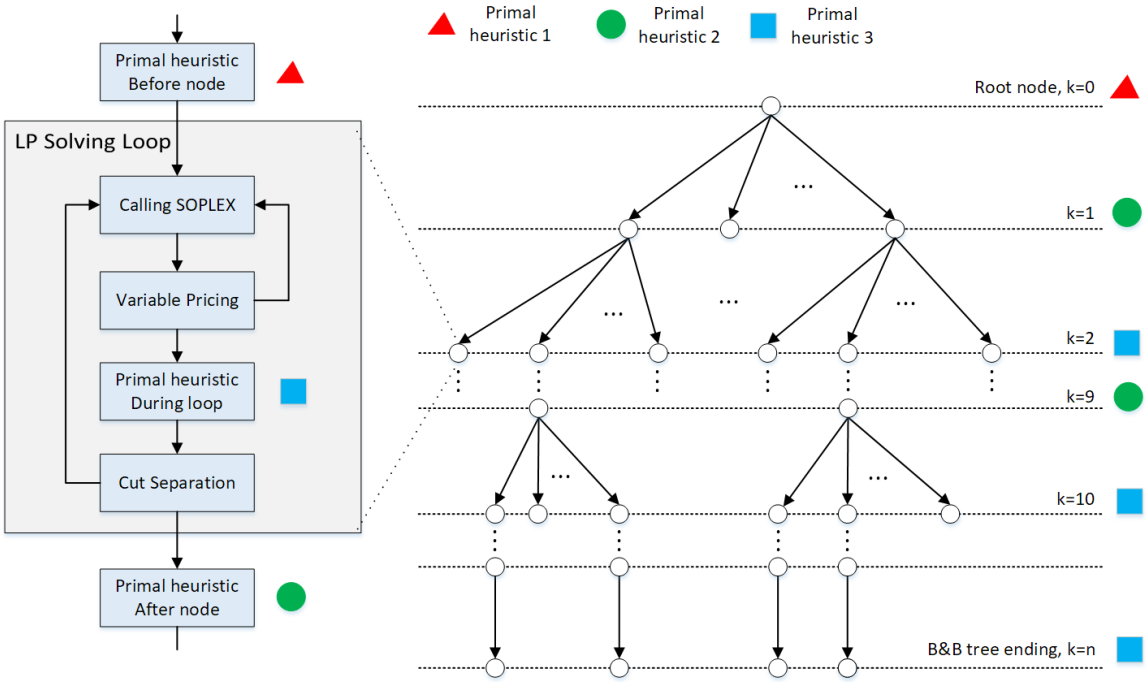


Figure 4.5: Branch-and-Bound tree search and primal heuristics execution calls in the SCIP solving process

The primal heuristic 1, represented with a red triangle, is only executed once in the entire tree search because its *frequency* and *frequency offset* properties are set to zero. Its *timing* property is set to ”before node” which means that before the solving of the first LP relaxation, the heuristic 1 is executed.

The primal heuristic 2, represented with a green circle, is called for the first time in the level $k = 1$ because its *frequency offset* property is set to 1. Due to its *frequency* set to 2, it means that it will be called every two levels of the tree search. The last call will happen at level $k = 9$ because its *maximum depth* is set to 10. Its timing property is set to ”after node” which means that its called after the sub-problem LP relaxation is solved.

Finally, the primal heuristic 3, represented with a blue square is called for the first time in the level $k = 2$ because its *frequency offset* property is set to 2. Due to its *frequency* set to 8, it means that it will be called every eight levels of the tree search. Since its *maximum depth* is set to ”no limit”, the primal heuristic will be repeatedly called until the last level $k = n$ of the tree search. Its *timing* property is set to ”during loop” which means that its called inside the sub-problem LP solving loop.

The *priority* property, as its name implies, prioritises the primal heuristics in each node. Therefore, in case the exemplified heuristics were to be activated in the same problem and run in the same node, the heuristic 2 would be first one to be called followed by heuristic 1 and finally heuristic 3.

4.3 Discussion

The branch-and-bound employed by SCIP allows to totally control the solution process and to access detailed information down to the guts of its solver. Therefore, it is possible to implement new heuristics as SCIP modules and incorporate them in the solution process and call them in any moment as many times in the tree search shown in the figure 4.5. Taking advantage of this optimisation software suite, a metaheuristic is to be implemented as a module of SCIP to improve even further the problem's known UBs. This implementation will be depicted in the next chapter.

Chapter 5

Genetic Algorithm

During the 1980s and 1990s, the trend shifted towards the development of nature-inspired evolutionary metaheuristics, known as Evolutionary Algorithms (EAs). These metaheuristics comprise a great variety of different concepts and paradigms such as Genetic Algorithms (GAs), capable of producing sub-optimal solutions to NP-hard problems in a comparatively shorter amount of time.

GAs were developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg has been tried on various optimisation problems with a high degree of success. Genetic Algorithm (GA) is a metaheuristic inspired by the natural phenomenon named 'survival of the fittest', *i.e.* individuals with variations that inherent a survival advantage through improved adaptations to the environment are most likely to survive to the next generation.

Genetic Algorithms are sufficiently randomized in nature performing much better than other search methods such as random local search in which various random solutions are tried keeping track of the best so far, as they exploit historical information as well [45].

5.1 Motivation to Solve the EATFM Problem with a Genetic Algorithm

Genetic Algorithms have the ability to deliver a 'good-enough' solution 'fast-enough'. This makes GAs attractive for use in solving optimisation problems. The reasons why GAs are needed are as follows. As mentioned in the chapter, ATFM problem is NP-Hard, which means even the most powerful computing systems take an immense amount of time to solve the problem to optimality or to near-optimality. In such a scenario, GAs prove to be an efficient tool to provide usable near-optimal solutions in a short amount of time.

Traditional calculus based methods work by starting at a random point and by moving in the direction of the gradient, until the top of the hill is reached. This technique is efficient and works very well for single-peaked objective functions like the cost function in linear regression. But, in most real-world situations, a very complex problem arises called as landscapes, which are made of many peaks and many valleys, which causes such methods to fail, as they suffer from an inherent tendency of getting

stuck at the local optima, see figure 5.1.

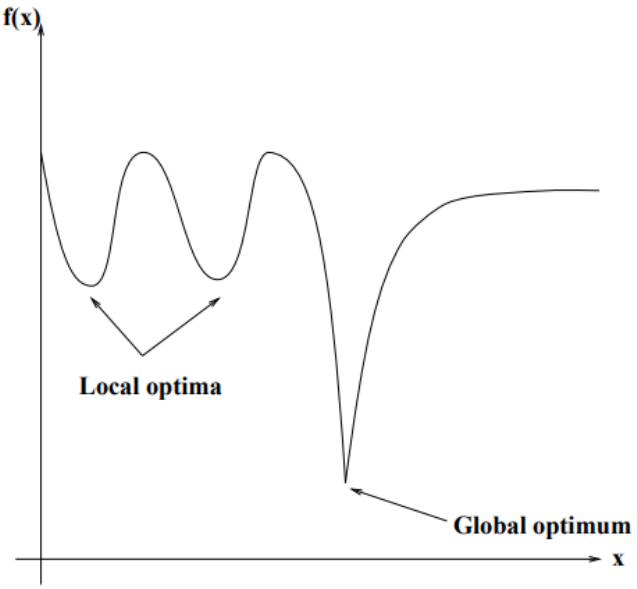


Figure 5.1: Local optima and global optimum, source [16]

GAs have several advantages over other metaheuristics such as:

- Search efficiently in problem with large spaces and large number of parameters involved;
- Robust with respect to the complexity of the search problem;
- Use a population of solutions instead of searching only one solution at a time.

However, they also present some limitations such as:

- Fitness values are calculated repeatedly which might be computationally expensive for some problems;
- Due to their stochastic nature, there are no guarantees on the optimality or the quality of the solution;
- A good understanding of the problem is required in order to properly implement, otherwise GA may not converge to the optimal or near-optimal solution.

5.2 Fundamentals of Genetic Algorithm

This section classifies GAs and introduces the basic terminology required to understand them. GA metaheuristic can be classified as according to some criteria such as:

- **Nature inspired:** it is based on the concepts of natural selection and genetics;
- **Memory usage:** it uses a memory that contains some information extracted on-line during the search which is limited to the population of solutions;

- **Stochastic:** different final solutions may be obtained from the same initial solution;
- **Population-based search:** in this algorithm a whole population of solutions is evolved;
- **Iterative:** start with a population of solutions and transform it at each iteration using some search operators.

5.2.1 Terminology

Before beginning a discussion on GA structure, it is essential to be familiar with the evolutionary algorithms' terminology.

1. **Population:** It is a subset of all the possible solutions in an encoded form to the given problem.
2. **Individual or chromosome:** is one such encoded solution to the given problem.
3. **Gene:** is one element position of an individual.
4. **Allele:** a variant of a gene, *i.e.* the value of a symbol in a specified position of the genotype.
5. **Genotype:** represents the population in the computation space in which the solutions are represented in a way which can be easily understood and manipulated using a computing system.
6. **Phenotype:** represents the population in the actual problem' solution space in which solutions are represented in a way they can be read by the actual mathematical model's objective function.
7. **Decoding and Encoding:** decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming an individual/chromosome from the phenotype to genotype space.
8. **Fitness:** An objective function associates a fitness value with every individual indicating its suitability to the problem.
9. **Genetic Operators:** these operators are responsible for altering the genetic composition of the individual. These include crossover and mutation which mimic the living beings' reproduction process.

5.2.2 Basic Structure

A basic structure of GA is to be described. GA starts with a set of initial solutions, namely initial population, which are represented by individuals. Using a selection technique, an individual from the population is picked depending on its fitness to be part of the parents population. The latter are subjected to reproduction by applying crossover and mutation operators to generate new offsprings which are constructed from the different attributes of individuals belonging to the current population. Finally, the resulting offsprings replace the existing individuals in the population and the process repeats iteratively. The figure 5.2 illustrates one iteration or generation of the described algorithm.

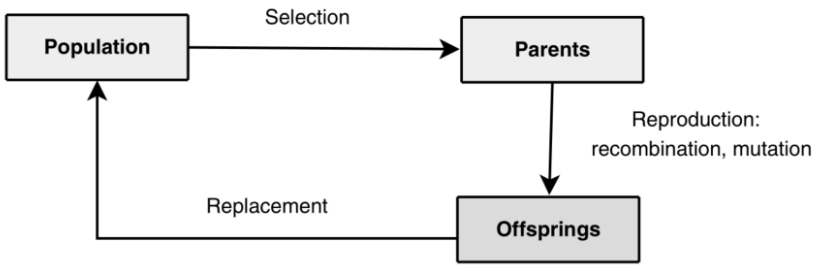


Figure 5.2: A generation in GAs, source [45]

A generalized pseudo-code for a GA is shown in the following algorithm scheme.

Algorithm 1 Template of the Genetic Algorithm

```
procedure GA
    initialize population
    evaluate initial population
    while stopping criteria is reached do
        parents selection
        crossover
        mutation
        evaluate offsprings
        population replacement
        find best
    end while
    return best individual
end procedure
```

5.3 Genetic Algorithm Available Frameworks and Libraries

In this section, the motivations for using a software framework for GA are outlined.

According to Talbi, there are three main approaches for the development of a GA framework.

- **From scratch:** One can develop it from scratch by creating the data-structure and designing every function of the program;
- **Only code reuse:** Reused third-party code available either as free individual programs or as already designed libraries. This allows a more flexible coding since the user calls the libraries at his will offering a greater control;
- **Framework usage:** Both design and code are reused programs available. Frameworks usually offer greater range of features options. With this approach there is no need to call user's code, instead the framework calls it. The latter controls the overall program's flow control allowing extensibility of the framework functionalities. However, the user should not modify the source code.

In order to redo as little code as possible to deal with different optimisation problems the best ap-

proach is to reuse both design patterns[1] and code. Useful design patterns related to a specific domain are in general implemented frameworks. Therefore, a suitable framework with design pattern related to metaheuristics, more specifically to Evolutionary Computation (EC), is to be used to implement a GA and to be included in SCIP for the ATFM problem.

Several metaheuristic's frameworks have been proposed in the literature. They mainly focus on a given metaheuristic family such as EAs. In the Web, there exists several libraries for ECs along its source code level available to any user. In order to choose the most suitable framework, some criteria must be met such as:

- **Accessibility and documentation support:** only non-commercial open source software suites are regarded. Moreover, the framework code should be well documented;
- **Maintainability:** it is important for a framework to be updated to keep up with the most recent developments such as new versions of the standard for the programming language C++.
- **Programming language preference and complexity:** due to the fact that the problem is modelled inside SCIP optimisation suite which in turn is programmed in C and has wrappers classes[2] for C++, frameworks implemented in C or C++ are preferred avoiding having to write a new interface. Also, programming language complexity must be taken into account since the author of this work is a novice in C++.
- **Maximum design and code reuse:** the framework must provide for the user a whole architecture design of the GA implementation approach and at the same time to reduce the necessity to redo code *i.e.*, to develop the minimal problem-specific code. Thereby, the framework must simplify considerably the development of GA and reduce its development time;
- **Flexibility and adaptability:** because GA implementation is problem-specific, the framework must allow for the user to easily add new features or change existing ones without implicating other components. Furthermore, the framework must allow a good integration in SCIP.
- **Robustness:** the execution of the algorithms must be robust to guarantee the reliability and the quality of the results;
- **User friendliness and efficiency:** The framework must be easy use and does not incorporate an additional cost in terms of time or space complexity. Moreover, it must preserve the efficiency of a special-purpose implementation.

According with the above criteria, an exhausting search was made and the most promising frameworks and libraries were identified. Among them, two powerful frameworks stand out such as **Evolving Objects**[3] and **Open BEAGLE** which are free software, designed to provide an EC environment that is programmed in C++, generic, robust, user friendly and efficient.

[1] The invariant part of solution methods to standard problem belonging to a specific domain are captured into special components named design patterns [46]

[2] Is a term meaning a class that "wraps around" a resource *i.e.*, that manages the resource.

[3] Population based metaheurisctis module of Paradiseo, a software framework for metaheuristics

Unfortunately, Evolving Objects is not maintained, the latest documentation update was back in 2012. It is also not recently used, the last found project that used Evolving Objects was in 2011 on a multi-objective optimisation problem by Liefooghe et al.. Therefore, maintainability is compromised.

Open BEAGLE framework has several versions being the most powerful one an *alpha* version released in 2010. Since then few maintenance efforts have been made and nowadays its project is abandoned supported by a poor documentation. There exists a stabilised version, however it is not updated since 2012. Once again, maintainability is compromised. Also, there are very few projects using this framework.

A better way to reuse the code of existing metaheuristics is through libraries [48]. The code reuse through libraries is better because these libraries are often well tried, tested, and documented, thus more reliable allowing a better maintainability and efficiency [45]. Others less complex and free frameworks and libraries were identified, such as *GALGO 2.0* [49], *jMetalCpp* [50], *OpenGA* and *GAlib*.

- **GALGO 2.0** although its simplicity, this set of template libraries doesn't allow chromosomes representation with more than 64 bits per string, precluding the pretended data structure implementation with thousands of bits (each bit representing a ATFM delay slot per flight) per chromosome;

- **jMetalCpp** is a framework with several metaheuristics libraries formerly developed in Java language and in 2012 a C++ version was released being often maintained. Unfortunately this version is poorly documented. Furthermore, only two applications using this framework written in C++ were found [51] and [52], integrated in a completely different environment from SCIP. Thus a possible integration of this framework in SCIP is not recommended;

- **OpenGA** is a free C++ library for GA [53]. It was recently developed in 2017 and it allows a great flexible customisation of the chromosomes and the GA operators. The library is placed in a single file which makes it easy to read. However, it is not well documented and there are still no successful GA implementation in the literature using this framework and thus implementation uncertainties arise;

- **GAlib** is another free C++ library for GA [54] well documented with many different applications in the literature such as in vehicle routing [55], in cutting stock problem [56], in travelling salesmen problem [57], computer networks [58], etc. This framework was developed in the 1990's and its source code was recently maintained. However, an investigation on Web forums about GAlib integration in the most recent compilers suggest that GAlib's libraries are prone to linking errors. Moreover, mailing lists intended for GAlib users to help each other is not active and unresponsive. This can greatly hinder its integration in SCIP.

The inherent difficulties of using the mentioned framework's libraries due to language complexity with steep learning curves, lack of documentation and users support represent majors drawbacks to GA implementation. For this matter, a simple GA program was identified namely *simpleGA* [59]. This program consists in a simple GA program originally written in *C* by Dennis Cormier and Sita Raghavan [60] and adapted to *C++* by John Burkardt in 2014. The whole program is placed in a single file and

every functions are well commented which makes it easy to read and to comprehend. This approach allows a maximum flexibility and adaptability having total control of the program flow.

5.4 Development of Genetic Algorithm Program

SimpleGA is a very simple real-coded GA with few simple features. The code is designed for maximisation problems where the objective function takes positive values only. There is no distinction between the objective value and the fitness of the individual. Regarding GA operators, the program uses proportional selection, elitist model, one point crossover and uniform mutation. The code does not make use of any graphics or even screen output, and should be highly portable between platforms.

From this code foundation a more complex and robust GA was modified and extended to match the problem' structure in the SCIP framework.

5.4.1 Algorithm Scheme

GA starts with an initial set of solutions, each one of them represented by a chromosome[4]. This initial set is known as the initial population. Then, the latter will be reproduced in the generation loop. In this phase, a new population of solutions is created. Firstly, the individuals will be subject to reproduction by using the genetic operators crossover and mutation originating the new individuals, namely offsprings. Secondly, according to a replacement strategy, the population is updated by replacing the previous individuals by the new ones. Thirdly, by using an appropriate selection strategy an individual from one population is picked depending on its fitness and used to form a new offspring. This process is repeated until GA reached the stopping criteria. Figure 5.3 shows the flowchart of a typical GA.

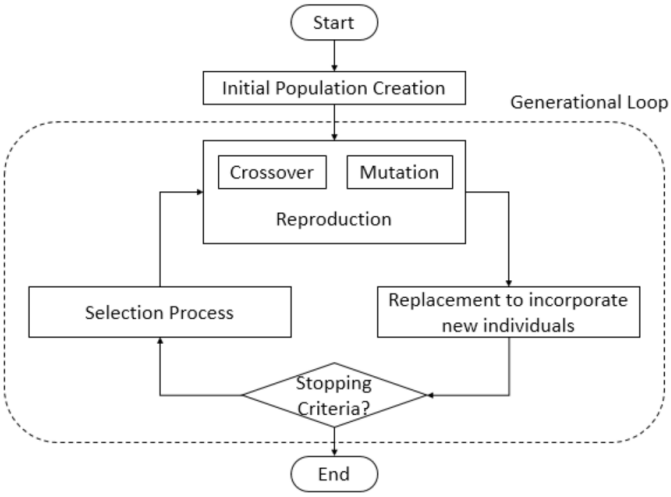


Figure 5.3: Typical GA flowchart

GA iteratively applies the generation of new population and the replacement of a new one, see figure 5.4. This is possible because the history of the previous search, *i.e.* the populations of the previous

[4]The solution represented by a chromosome is called as individual

generations, is stored in a memory which can be used in the generation of the new population and the replacement of the old one.

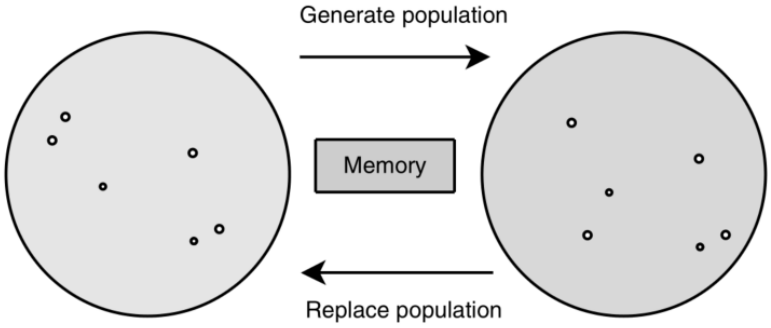


Figure 5.4: GA generation and replacement principle, source [45]

5.4.2 Fitness Landscape Analysis

Before the designing of GA gets under way, it is essential to study the problem's fitness landscape. The effectiveness of a GA will depend on the properties of the problem's landscape associated with the instances to solve, thus it is an important aspect in designing a GA, namely its search components, *i.e.* the solution representation, search operators (selection, crossover and mutation) and the objective function [45]. The goal of this analysis, is to attempt to predict the behaviour of the GA' search components. Furthermore, this study provides a better understanding of the search operators' run over the search space.

Data from the 3rd day of June of 2014 was used to perform this analysis. Conflict costs are not taken into account for this analysis. The figure 5.5 illustrates a plot of all the 315 468 data points which represent all the start-slot options' delay costs per flight. Looking into all delay costs in the same figure, it is possible to notice an increasing of delay costs over the start-slots for every flight. The minimum value of a delay cost is 0 $ which is located in the first start-slot option and the maximum is 114 500 $ which is located in the twelfth start-slot. The table 5.1 shows the mean value and variance of the delay cost across all flights per start-slot. It also shows the minimum and maximum delay cost per start-slot. By inspecting these statistics, it is noticeable that all the delay costs' statistical measures increase with the start-slots options.

	Start-slot Options											
	1	2	3	4	5	6	7	8	9	10	11	12
Mean	0	339.60	1019.37	2161.58	3303.79	4901.88	6499.96	8650.8	10401.7	12871.3	15340.8	19801.6
Variance	0	19542.7	207548	1.028×10^6	2.472×10^6	5.601×10^6	9.990×10^6	1.712×10^7	2.616×10^7	4.047×10^7	5.790×10^7	2.832×10^8
Minimum	0	160	430	845	1260	1825	2390	3070	3750	4597.5	5445	6200
Maximum	0	970	3050	6645	10240	15320	20400	26625	32850	40767.500	48685	114500

Table 5.1: Statistical data per start-slot

Recapping from the previous chapter, the first start-slot has no departure delay associated, the following start-slots have an increasing departure delay of 15 minutes per start-slot (*i.e.* the second start-slot has 15 minutes of departure delay, the third one has 30 minutes, and so on) and the twelfth

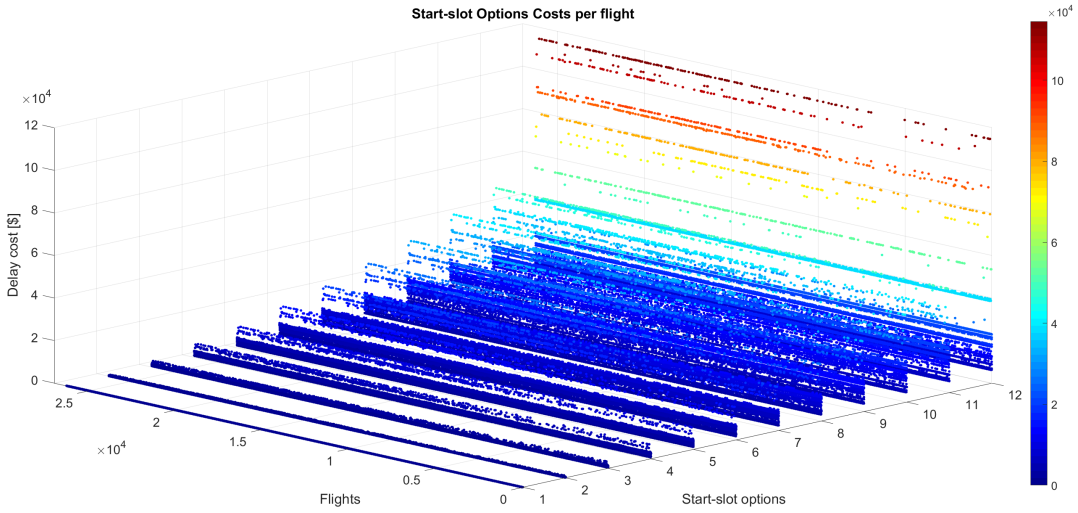


Figure 5.5: Start-slots options cost per flight

start-slot corresponds to the cancellation of a flight's departure. Hence, it is expected that an increasing of the total delay cost happens if start-slots with high departure delays are allocated. The figure 5.6 depicts a sum of the flights delay costs per start-slot in which also with the statistical data in table 5.1 it is possible to confirm the previous thought. The first start-slot has zero cumulative delay cost as no departure delay is associated to it. There is a clear increasing of costs over the next start-slots until the departure cancellation start-slot which has the highest cumulative delay cost of $5.206 \times 10^8\$$.

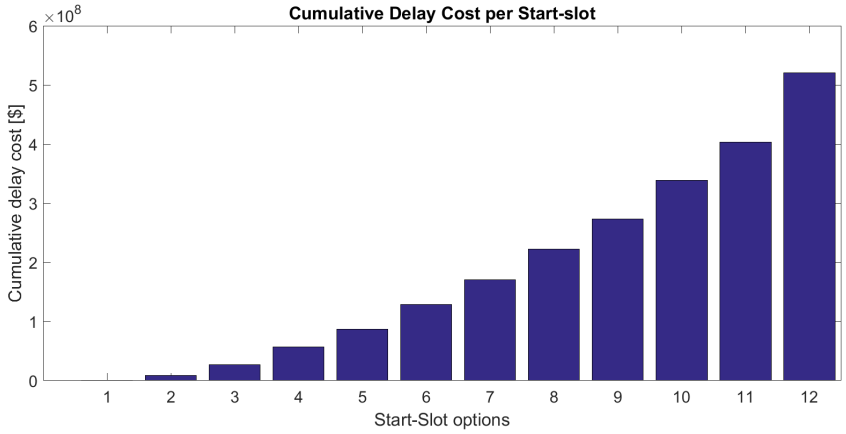


Figure 5.6: Cumulative delay cost per start-slot

Having these informations in mind, the GA design can proceed.

5.4.3 Solution Representation

In GAs, the genotype represents the individual while the phenotype represents the solution. Therefore, the genotype must be decoded to generate the phenotype. The reproduction acts on the genotype level using the genetic operators while the fitness function will use the phenotype of the associated individual to evaluate it and compute its fitness, see figure 5.7.

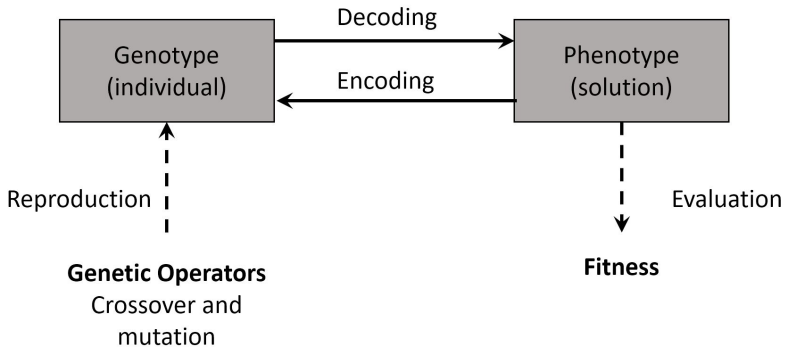


Figure 5.7: Genotype versus phenotype in GAs

Designing any GA needs an encoding scheme, *i.e.* a representation, of a solution. This is a fundamental design question in a development of a GA. The encoding plays a major role in the efficiency and effectiveness of a GA. It has been observed that improper representation can lead to poor performance of the GA. The encoding must be suitable and relevant to the tackled optimisation problem and for this reason it is highly problem problem specific. To define this representation, it must be taken into account how the solution will be evaluated and how the genetic operators will operate the individuals. According to Talbi, a representation must have the following requirements:

- **Completeness:** All solutions associated with problem must be represented;
- **Connexity:** A search path must exist between any two solutions of the search space and any solution in it can be attained;
- **Efficiency:** The representation must be easy to manipulate by the genetic operators. The time and space complexities of the operators dealing with the representation must be reduced.

To comply with these requirements, a linear representation structure was chosen to represent the problem' solution space. The ATFM problem' solution is represented by binary variables, each variable dictates if the specific start-slot option was allocated to a particular flight. The size of the solution is given by the product of the number of flights and the number of start-slot options. There are 12 start-slot options. According to the set partitioning constraint 3.4, only one start-slot option can be assigned to a given flight as the table 5.2 depicts. Thus, in the encoding space, the solution representation uses the following encoding variable, namely *gene*.

$$\varphi(f) = d \qquad \text{if the start-slot } d \text{ is assigned to flight } f \tag{5.1}$$

Each $\varphi(f)$, *i.e.* gene, has a value, designated as *allele*, corresponding to the start-slot option d_k index within the discrete interval $[1, 12]$. In the table 5.2, the flight f_1 was allocated to the third time-slot d_3 forming the non-zero binary variable $x_{1,3}$. By applying the encoding scheme, the non-zero binary variable $x_{1,3}$ is transformed into the gene $\varphi(1)$ which takes the allele value 3. The same transformation is applied for every non-zero binary variables of each problem solution in the solution space or phenotype resulting in one chromosome or individual flight's vector in the encoding space or genotype depicted

in the table 5.3. The phenotype is composed by all possible problem' solutions represented by binary variables and the genotype is composed by all possible individuals represented by integer variables that can span from 1 to 12.

		Start-slot options											
		d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}
Flights	f_1	0	0	1	0	0	0	0	0	0	0	0	0
	f_2	0	0	0	0	0	0	0	0	0	0	0	1
	.						.						
	.						.						
	.						.						
	f_n	1	0	0	0	0	0	0	0	0	0	0	0

Table 5.2: Problem' solution variables

		Alleles = d_k
		$\varphi(f)$
Genes = Flights	$\varphi(1)$	3
	$\varphi(2)$	12
	.	.
	.	.
	.	.
	$\varphi(n)$	1

Table 5.3: Flight's vector

Using this encoding, all the three requirements are satisfied. This transformation ensures that the start constraint is always satisfied for every flight. Moreover, this representation reduces the solution vector by ten times resulting in an encoded vector of size equal to number of flights in the ATFM problem. Consequently, this reduces the original search space which GA has to explore in the same scale improving GA's efficient.

The mapping between the phenotype and the genotype is done in an indirect fashion, *i.e.* the encoding is not actually a complete solution for the problem [45], it only represents the non-zero binary variables. For this reason, the encoded solution cannot be evaluated by the objective function and be used to validate the solution against the capacity constraints within the optimisation problem. The objective function remains the same. Therefore, a decoder must be specified to express the solution given by the encoding.

To implement a decoder, the allele of the gene must be read to get the index of the non-zero binary variable. With this information, it is possible to allocate a non-zero binary variable into the problem' solution. Take this example for instance: in the table 5.3, the gene $\varphi(2)$ has the allele 12 which indicates that the non-zero binary variable $x_{2,12}$ must be allocated in the row 2, column 12 in the problem' solution variables. Therefore, the flight f_2 was allocated with a cancelled flight departure slot. The same decoding scheme is applied for all the other flights resulting in the problem' solution which can be evaluated by the optimisation problem's objective solution and validated against its capacity constraints.

In conclusion, choosing a proper representation, having a proper definition of the mappings between the phenotype and genotype spaces is essential for the success of a GA.

5.4.4 Initial Population

GA starts from an initial population of solutions before starting its iterative process. This step plays a crucial role in the effectiveness of the algorithm and its efficiency.

Two main strategies can be used to generate the initial solution: random and greedy strategies. The

former draws random alleles to the genes and is known to create diverse and not so fit individuals and the latter generates fit individuals from just one individual with low computation costs but compromises the population diversity. A trade-off between the quality and the diversity of solutions, and the computational time must be taken into account according to the criteria [45].

In the generation of the initial population, the most important criterion is diversification. If the initial population is not well diversified, a premature convergence might occur [45, 61]. Therefore, strategies that promote population's diversity are favoured. For this algorithm, a method of random generation of individual was implemented. This method makes use of two main features:

- **Death penalty function:** only the feasible individuals will take part of the initial population;
- **Problem-specific knowledge:** Due to the fact that this problem is highly constrained, it is very hard to find feasible solutions using a random search method. Thus, there is the need to introduce some problem's knowledge in this method. The more flights in network the more en-route interconnections there will be responsible for toughen the solution's feasibility. Hence, by cancelling some flights it is possible to reduce the number of en-route flights consequently reducing the interconnectivity and easing the search for feasible solutions.

So, an initial population comprised with individuals constituted by only cancelled flights, *i.e.* all their alleles are set to the last start-slot option d_{12} forming identical vectors with all their elements' value set to 12, is to be randomly generated. This method draws alleles randomly ensuring that each allele has the same probability to be drawn. The number of genes to be drawn is also random ensuring a considerable amount of flights cancelled per individual. By analysing the result of 10 runs in the table 5.4 it is observable a great reduction in the computational time of the population initialisation. However, if we look into the fitness values results, these individuals present bad fitness values due to high number of cancelled flights which increased from 220 003 to 1 636 265.

	$t(s)$	Best Fit	Worst Fit	Mean Fit	#Cancelled
Without problem specific knowledge	612.3	1.83×10^8	1.88×10^8	1.86×10^8	220003
With problem specific knowledge	29.9	1.98×10^8	5.16×10^8	3.82×10^8	1 636 265

Table 5.4: Random search results with and without problem-specific knowledge

5.4.5 Selection Strategy

As the size of the population is constant, it allows to withdraw individuals according to a given selection strategy. The latter concerns the parents selection for the next generation with a bias towards better fitness.

The main principle of selection is “the fitter is an individual, the higher is its chance of being parent.” This principle inflicts a selection pressure on the individuals which is responsible to drive the population to better solutions. However, worst individuals, *i.e.* unfit individuals, should not be discarded because they may have useful genetic material. Therefore, they have some chance to be selected. To determine such an individual’s ranking from the best to the worst it is necessary to accomplish a fitness assignment upon the candidates individuals. Two different fitness assignment to the candidates individuals approaches are studied in this work:

- **Proportional:** in which absolute fitnesses are assigned;
- **Rank-based:** in which relative fitnesses are assigned.

After the individual’s ranking is done, they are selected according to their fitness by means of a chosen selection strategy. Two different strategies are applied for each fitness assignment approach respectively.

Roulette Wheel Selection

This strategy assigns to each individual a selection probability that is proportional to its relative fitness. Being f_i the fitness of the individual P_i in the population Θ its probability to be select is:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_i} \tag{5.2}$$

To better understand this strategy, one can visualise a pie graph in the figure 5.8 where each individual is assigned a pie’ slice on the graph which corresponds to its fitness. Around the pie, an outer roulette wheel is set. The selection of μ individuals is performed by μ independent roulette wheel’ spins. Each spin will select just one individual. Fitter individuals have bigger slices, thus more chances to be chosen.

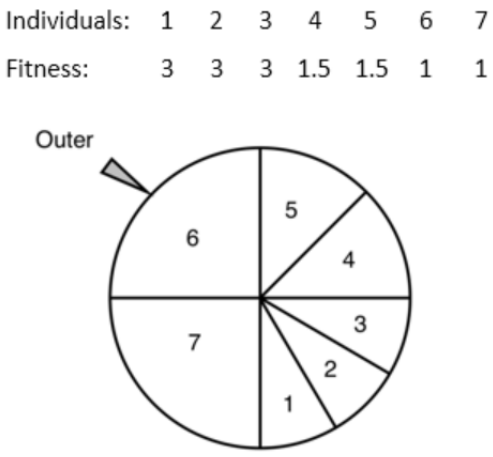


Figure 5.8: Roulette selection strategy. Each spin selects a single individual. The actual problem is to be minimised and thus individuals with lower fitness values are actually fitter. Adapted from source [45].

Tournament Selection

This strategy consists in selecting k individuals randomly. The parameter k dictates the number of contestants in the tournament. A tournament is then applied to the k members of the group to select the best one as the figure 5.9 illustrates. To select μ individuals, the tournament procedure is performed μ times.

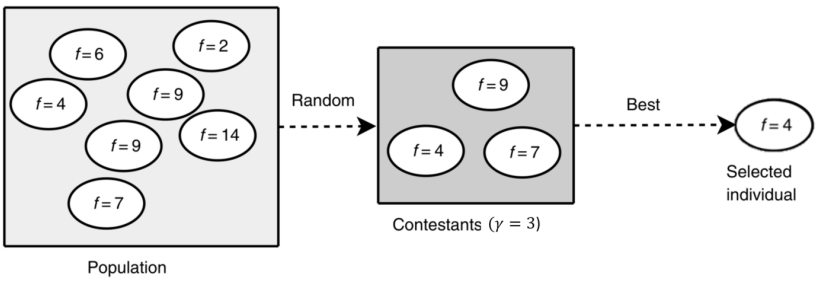


Figure 5.9: Tournament selection strategy. In this example, a tournament of size $\gamma = 3$ is carry out. Three solutions are selected randomly from the population. The best individual is then selected. Adapted from source [45]

5.4.6 Genetic Operators

Once the selection of individuals to form the parents population is performed, the reproduction phase takes place with the application of genetic operators such as the crossover and mutation.

Crossover

The function of crossover is to interchange some genetic material, *i.e.* characteristics, of the two parents to generate offspring. Its design mainly depends on the representation (encoding) used. According to Talbi, when designing this genetic operator two aspects must be taken into account:

- **Heritability:** the crossover operator should inherit genetic material from both parents. If two identical individuals generate identical offspring the crossover is a pure recombination operator, *i.e.* it has a strong heritability;

- **Validity:** the crossover operator should produce valid solutions. This aspect is very difficult to fulfil because this optimisation problem is highly constrained shrinking the whole solution space to a feasible subset.

Moreover, the performance of this operator largely depends on its user-defined parameter, the crossover rate p_c that spans from 0 to 1. This parameter represents the proportion of parents on which the crossover will perform. In the literature, there are several proposed crossover operators. In this work, some of them are studied and implemented such as the n-point crossover and uniform crossover.

The n-point crossover

This is the generalised form of a group of different crossover operators, each one differentiating by the number of crossing points which originates segments of the chromosome[5] ready to be interchanged among the parents' chromosomes.

In the 1-point crossover, a crossover point κ is computed randomly which spans the chromosome's length. Then, two segments per chromosome are formed separated in the κth position and thereafter interchanged them resulting in two offspring as the figure 5.10 depicts.

In the 2-point crossover, two crossover points are computed randomly and then following the same method as the previous operator, they interchange genetic material within the two points as the figure 5.10 depicts.

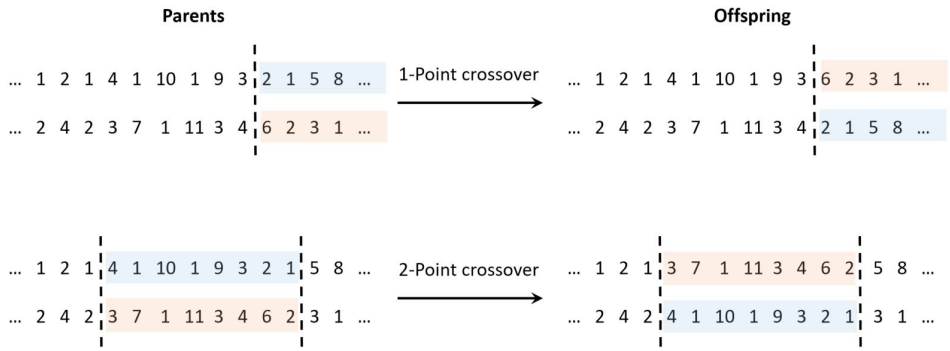


Figure 5.10: n-Point crossover operator. In the upper part of image it is illustrated the 1-point crossover and in the bottom part of the figure it is illustrated the 2-point crossover.

The uniform crossover

In the uniform crossover, two individuals can be recombined without taking into account the size of segments. Each element of the offspring is selected randomly from either parent. Each parent will contribute equally to generate the offspring as the figure 5.11 shows.

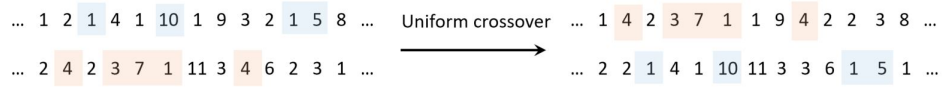


Figure 5.11: The uniform crossover operator.

Mutation

The function of mutation is to perform independently small changes in selected individuals of the population in order to introduce in the offspring's some new features (hopefully desirable features) which are not possessed by its parents. These small changes in the genetic material of the offspring, maintain and introduce diversity in the genetic population. Moreover, it allows the GA to explore a new, maybe better

[5]Or individual

part of the feasible region than the ones previously considered and consequently escaping from local minima.

According to the literature, mutation should be performed in a low rate. This rate is controlled by the user-defined parameter p_m which defines the probability to mutate each gene of the chromosome. Thus, small values are recommended for this parameter. According to Talbi, when designing the mutation operator three aspects must be taken into account:

- **Ergodicity:** it should allow every solution of the search space to be reached;
- **Validity:** it should produce valid solutions. This aspect is very difficult to fulfil because this optimisation problem is highly constrained shrinking the whole solution space to a feasible subset;
- **Locality:** is the effect on phenotype when performing a change in the genotype [45]. The mutation operator should have a strong locality, *i.e.* it should generate minimal changes in the genotype and consequently reveal small changes in the phenotype. Its impact is crucial and should be controllable. If a small change in the genotype produces a great perturbation in the phenotype the mutation is said to have a weak locality and possibly making the search to converge toward a random search in the landscape.

For this problem, a mutation in the discrete representation is to be implemented. In the literature, there are several proposed mutation operators for integer representation. According to [61] the main forms of mutation for integer representation are the random resetting mutation and the creep mutation. One more form of mutation was designed which was obtained by extending the random resetting one, namely bias resetting mutation.

Random resetting mutation

In the random resetting mutation, a random allele from the set of permissible values Ω is assigned to a randomly chosen gene. The set Ω comprises the starting-slots and thus $\Omega \in [1; 12]$. All the genes alleles x are equally likely to be chosen with the probability $P(d_k) = \frac{1}{12}$.

Bias resetting mutation

A new operator was designed by extending the random resetting one. Instead of setting a equal probability to all alleles, it is possible to set a distribution from which the random numbers are drawn. The goal of using these modelled distribution is to control the likelihood of the numbers' drawing in such a way that the first start-slot options are more likely to be drawn than the last ones avoiding allocations of start-slots with high delay. For that purpose, the following distribution was modelled.

Half-normal distribution

The half-normal distribution is a special case of the folded normal distribution by setting the mean to zero. The equation 5.3 depicts its probability distribution function (PDF).

$$f_X(x) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \quad x \geq 0 \tag{5.3}$$

Where the mean is $\mu = 0$ and the scalar parameter σ is to be tuned according to the mentioned preference. The figure 5.12 shows the probability functions and the cumulative functions for the start-slot options. With this distribution, it is more likely to delay less the flights and consequently reducing the delay costs.

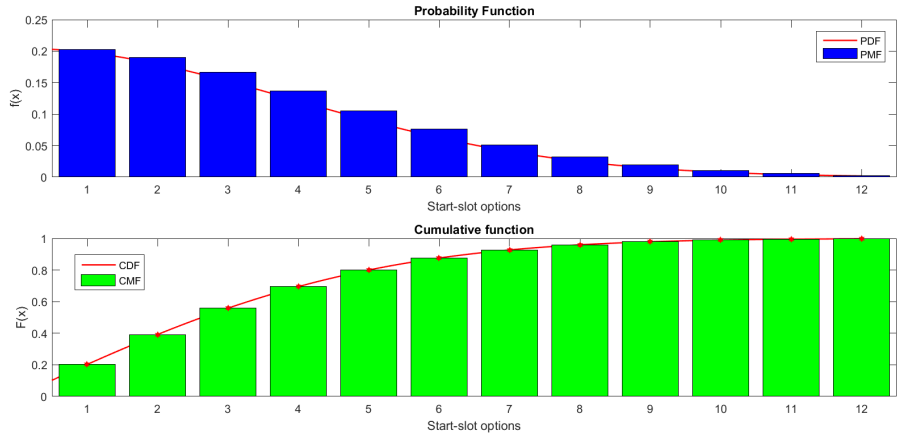


Figure 5.12: Half-Normal distributions.

Creep mutation

This operator scheme was designed for representations with ordinal attributes and therefore fits in the encoding of this problem. Its function is to add small positive or negative values to each gene with a given probability p_i. From a distribution that is symmetric about zero, the probabilities p_i associated with the values to be added are sampled randomly for each position. Thereby, is more likely to generate small changes than large ones.

To design this operator a distribution must be chosen and its parameters must be tuned, hence controlling the distribution from which the random alleles are drawn. For this purpose, the well known normal distribution is the one to be used. The figure shows the normal distribution used in this operator with mean $\mu = 0$ and standard deviation $\sigma = 2.7$.

5.4.7 Replacement Strategy

The last step in GA consists in selecting the new solutions from the union of the current population and the generated one. A replacement strategy must be specified according the survival of the fittest natural phenomena. Therefore, this phase concerns the survivor selection which updates the population with the new offspring which compete with old individuals for their place in the next generation. The strategy implemented was the traditional generational replacement proposed by J.H.Holland. This strategy replaces the whole population of size μ. The offspring population will replace systematically the parent population.

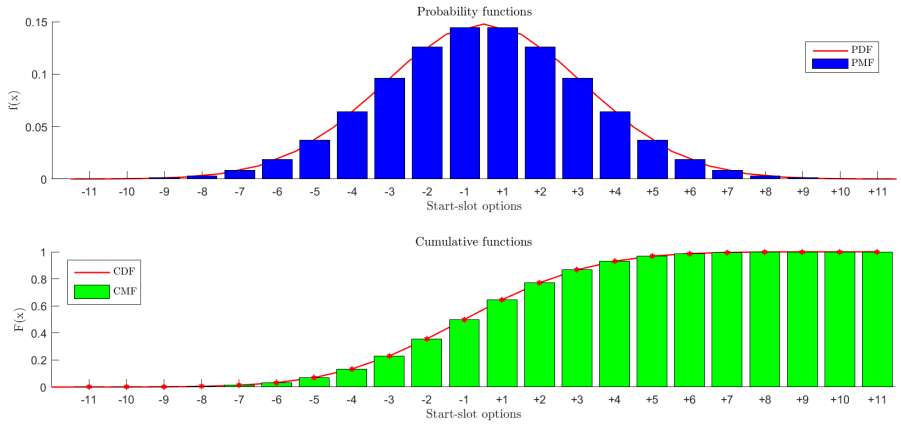


Figure 5.13: Normal distributions.

Other strategy is used namely elitism, in which best individuals from the parents and offsprings population are chosen to take part of the next generation as it is shown in the figure 5.14. This approach provides a faster convergence. Since this approach applies selection pressure, care should be take because it could lead to a premature convergence and consequently to be trapped in a local minima if high elitism pressure is applied.

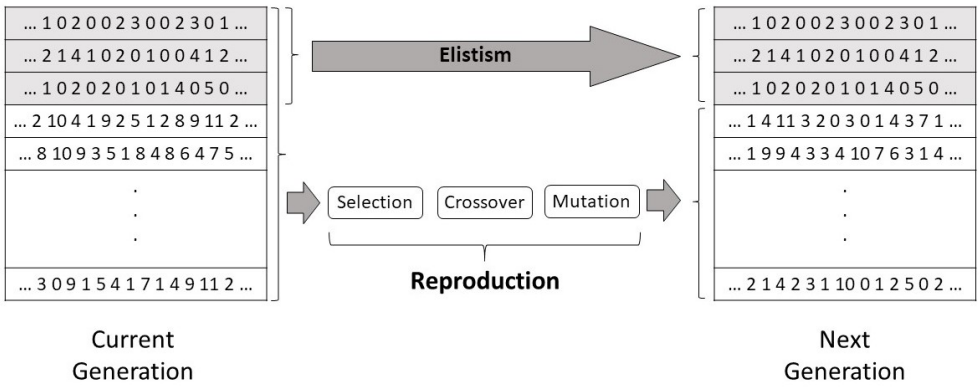


Figure 5.14: Replacement strategy - Elitism.

5.4.8 Stopping Criterion

Finally, a suitable criterion to stop the algorithm is to be implemented. GAs are stochastic algorithms and mostly there are no guarantees to reach an optimum, hence they may never stop searching for the best solution. Moreover, one of the problems has an unknown optimal objective value.

To ensure GA's termination, a static procedure was chosen. In such a procedure, the algorithm stops its search certainly and the end of the search is defined *a priori*. A variety of this type of stopping criteria spans in the literature such as: the population diversity drops under a given threshold, maximum time elapsed by the CPU, maximum number of generations and so on. For this algorithm, a maximum number of generation condition was implemented as the GA stopping criterion. Therefore, when a given

number of generation is completed, the GA stops its iterative process and returns the best solutions found and its iteration' statistics.

5.5 Genetic Algorithm Integration in SCIP

In this section, the integration in SCIP of a GA, namely *atfmGA*, using the problem' structure will be explained.

5.5.1 Genetic Algorithm as a SCIP Module

Thanks to SCIP's user defined *plug-in* approach, the integration of new code is facilitated. To start implementing the GA in SCIP, its algorithm was coded inside the execution method of the primal heuristic *plug-in*. The latter, executes *atfmGA* functions which are used in the execution method, within the SCIP' solution process every time it is called.

Then, it is necessary to create the SCIP heuristic, designated as *heur_atfmGA*, and include it in SCIP's data structure object so that SCIP can use the callback functions to execute the *heur_atfmGA*. Figure 5.15 shows a scheme of all interfaces used. The NFE data which comprises the flights and the network elements' data can be found in Matlab data files. In *scipmex* interface, the NFE data is used as input. In order to create the problem data structure and consequently the ATFM problem in SCIP such as the variables, constraints and the objective function. The main function of this interface is executed when the *mex* file *AllocateSlotsNFE* is called in Matlab after being compiled by the *mex* compiler explained in section B.2. It is crucial to include the source code and the header file of *atfmGA* in order to compile the former. Inside SCIP, *heur_atfmGA* runs the algorithm according to the primal heuristic properties with a population data structure defined in *atfmGA* by making use of its functions, *i.e.* the GA operators described in the previous sections. It is possible to tune the *atfmGA* parameters such as the population size, probability of mutation, crossover and so on in *atfmGA*'s header file. After the solution process is completed, *scipmex* writes the solver' statistics and returns the best solution found in form of a vector of start-slots along with its objective value.

The user has total control of the SCIP' solution process by choosing the combinations of properties' settings of all modules being used which SCIP offers to match his/her necessities. Thereafter, it is possible to control when and at what frequencies the *atfmGA* is called in the solution process.

5.5.2 Population Data Structure and Solution Flow

As explained in the subsection 5.4.3 there are two different solution representations, the genotype of flights and the phenotype of variables. In order to handle these two different representation, two data structure are used in this algorithm. In the genotype, a data structure namely *population*, is used. It comprises vectors of size *nFLIGHTS* named *individuals*, and a variable which represents its fitness named *fitness*. The *individuals* vector elements consist of the delay for each flight. In the phenotype, the solutions are stored in the so called SCIP solution storage in which only the non repeated feasible

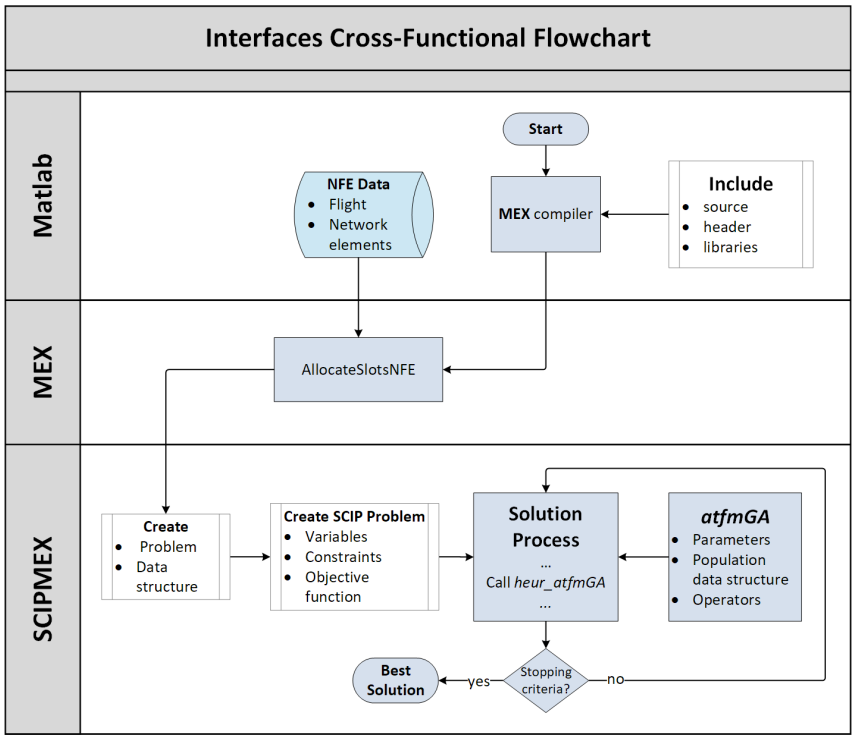


Figure 5.15: Interfaces cross-functional flowchart modules

solutions are stored within a fitness ranking list, *i.e.* a list which sorts the unique feasible solutions from the fittest to the most unfit. There are three different denominations for the population structure depending in which stage they are depicted in figure 5.16. In the first stage of GA, an initial population is created. During its creation, the individuals are decoded and checked for feasibility. The feasible solutions decoded from the feasible individuals are added to the SCIP solution storage. When a certain number of feasible solutions is reached, the initial population gets updated with encoded individuals from the SCIP solution storage. This way, only feasible individuals take part of the initial population ready to enter the generational loop.

In the generational loop, the initial population becomes the parents of the first generation. After the reproduction stage, new individuals are ”born” by genetically modifying the population of parents and consequently becoming the children population. The children feasibility is checked and the ones which are feasible are the ones that are going to take part of the SCIP solution storage if and only if their objective value are better than the objective values of the last element in the storage.

Then, follows the selection stage and a new parents population is created replacing completely the previous parents population. Due to the elitism scheme, the top solutions in the SCIP solution storage are encoded to take part of the new parents population in every generation.

5.5.3 GA Flowchart

The figure 5.17 illustrates the flowchart of the proposed GA. The program flow of GA is divided into two main blocks: *Initial Population Creation* in which the initial population is created and the *Generational Loop* in which the individual are created and the solutions tried. The latter is subdivided in four modules,

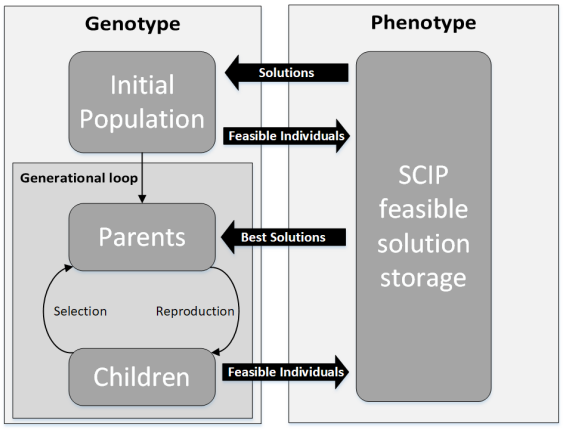


Figure 5.16: Population data structure and solution data structure flow

the *Selection*, the *Reproduction*, the *Evaluation* and the *Replacement*. The encoding of the solutions and the decoding of the individuals is accomplished by the functions `CreateIndividualFromSol()` and `CreateSolFromIndividual()` respectively.

In the *Initial Population* block, the function `CreateRandomIndividual()` applies the random resetting mutation to generate random alleles in the individuals genes. Then, `CreateSolFromIndividual()` decodes the individual to the solution space to make it possible to evaluate them using the SCIP function `SCIPtrySol()`. If the solution is feasible, it is stored in the SCIP solution storage incrementing the number of feasible solutions `nSol`, otherwise it is deleted. This process is repeated until the number of feasible solutions is equal to the size of the population. After the stopping criteria is reached, the feasible solutions are encoded using the function `CreateIndividualFromSol()` and thus creating the initial population. To speed-up this process, a considerable number of start-slot d_{12} is drawn and consequently cancelling flights. By doing this, it is easier to find feasible solutions since less flights are entering in the network. However, the solutions have an high cost.

It follows the *Generational Loop* block in which the initial population becomes the parents of the first generation. In the *Selection* sub-block, the function `Tournament()` selects the children with a selection pressure towards to the fittest individuals to become the parents of the current generation. Then, the parents will mate in the *Reproduction* sub-block. Firstly, the `Crossover` unites pairs of parents which will mate, originating two children per pair. Secondly, the resulting children will then be mutated in some of their genes using `Mutation()` function. In order to evaluate the resulting children, they must be decoded into the solution space. Thereafter, the SCIP function `SCIPsolGetOrigObj()` evaluates each children assigning to each one of them the respective fitness value. Then, their feasibility is checked and if they are feasible they will be stored in the SCIP solution storage incrementing `nSols`, otherwise they are deleted.

Finally, the function `Elitism()` partially replaces the children population by the top individuals found so far encoded from the solutions *sols* sorted in the SCIP solution storage. This loop continues until the stopping criterion is reached, *i.e.* the loop will break when the number of generation reaches its maximum allowed, `MAXGEN`. Ended the search, it returns the best solutions found and its iterative statistics.

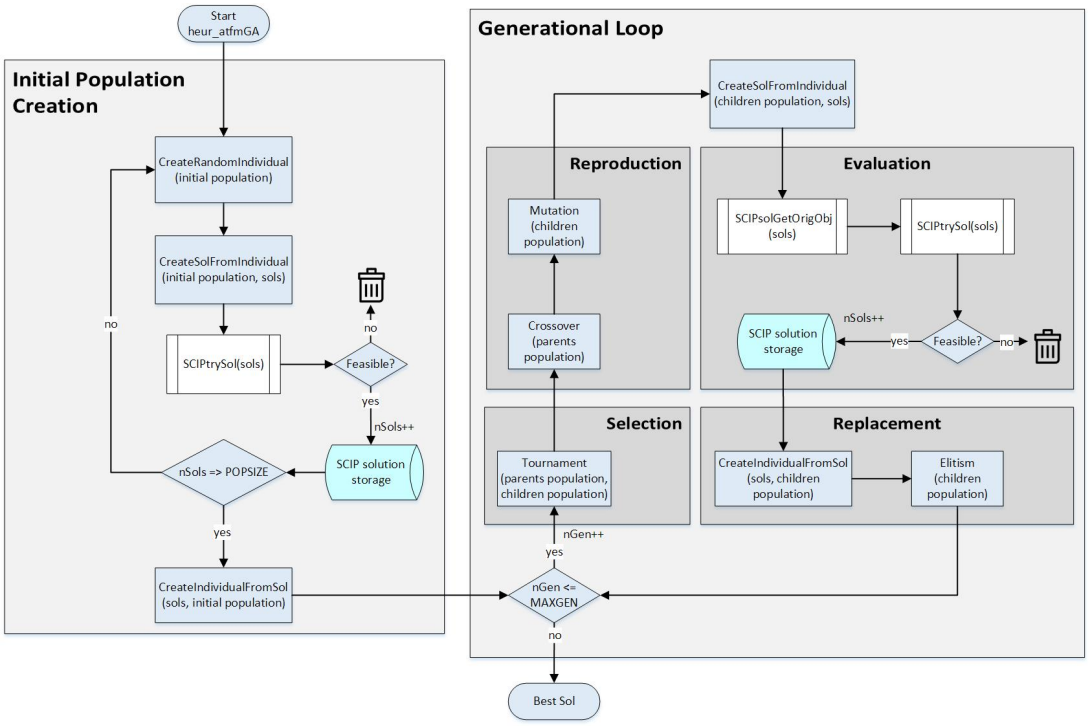


Figure 5.17: *atfmGA* flowchart

5.6 Discussion

To develop this algorithm in an efficient and timely manner, a GA framework was proposed to be used. Firstly an investigation on the available frameworks was carried out. Unfortunately, many of those frameworks are outdated and abandoned by their own developers. Due to time constraints and interface limitations, a decision of reuse code of a simple GA was made in section 5.3. From that base code, a more robust GA was to be developed.

After the development of the *atfmGA* throughout the sections of this chapter, a GA was integrated in SCIP ready to be used at any level of the branch-and-bound tree search. Due to the immense and complex search space of the EATFM problem, it was ”impossible in any ways” to produce a visual representation of it. Although, it was possible to represent its fitness landscape in the 5.4.2. To implement the *atfmGA* components such as the bias resetting and the generation of the initial population, the fitness landscape knowledge was taken into account to improve the solutions.

The algorithm components to be tested in the next chapter are summarised in the table 5.5.

	Solution Representation	**Initial Population**	**Selection Strategy**	**Crossover**	**Mutation**	**Replacement Strategy**	**Stopping Criterion**
Features	Integer	Random with problem specific knowledge	Roulette wheel Tournament	1-Point (1-P) 2-Point (2-P) Uniform (UX)	Random Resetting (RR) Bias Resetting (BR) Creep (CM)	Elitism	Number of iterations
Parameters	-	Population size (`PopSize`)	Tournament size (`TournSize`)	Probability of crossover (`PXover`)	Probability of mutation (`PMutation`)	Elitism ratio (`ElitRatio`)	Maximum number of generations (`MaxGen`)

Table 5.5: *atfmGA* summary of the features and respective parameters

Chapter 6

Results

The performance of the developed GA, namely *atfmGA*, inside the SCIP module `heur_atfmGA` will be assessed for different scenarios and instances which are described in section 6.1. It follows the *atfmGA* performance assessment in section 6.3. Firstly, experiments of *atfmGA* operators combinations designed in section 5.4 are evaluated against each experiment. Secondly, a sensitive analysis is performed on the chosen operators parameters. The last two sections are devoted to the optimal and stochastic solving results of the ATFM problem with and without conflicts. The latter, is sub-divided in 4 different instances. Due to the fact that SCIP is not able to find Upper Bounds (UBs) for the large scale ATFM problem with conflicts, *atfmGA* is to be used to provide UBs and thus be able to compute an optimality gap which measures how far the UB found by *atfmGA* is from the Lower Bound (LB) found by SCIP. The huge search space size of the problem combined with the stochastic nature of GA search method strongly complicates the search for feasible solutions. Further adjustments on the *atfmGA* namely, the implementation of an on-line dynamic parameter update scheme and a fitness penalty function, was accomplished to try to overcome obstacles found. Finally, the results of a novel GA integration in SCIP are presented.

6.1 Solved Scenarios: Conflict-free and Conflicted

To carry out the performance assessment of the solver, two main scenarios are used, the conflict-free scenario and the conflicted scenario. The first one will be used to evaluate the performance of the search operators designed in the previous chapter. The second is subdivided into 4 different instances and it will be solved with the most suitable search operators and the best set of parameter values determined for the first scenario.

For both scenarios, the corresponding date is 3-06/2014 (see Chapter 3). All flights that should departure on this day are considered. The number of flights to be regulated is 26 289. A total of 107 time intervals are included, which corresponds to a time of 26h45min. The 107 time intervals again indicate the period from the earliest start time of a flight to the latest (planned) landing time plus the maximum possible delay. The number of considered sectors amounts to 617. The number of considered airports

is 1 154. This results in a total of 312 975 capacity restrictions for the sectors and airports for both arrival and departure time intervals. Figure 6.1 shows an example of trajectories in one full day.

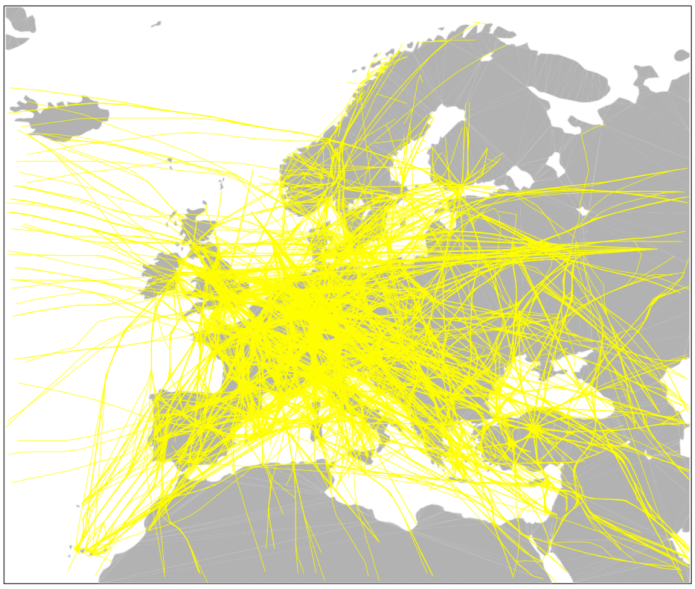


Figure 6.1: Example of a representation of the timetable route network for a full day scenario (NFE Output), source [63]

6.2 GA Start-Slots Allocation Mechanics

To make the *atfmGA* start-slot allocation mechanics clear to the reader, a visualisation of how the allocated start-slots are to be distributed throughout the generations is presented. The figure 6.2 depicts three distinct solutions found by *atfmGA* in three different moments of the search.

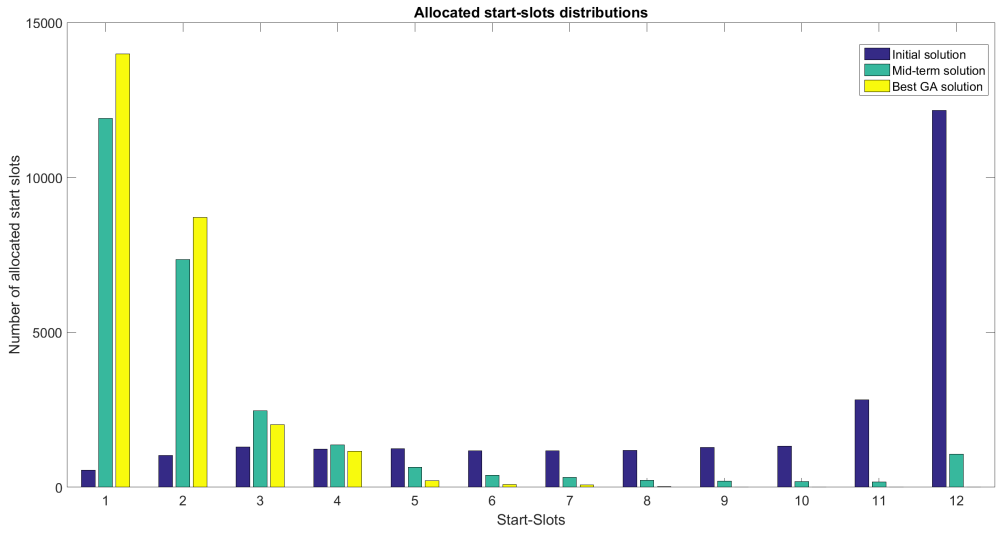


Figure 6.2: Allocated start-slots distributions for three different solutions found by *atfmGA* search

The blue one represents a solution found in the creation of the initial population resulted from the

random slots allocation in which lots of flights are cancelled and hence, the solution has high delay cost. After a certain number of generations, in the search mid-term, the biased stochastic search method which *atfmGA* employs, allocates more start-slots with lower delay costs and consequently the distribution is shifted to the left as it can be seen in the picture by comparing the blue distribution and the green one. In the end of the search, the best solution presents a distribution even more shifted to left characterized by a majority number of slots allocated in the first start-slots and thus, the solution has lower delay cost than the previous two.

6.3 GA Performance Assessment

The performance of GA depends on many factors[1] associated with the selection strategy, the genetic operators, the replacement strategy and the generation of the initial population. Moreover, it also depends of their parameter values. Hence, adapting the suitable *atfmGA* strategy and components along with the most suitable parameter values is very important because it guides the search process to better results, *i.e.* it improves the quality of the solutions. This can be accomplished by executing experiments using different components and parameters. Then, statistical analysis will be applied to evaluate the obtained results. For all the different experiments, 10 computational tests per experiment are to be carry out to derive statistical results. The selected instance is the full day scenario without conflicts described in 6.1. Also, every best solution found throughout the experiments, its UB will be recorded and the optimality gap updated relatively to the problem known optimal solution $\mathbf{7.595 \times 10^3}$ \$. This gap, will be computed with the obtained *atfmGA* UB and the known LB which is the optimal solution. The optimality gap or just simply gap, is given by the equation 6.1.

$$Gap = \frac{UB - LB}{min(UB, LB)} \times 100\% \tag{6.1}$$

To run all the proposed experiments, the availability of 5 different computers in the computers room in "Laboratórios de Controlo, Automação e Informática Industrial" of "Mecânica III" plus the personal computer of the author were taken advantage. The computational tests were ran on Windows 10 operating system with Intel 3.30Ghz processors and 8 GBs of installed RAM memory. In two of the computers, extra 8GBs were installed to be able to run SCIP solver for the ATFM problem with 10% of minimum probability of conflict.

6.3.1 GA Design Performance

To evaluate the performance of the designed GA components, several computational tests were conducted to assess which combination of GA components among the search operators, could produce the best results. To assess their performance, the Best Fitness (BF) and the Mean of the Best Fitness (MBF) of the individuals are used. The latter, represents the average of the best individuals found each

[1] Excluding the computer and software features in which the GA is tested.

trial for one experiment. For these experiments, the *atfmGA* parameters described in Chapter 5 were fixed to the following values:

- `MaxGen` = 100;
- `PXover` = 0.9;
- `PMutation` = 0.1;
- `PopSize` = 100;
- `ElitRatio` = 0.1.

The goal of the following experiments is to find the best combination of GA operators (selection, crossover and mutation) evaluating each combination by MBF and BF as performance indicators. Regarding the decision criterion, MBF has more weight in the final decision than the BF. The results can be seen in the tables 6.1 and 6.2.

			Crossover		
		$\times 10^7$	1-P	2-P	UX
Mutation	RR	**BF**	12.371	11.924	10.271
		MBF	17.120	17.101	17.690
	BR	**BF**	3.314	**3.216**	3.229
		MBF	6.048	**5.239**	5.297
	CM	**BF**	7.315	8.121	5.215
		MBF	10.210	10.129	10.011
atfmGA BF			SCIP best solution		Gap
$\mathbf{3.216 \times 10^7}$ $			$\mathbf{7.595 \times 10^3}$ $		**423 336%**

Table 6.1: Performance of different experiments of GA components using Roulette Wheel selection strategy. RR:Random Resetting, BR: Bias Resetting, CM: Creep Mutation. 1-P: One point, 2-P: Two point, UX: Uniform. Total experiments computational time: 38.4 hours

Analysing the tables, it is noticeable that the operator Bias Resetting outperforms the others mutation operators. The combination Roulette Wheel selection (RR), Bias Resetting (BR) mutation and Two-Point (2-P) crossover was the one which produced the best MBF $\mathbf{5.239 \times 10^7}$. Although, the combination Tournament selection with 4 contestants (`TournSize`=4), BR mutation and 1-Point (1-P) crossover produced the best result $\mathbf{3.157 \times 10^7}$ its MBF is considerable worse than the first combination mentioned. So far, the last mentioned result is the best solution found so far by *atfmGA* with a optimality gap (or just gap) of **415 678%**. Therefore, it was concluded that the best combination of search operators is Roulette Wheel selection combined with Bias Resetting mutation and Two-Point crossover.

			Crossover								
			TournSize = 2			TournSize = 3			TournSize = 4		
		$\times 10^7$	1-P	2-P	UX	1-P	2-P	UX	1-P	2-P	UX
Mutation	RR	**BF**	10.224	11.991	10.290	12.924	10.218	10.221	11.219	11.231	11.321
		MBF	13.227	14.952	14.152	16.234	16.291	15.129	15.213	15.210	14.188
	BR	**BF**	5.413	4.219	4.211	4.621	3.371	4.951	**3.157**	4.326	6.951
		MBF	7.321	7.223	5.941	6.552	8.106	7.703	**7.258**	8.213	10.703
	CM	**BF**	7.011	6.438	8.318	8.121	7.438	9.318	7.319	3.438	8.521
		MBF	10.121	9.213	13.241	10.311	9.123	11.310	11.311	8.216	12.821
		atfmGA BF: $\mathbf{3.157 \times 10^7}$ $			SCIP UB: $\mathbf{7.593 \times 10^3}$ $			Optimality gap: **415 678%**			

Table 6.2: Performance of different experiments of GA components using Tournament selection strategy. RR:Random Resetting, BR: Bias Resetting, CM: Creep Mutation. 1-P: One point, 2-P: Two point, UX: Uniform. Total experiments computational time: 112.5 hours

6.3.2 Sensitivity Analysis

After the exhaustive performance assessment of the GA components, there is the need to ”tune” their respective parameters. The latter may have a considerable influence on the efficiency and effectiveness of the search. Moreover, it is not obvious to define *a priori* which parameter setting should be use due to the random nature of this type of metaheuristic. They are problem-specific regarding its structure and instance to deal with. Thus, unfortunately, an universally optimal parameter values set for a given GA does not exist [45].

To ”tune” the GA parameters, an *off-line* sensitivity analysis on the parameters `Pmutation`, `PXover` and `PopSize` was performed in order to ”feel” which of their values produce the best results. A sequential one-by-one parameter ”tuning” is to be done however, it does not guarantee to find the optimal setting. To overcome this drawback, a huge number of experiments needs to be done in order to cover different combination values for all the parameters. Furthermore, due to the ”stochasticity” of GA, several trials per experiment need to be executed to validate their performance. To perform such a procedure, a great deal of time and computational cost is needed and consequently, this procedure will not be carried out in this work due to time constraints. Instead, the parameters `Pmutation` and `PXover` are tested in a combinatorial way however, `PopSize` is tested at a time and their optimal values are determined empirically.

Probability of Crossover Vs Probability of Mutation

These two parameters are responsible for how many alleles should be drawn (see 5.4.3), *i.e.* how many start-slots will be allocated. The bigger the probability of crossover (`PXover`) is, the bigger the number of individuals will mate and thus, a greater amount of flight vectors will be interchanged and vice-versa. The bigger the probability of mutation `Pmutation` is, the bigger the number of individual’s genes will

be mutated, *i.e.* a greater amount of new start-slots will be allocated into flights. For the crossover and mutation parameters, the same performance assessment was applied and the same evaluation parameters BF and MBF are used. Furthermore, an average of the BF found in each set of experiments per parameter value was computed. The goal of these experiments is to find the best combination of `Pmutation` and `PXover` parameter values using the same decision criterion in 6.3.1. The table 6.3 depicts the results obtained.

			`PXover`					**Average**
		$\times 10^7$	0.6	0.7	0.8	0.9	0.99	`PMutation` **BF**
`PMutation`	0.2	**BF**	6.214	6.001	6.219	6.012	5.299	5.949×10^7
		MBF	7.921	6.679	7.569	7.918	6.473	-
	0.15	**BF**	6.012	7.421	5.243	4.921	4.913	5.702×10^7
		MBF	6.712	8.386	7.783	7.735	5.853	-
	0.1	**BF**	5.141	3.215	7.341	6.097	4.837	5.326×10^7
		MBF	6.727	5.685	8.465	7.197	7.537	-
	0.05	**BF**	3.378	3.490	3.571	3.251	**3.180**	$\mathbf{3.374 \times 10^7}$
		MBF	4.950	4.890	4.867	4.738	**4.026**	-
	0.01	**BF**	3.195	5.216	7.213	4.100	3.975	4.740×10^7
		MBF	9.155	9.701	8.898	7.746	7.975	-
	Average `PXover` **BF**		4.788×10^7	5.069×10^7	5.917×10^7	4.876×10^7	$\mathbf{4.441 \times 10^7}$	

Table 6.3: Performance assessment for different values of the parameters Probability of Mutation (`Pmutation`) and Probability of Crossover (`PXover`). Total experiments computational time: 104.2 hours

By inspecting the BF average of the `Pmutation` on the last column, it is notable the decreasing values while reducing the mutation parameter value, reaching its minimum $\mathbf{3.374 \times 10^7}$ at $\texttt{Pmutation} = 0.05$. For the crossover parameter, it is notable an increasing of the BF average until $\texttt{PXover} = 0.8$ and then a decreasing until $\texttt{PXover} = 0.99$ reaching its minimum $\mathbf{4.441 \times 10^7}$. Looking at the BF values, using the combination of parameters values 0.05 and 0.99 it was obtained the best result $\mathbf{3.180 \times 10^7}$. Thus, for the next experiments these parameters will be fixed to $\texttt{Pmutation} = 0.05$ and $\texttt{PXover} = 0.99$.

Population size

For the population size parameter, the same performance assessment was applied but introducing more performance measures such as the Average Fitness Value (AFV) and the Standard Deviation (STD) of the individual fitnesses. Moreover, the runtime of each generation using different population size and the memory usage are also important factors for the decision making. The goal is to find the appropriate population size to solve the current problem. The table 6.4 depicts the results obtained. This parameter

is closely related with the population diversity. The bigger the population is, the more space for different individuals it will have and, as a consequence, increases its diversity and vice-versa. Nevertheless, the evaluation computational time and the memory size associated with the population must be taken into account. Thus, a trade-off between diversity and computational requirements is to be done.

	PopSize					
$\times 10^7$	10	50	100	200	500	1000
BF	10.922	3.153	3.186	3.334	3.532	**2.241**
MBF	12.271	3.911	**3.813**	4.502	5.822	4.982
AFV	7.646	6.822	6.165	5.969	6.838	6.761
STD	1.641	1.039	9.871	1.141	1.567	1.230

atfmGA BF	SCIP UB	Gap
$\mathbf{2.241 \times 10^7}$ \$	$\mathbf{7.595 \times 10^3}$ \$	**294 963%**

Table 6.4: Performance assessment for different values of the parameter Population Size (PopSize). Total experiments computational time: 55.4 hours

Analysing the table 6.4, one can notice that for $\texttt{PopSize} = 1000$ the best BF was obtained $\mathbf{2.241 \times 10^7}$. This was the best result found so far by *atfmGA* reducing the gap to **294 963%**. However, as the figure 6.3 shows, using such a population augments drastically the generation runtime and the memory usage. The biggest drawback of using large populations is indeed the generation runtime. For $\texttt{PopSize} = 1000$, it takes almost 200 seconds to accomplish just one iteration, almost 10 times more than with populations of size $\texttt{PopSize} = 100$. For that reason, it is not worth it to use such a large population. Looking to the other results in 6.4, good results were obtained using $\texttt{PopSize} = 50$ and $\texttt{PopSize} = 100$. Although, with the former parameter value a better BF was obtained, using the latter lower AFV and STD were observed. Hence, the population size of $\texttt{PopSize} = 100$ will be continued to be used in the following experiments.

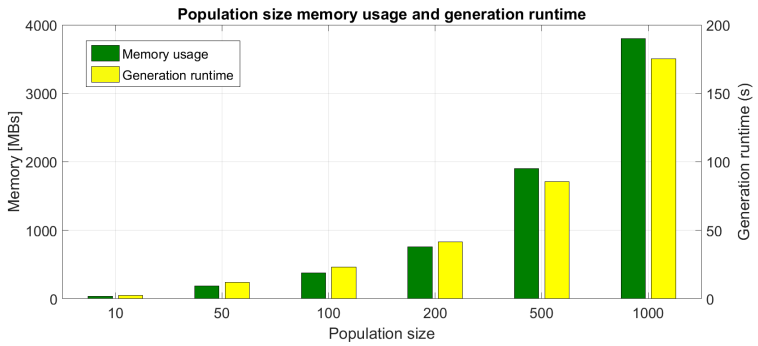


Figure 6.3: Median of the generations runtime and memory usage per generation. As the population size increases, its memory and generation runtime increases.

6.3.3 Online Parameter Initialisation

The last parameter to be tuned `ElitRatio` is very important regarding the GA convergence and it is closely related with one of the fundamental questions in the Evolutionary Algorithms (EA) community. How much should a EA explore and exploit? "Exploration and exploitation are the two cornerstones of problem solving by search" [64]. The terms exploration and exploitation have been playing an important role in describing the working of an algorithm. According to Schippers and Eiben in 1998, the search operators crossover and mutation are responsible for the algorithm's exploration in the search space whilst the selection operator is responsible for the algorithm's exploitation of the problem' solutions.

These two cornerstones can be seen as methods of traversing the problem space being both of them used in conjunction. GA should explore the problem space through crossover and mutation but it should do so by favouring solutions near to other good solutions. This can be accomplished by using biased selection strategies with or without a flavour of elitism. The trick is in finding the right balance. One can go too far into exploitation and it will get stuck in local minima or go too far to exploration and it will waste time on solutions that are less likely to be good. In this GA scheme, elitism is the main driver of exploitation in the search space. It guarantees that the best individuals will take part in the population of the the next generation. The rest of the individuals in the population are subject to exploration encharged by the GA search operators.

To do a performance assessment to this parameter, two types of experiments are carried out. One evaluates different static values of this parameter, the other evaluates an on-line parameter initialisation of `ElitRatio`, `Pmutation` and `PXover` throughout the GA iterations. The goal of these experiments, is to find the right amount of elite solutions that will take place in each generation and consequently evaluate the balance between exploration and exploitation.

The figure 6.4 illustrates the convergence of different experiments with different values of `ElitRatio` in the short run (`MaxGen=100`). One can notice that, for `ElitRatio` $= 0$, which means that elitism is disabled, the convergence stops sooner than the other experiments using different `ElitRatio` values. When elitism is enabled it is noticeable that the curves are shifted to the left, which means that the better solutions are found faster. However, for some experiments using elitism such as `ElitRatio` $= 0.2$ and `ElitRatio` $= 0.4$, the algorithm stopped to improve the solutions quite early but it was able to find better solutions than without elitism. For higher values of `ElitRatio`, the algorithm seems to converge finely in the short run.

Dynamic Parameters Update

To avoid *atfmGA* to be stuck in a local minima and better explore the search space, a Dynamic Parameter Update (DPU) initialisation was implemented. The figure 6.5 shows the dynamic trade-off between exploration and exploitation using the parameters `ElitRatio`, `Pmutation` and `PXover`.

Throughout the generations, two distinct search phases arise namely, the exploration and exploration phases. In the first quarter of the generations the mutation and crossover operators are more active due to high `Pmutation` and `PXover` values and thus intensifying the exploration whilst the elitism only

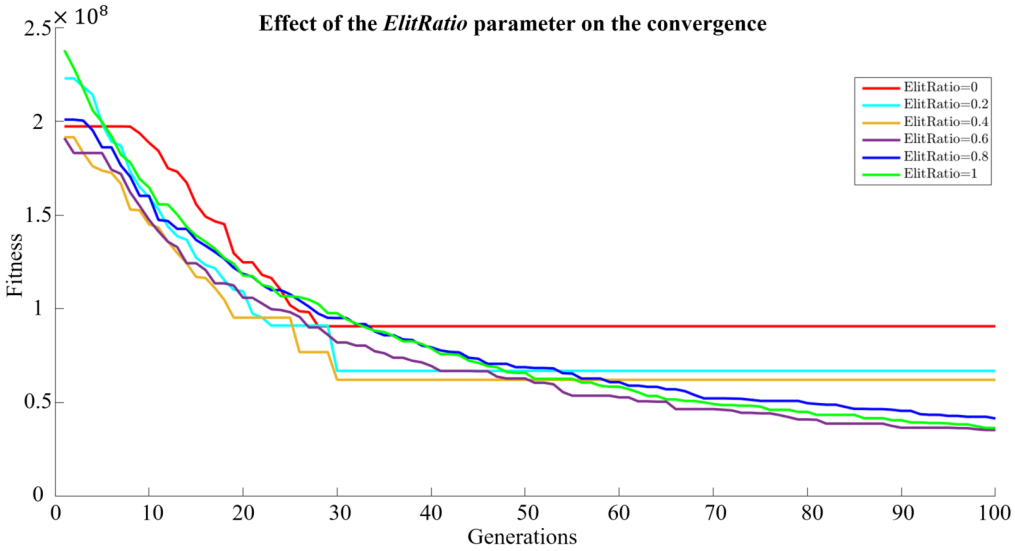


Figure 6.4: Effect of the `ElitRatio` on the convergence within 100 generations. Total experiments computational time: 25 hours.

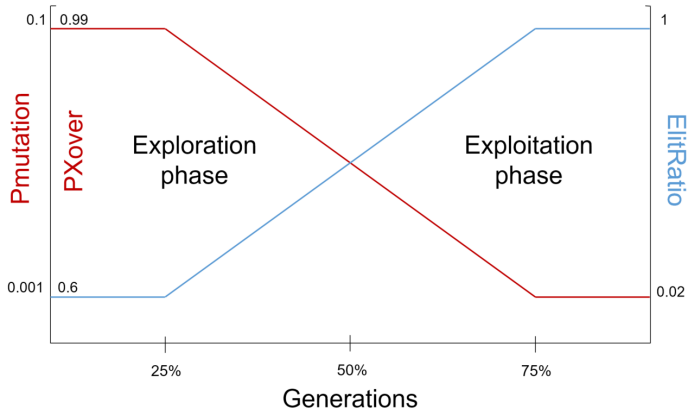


Figure 6.5: Dynamic parameter values update `ElitRatio`, `Pmutation` and `PXover`.

guarantees 2% of the best individuals. After the first 25% of the generation, these parameters change linearly along the generations. The `Pmutation` and `PXover` decrease while `ElitRatio` increases. The exploration phase is overruled by the exploitation phase exactly after the first 50 % of generations. When the search hits 75% of the generations elapsed, the parameters values stabilize into the values shown in the picture 6.5.

Extending the horizon of the search, one can notice in the figure 6.6 that the algorithm, using very low and very high `ElitRatio` values, doesn't improve the UBs, *i.e.* the best solution so far, after few generations. These results indicate that too much exploration doesn't improve the UBs because the algorithm spends too much time to try different solutions that are less likely to be good, ignoring the information about the best individuals already gathered and thus it is not able to improve them. On the other hand, even though in the short run high `ElitRatio` values seemed to be promising, too much exploitation like in the experiment using `ElitRatio` $= 0.8$, causes the algorithm to converge to local minima and get stuck at that point. So a trade-off between exploration and exploitation was found using

dynamic parameters values. In the picture 6.6, the yellow curve presents a total different behaviour from the other two. In the beginning (first 250 generations) the dynamic parameters curve presents a similar behaviour to the blue curve due to the fact that these two experiments have high exploration rates, *i.e.* high `Pmutation` and `PXover` values. After the first quarter of generations, the `ElitRatio` begins to increase and the exploration parameters decrease allowing the GA to start to exploit the diverse individuals which were found by the search operators. In the last quarter of the search, the exploitation dominates and the yellow curve presents a similar behaviour to the red one with some slight UB improvements. In the end of this computational test an improved *atfmGA* BF was found with a cost of $\mathbf{1.075 \times 10^7}$ which reduced the gap to **138 828%**.

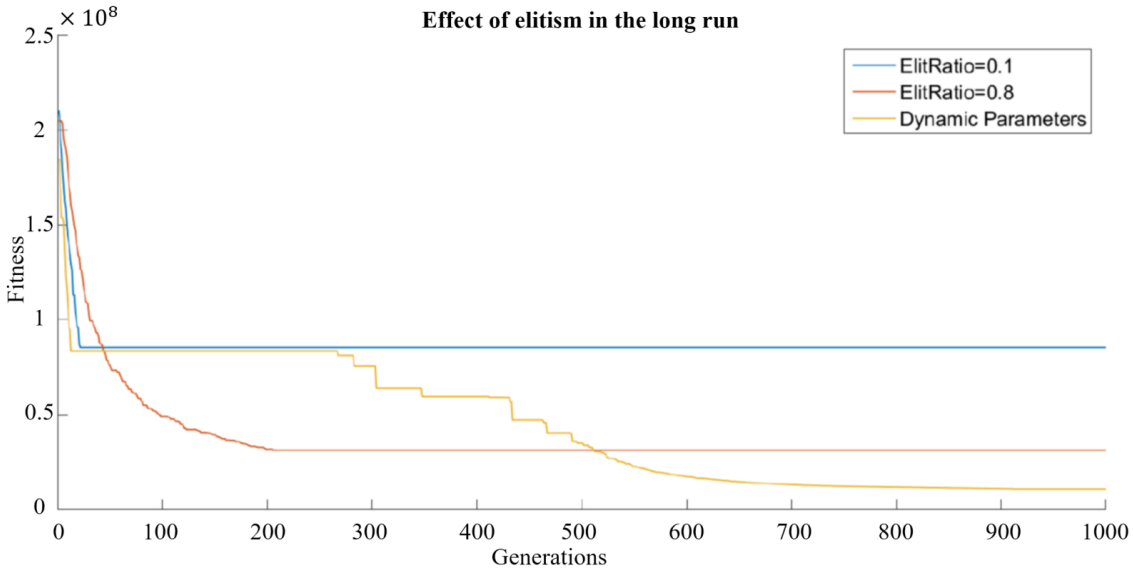


Figure 6.6: Effect of the `ElitRatio` on the convergence within 1000 generations. Total experiments computational time: 15 hours

6.3.4 GA Performance Rate

Another way of measuring the *atfmGA* performance is by analysing its successful rate of finding fitter individuals which improved the best individual found so far which means, in the solution space wise, solutions that improve the UB. Five computational tests using 1000 generations, *Popsize* = 100 and dynamic parameter values were executed in order to evaluate the *atfmGA* performance rate PR. The latter takes into account the computational effort by considering number of solutions that are better than the UB, *i.e.* number of improvements, over the number of objective function evaluations in all the runs. It is computed with the equation 6.2.

$$PR = \frac{\#Improvements}{\#Evaluations \times \#Runs} \tag{6.2}$$

The feasibility rate FR is also computed to assess the *atfmGA* likelihood in finding feasible solutions. It represents the number of feasible solutions over the number of objective function evaluations like in the equation 6.3.

$$FR = \frac{\#Feasible}{\#Evaluations} \tag{6.3}$$

The median of the number of solution evaluations, feasible solutions found, improvements and the BF obtained per each computation test are computed and also used to assess its performance. The table depicts the results obtained.

MBF [$]	t(s)	#Evaluations	#Feasible	#Improvements	FR	PR
1.248×10^7	19 231	75 621	20 182	479	**26.69%**	**0.63%**

atfmGA BF	SCIP best solution	Gap
$\mathbf{1.025 \times 10^7}$ $	7.595×10^3 $	134 828%

Table 6.5: Performance rate assessment of *atfmGA*

The results show that, on average, *atfmGA* finds feasible solutions with a FR = **26.69%** for all solution evaluations. The performance rate, is:

$$PR = \mathbf{0.63}\% \tag{6.4}$$

This result shows how low is the likelihood in finding better UBs using *atfmGA*. Therefore, it was concluded that the search for better UBs using a biased stochastic guided search method is extremely hard. Nevertheless, a new improved BF was found $\mathbf{1.025 \times 10^7}$$. For the best computational test in this experiment, a further analysis was performed to better understand the search process of *atfmGA*. Figure 6.7 illustrates three graphs of the best computation test. The graph on top, shows the number of found feasible solutions per generation. The other two graphs are on the bottom being the blue one (y-axis on the left) the representation of the evolution of the total number of found feasible solutions throughout the generations and the green one (y-axis on the right) the representation of the total number of improvements, *i.e.* the number of solutions which improved the UB. With the last graph one can compute the *atfmGA* performance rate per generation for this simulation.

It is of interest to know how does the number of found feasible solutions evolves throughout the generations. On the graph on top, one can notice that in the first few generations, a great amount of feasible solutions are found. This is explained by the fact that the initial solutions before the generational loop have plenty of cancelled flights and thus it is easier to find feasible solutions by allocating some start-slots other then the cancelled one. The number of found feasible solutions, reaches its minimum at half way of the exploration phase and the curves of the graphs below remain constant. Due to high numbers of draw start-slots in this phase, the search is mainly random and thus, *atfmGA* has difficulties in finding feasible solutions. However, this random search improves the population diversity which will be exploited in the next search phase. After the first quarter of the generations, the exploration rate decreases and the exploitation rate increases. Consequently, the number of feasible solutions and improvements increases which starts to intensify after the first half of the search. This acceleration ceases at the last quarter of

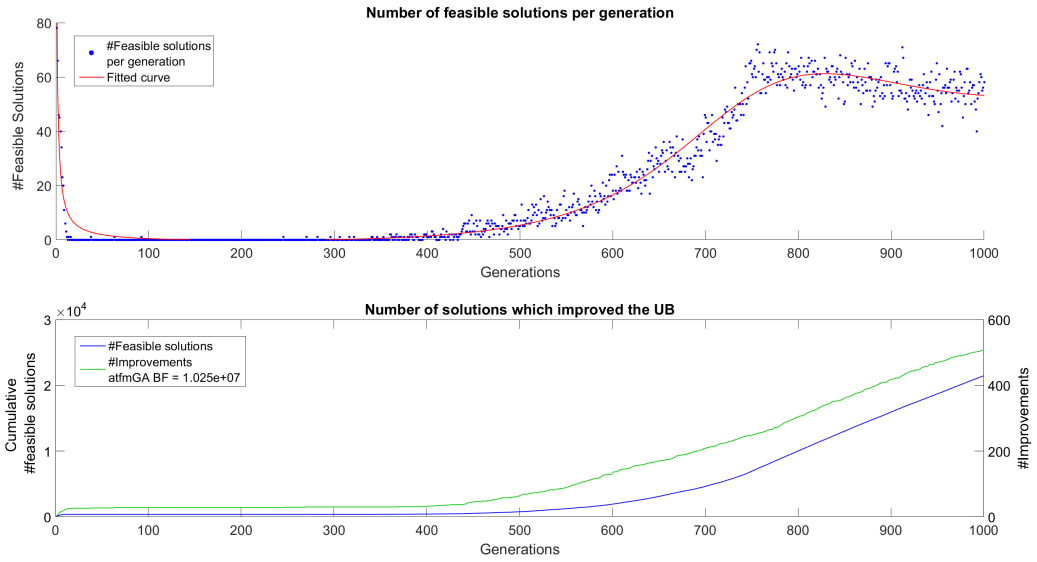


Figure 6.7: Number of feasible solutions and number of UB improvements of the *atfmGA* search

the search in which the search is now exploring with low exploration rates allocating fewer start-slots and fully exploiting only the best solutions which are stored in the SCIP feasible solution storage. Looking at the green graph, the number of improvements continues to increase which is very promising for the next computational tests!

6.3.5 Fitness Penalty Function

To reduce the total evaluation time, an experiment was carried out. Because not all individuals are fit, one could apply a fitness threshold of evaluation acceptance. For this purpose, a simple fitness penalty function was implemented as follows. Instead of evaluating all the individuals in each population, only the ones with better fitness than the UB are worthy to be evaluated. By setting this penalty function, the number of evaluations and consequently the total runtime are expected to be reduced. For the same scenario using the dynamic linear parameters with a `PopSize`=100, a simulation with the same computational time was performed and then fairly compared with the best solution found within this time-frame. The comparative results are shown in the table 6.6.

ATFM Without Conflicts	UB [$]	Gap	t(s)	#Generations	#Evaluations	#Feasible	#Improvements	PR
Without penalty function	1.025×10^7	134 828%	19 231	1 000	85 933	21 468	508	0.591%
With penalty function	$\mathbf{1.429 \times 10^7}$	188 029%	19 231	**1 656**	**81 050**	**471**	**471**	**0.581%**

Table 6.6: *atfmGA* results with and without penalty function

Analysing the table, in the same runtime of 19 231s, *atfmGA* with penalty function can elapse **1 656** generations more than the without penalty function. The amount of evaluations per generation reduced and thus, *atfmGA* can save computational time, respectively 3 582s ($\approx$ 1 hour) in 1 000 generations.

Yet, the UB deteriorated from $\mathbf{1.025 \times 10^7}$ to $\mathbf{1.429 \times 10^7}$. The number of feasible solutions and improved solutions are the same which were reduced significantly, respectively to **471**, and consequently decreasing its PR to **0.581%**. These facts penalizes strongly this approach. Taking into account these results, the fitness penalty function strategy will not take part of the *atfmGA* structure.

After the GA components have been chosen, their parameters tuned and a convergence strategy implemented, the final structure of *atfmGA* is closed. It is now the time to test out the enhanced GA for higher computational times.

6.4 Conflict-free ATFM

The scenario without conflicts doesn't contemplate any en-route potential conflicts and thus, the goal of solving this scenario is purely the minimisation of the delay cost.

6.4.1 Optimal Results

In exact optimisation methods, the efficiency in terms of search time is the main indicator to evaluate the performances of the algorithms as they guarantee the global optimality of solutions. The SCIP solver employs B&B tree search and for each node the relaxed problem is solved by SoPlex using the simplex method.

	UB [$]	LB [$]	Gap	t(s)	#Node	#LP iterations	#Evaluations	#Feasible	#Improvements	PR
ATFM without conflicts	7.595×10^3	7.595×10^3	**0%**	**9.53**	1	**23 688**	23 688	1	1	**0.004%**

Table 6.7: ATFM without conflicts optimal results using SCIP solver without *atfmGA*

SCIP is able to solve this scenario to optimality which means that the UB and LB met each other in the root node after **23 688** LP iterations in **9.53s** and thus reducing the optimality gap to **0%** proving that the optimal solution with cost $\mathbf{7.595 \times 10^3}$ $ was found. SoPlex simplex method has a very low PR however it is extremely fast in finding the optimal solution thanks to LP relaxation which allows the algorithm to freely move through fractional variables exploring the CFPs of the polyhedron always guided towards to the optimal value.

6.4.2 Stochastic Search Results

In order to run *atfmGA* module in SCIP before the B&B branching, it must be called before the root node, recap the sub-section 4.2.2. For that, its properties settings are set to `priority` = 999999, `freq` = 0, `freqofs` = 0, `maxdepth` = 0 and `timing` = "before node". The best result obtained is shown in the table 6.8.

The results for this scenario shows that for 20 000 generations which took 380 468 seconds (more than 4 days) the best result obtained by *atfmGA* was $\mathbf{8.550 \times 10^6}$ $ which is very far way from the optimal solution with a optimality gap of **112 471%**. The AFV of all **4 150 615** individuals evaluations

	UB [$]	LB [$]	Gap	t(s)	#Node	#Generations	#Evaluations	#Feasible	#Improved	PR
ATFM Without Conflicts	$\mathbf{8.550 \times 10^6}$	7.595×10^3	**112 471%**	951 170	-	10 000	4 150 615	861 700	20 850	**0.502%**

Table 6.8: ATFM without conflicts stochastic results using *atfmGA*

was 10.319×10^6 $. It is worth to point out that the share of time spent on evaluating the solutions is a main drawback for the simulations runtime. Each solution evaluation takes 0.1702s. Computing the total time it was found out that for this simulation 282 573s (≈78,5 hours) were spend in evaluating the solutions which represents **74.27%** of the total runtime. However, *atfmGA* is excellent in finding a large number of feasible solutions. In total, 861 700 feasible solutions were found and 20 850 improved the UB with a $PR =$ **0.502%**

6.4.3 Comparative Results

The comparative results of the best solutions found are now presented. The key measurement in this study is the delay cost found by each method. The figure 6.8 depicts the number of start-slots allocations for the optimal solution vector and for the best solution found by *atfmGA* and its performance measures summarized in the table 6.9.

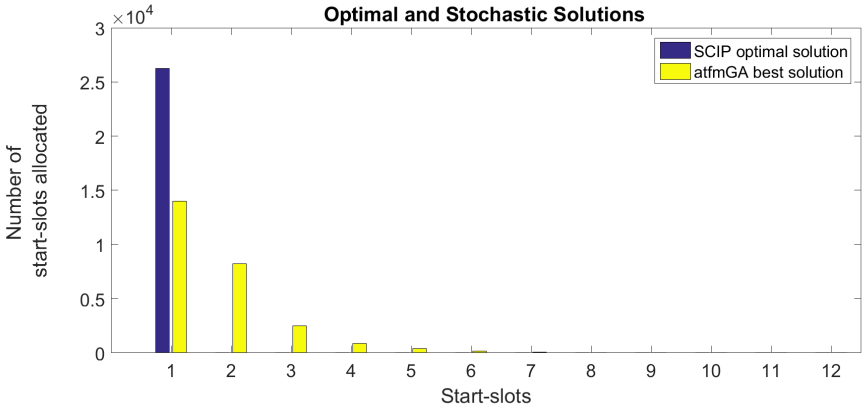


Figure 6.8: Optimal result and *atfmGA* best solution found for the ATFM without conflicts problem

	1st Solution found				**Best Solution found**			
	Solution [$]	$t_{1st}(s)$	Gap_{1st}	Regulated Flights	UB [$]	Gap_{best}	$t_{UB}(s)$	Regulated Flights
SCIP	7.595×10^3	9.53	0%	0.126%	7.595×10^3	0%	9.53	**0.126%**
atfmGA	5.162×10^8	**0.24**	679 648%	92.320%	$\mathbf{8.550 \times 10^6}$	112 471%	380 468 (105.69h)	**46.496%**

Table 6.9: Best results found by SCIP and *atfmGA* for the ATFM without conflicts problem

The table 6.9 shows that *atfmGA* is the first to find the first solution in **0.24**s whilst SCIP manages it in 9.53s. However, the quality of the first *atfmGA* solution is very poor with a first optimality gap of Gap_{1st} = 6 796 476% due to the fact that 92.32% of the flights are regulated.

Analysing the figure 6.8 and the table 6.9 one can notice that in the optimal solution only a tiny percentage of flights, respectively **0.126%**, is regulated whilst in the best solution found by *atfmGA* a

significantly percentage of flights, respectively **46.496%**, are regulated which reflects in an high total delay cost.

6.5 Conflicted ATFM

The scenario with conflicts contemplates conflict probabilities and thus, now the goal of solving this problem is the minimisation of the delay cost and the conflict cost.

The ATFM scenario with conflicts represents an huge large-scale problem. Due to the uncertainties of departure time and the flight's routes interconnectedness, there is an huge number of possible en-route conflicts. For the same date, 93.2% of all fights have conflict probabilities. In total, this accounts for 16 391 398 flight slot pairs that have conflict probabilities. For each of those conflicted flight slot pairs, a conflict surrogate variable and a conflict constraint arises, see 3.18. Consequently, the problem' search space becomes incredibly constrained. Moreover, the size of the problem becomes immensely large and impractical to solve. To reduce its size, a minimum probability of conflict (`minProb`) can be set. The latter will threshold the conflict probabilities and thus reduce the problem' size. The table 6.10 shows the size of the problem for different `minProb` thresholds.

`minProb`	Number of conflicts	SCIP Solution Memory (MB)	$t_{check(s)}$
0.5	7 239	3.8	0.172
0.4	30 439	3.9	0.179
0.3	109 212	5.6	0.215
0.2	379 374	8.0	0.288
0.1	2 177 695	28.6	0.811
0	16 381 398	401.0	-

Table 6.10: ATFM with conflicts size for different minimum conflict probabilities (`minProb`)

The above table shows that when the `minProb` decreases, the size of the problem increases. Since more variables are added to the solution vector, the size of the latter also increases. Also, the computational time to check the feasibility of the solution (done by `SCIPtrySol`) increases. Solving the problem for such a large instance using a population-based metaheuristic like GA can become impractical if the used computer to run simulations lacks of RAM memory. Due to this computational cost constraint, the instance with all the conflicts will not be solved in this work. Instead, four other different instances of the ATFM with conflicts scenario will be solved being the `minProb`=0.4, 0.3, 0.2 and 0.1 the respective thresholds applied for each instance.

6.5.1 Optimal Results

The same SCIP solver was used to solve the four different instances of the ATFM with conflict probabilities. The results are depicted in the table 6.11.

	ATFM with conflicts			
	minProb=0.4	minProb=0.3	minProb=0.2	minProb=0.1
UB [$]	6.091×10^4	2.381×10^5	6.428×10^5	-
LB [$]	6.091×10^4	2.381×10^5	6.428×10^5	$\mathbf{2.395 \times 10^6}$
Gap	0	0	0	-
t(s)	25.79	62.83	452.86	100 213
#Node	1	2	26	2
#LP iterations	47 373	74 298	113 532	1 114 865
#Evalutations	47 373	74 298	113 532	1 114 865
#Feasible	2	7	15	0
#Improvements	2	7	14	0
PR	0.004%	0.009%	0.012%	-

Table 6.11: ATFM with conflicts optimal results using SCIP solver without GA

The results obtained show that SCIP is able to find optimal solutions for instances `minProb` = 0.4, 0.3 and 0.2. However, it cannot find integer solutions for the instance `minProb` = 0.1, *i.e.* any feasible solutions were found. The LB found by SoPlex for a runtime of 100 213s (27.84h) was $\mathbf{2.395 \times 10^6}$.

6.5.2 Stochastic Search Results

The same *atfmGA* module properties in 6.4.2 were used to run the following simulations. The best results obtained are presented in the table 6.12.

ATFM With Conflicts	UB [$]	LB [$]	Gap	t(s)	#Node	#Generations	#Evaluations	#Feasible	#Improved	PR
minProb = 0.4	1.672×10^7	8.091×10^4	20 559%	263 739s (73.26h)	-	15 000	768 700	112 243	1803	0.235%
`minProb` = 0.3	4.141×10^7	2.381×10^5	17 290%	242 810s (67.45h)	-	10 000	339 961	2 412	1 391	0.409%
`minProb` = 0.2	9.403×10^7	6.428×10^5	14 528%	387 782 (107.6h)	-	10000	552 422	93 006	1 540	0.279%
`minProb` = 0.1	1.754×10^8	2.395×10^6	7 225%	246 456s (68.46h)	-	1500	22 049	16	2	0.009%

Table 6.12: ATFM with conflict probabilities stochastic results for different instances using GA

For all the instances, *atfmGA* was able to find UBs. Although the computational time is high, *atfmGA* achieved to find UBs for the instance `minProb` = 0.1 for which SCIP has never did.

6.5.3 Comparative Results

The comparative results of the first and best solutions found are now presented. The figure 6.9 depicts the number of start-slots allocated in the optimal solution vector and in the best solution vector found by *atfmGA*. Their performance measures are summarised in the table 6.13.

Once again, *atfmGA* has proven that it can find the first solution faster than SCIP however badly the solutions are.

		1st Solution found				Best Solution found			
	minProb	Solution [$]	$t_{1st}(s)$	Gap_{1st}	Regulated Flights	UB [$]	Gap_{best}	$t_{UB}(s)$	Regulated Flights
SCIP	0.4	8.332×10^4	11.04	3.16 %	1.305%	8.091×10^4	0%	25.70	0.905 %
	0.3	2.444×10^5	15.67	3.13 %	2.654 %	2.381×10^5	0%	61.50	1.775 %
	0.2	6.794×10^5	25.06	6.72 %	4.174 %	6.428×10^5	0%	429.13	3.922 %
	0.1	-	-	-	-	-	-	-	-
atfmGA	0.4	5.121×10^8	1.56	647 509 %	99 741 %	1.672×10^7	20 559%	263 689s (73.25h)	36.118%
	0.3	5.111×10^8	1.87	209 200 %	99 821 %	4.141×10^7	17 290%	242 668s (67.408h)	42.733%
	0.2	5.107×10^8	5.10	75 066 %	99.214 %	9.403×10^7	14 528%	387 232s(107.6h)	37.997%
	0.1	4.627×10^8	22.09	19 223 %	98.502 %	1.754×10^8	7 225%	20 872s (5.8h)	83.974%

Table 6.13: Best results found by SCIP and *atfmGA* for the ATFM with conflicts for the different instances

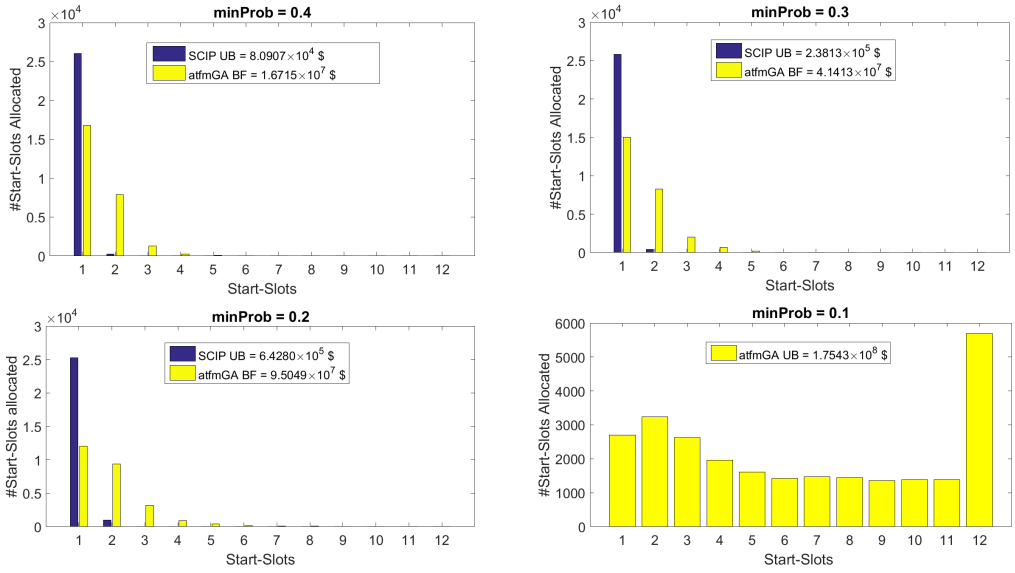


Figure 6.9: Optimal result and *atfmGA* best solution found for the different ATFM with conflicts problem's instances. For the instance minProb = 0.1, only the algorithm *atfmGA* successfully found an UB.

6.5.4 GA Incorporated in the SCIP B&B

To leverage the SCIP' solution process, *atfmGA* could be incorporated in the B&B tree search. This combination can bring good improvements such as:

- Improvement of the effectiveness of heuristics search methods;
- Allows the design of more efficient exact methods.

In this work, a demonstration of how a GA could be integrated in a B&B tree search will be performed. Two conflicted scenarios with `minProb`=0.2 and 0.1, will be used for this purpose.

Recapping the section 4.2.2, one can call a SCIP's module at different stages of the solving process. For instance, the *atfmGA* module `heur_atfmGA` could be firstly called in the first deep level (root node) of tree search and thereafter be re-called in every following deep levels until SCIP finds the optimal solution. For this purpose, the properties of `heur_atfmGA` are set to:

- `priority` = 999999;
- `freq` = 1;
- `freqofs` = 0;
- `maxdepth` = ”no limit”
- `timing` = ”before node”

The goal of this experiment is to solve the problem to optimality using GA integrated in the SCIP B&B. Since SCIP already solves it to optimality in a few hundred of seconds, the number of generations should be low to avoid a long runtime. Therefore, *atfmGA* will run just one generation each time it is called. The previous best found static parameters are now used. In SCIP settings, was activated one heuristic namely *simplerounding* to provide solutions (UBs) to *atfmGA* in the tree search. This heuristic iterates over the set of fractional variables of a LP-feasible point. Then, it performs ”roundings” on the fractional variables. The resulting solution will be integral and may be feasible.

Figure 6.10 illustrates the convergence result for the same instance. The optimality gap can be observed as the distance between the UB and the LB. In the beginning, GA creates the initial population and stores the feasible ones in the SCIP feasible solution storage. Around 180s of simulation time, the SCIP heuristic *simplerounding* found a very good UB and stores it in the SCIP storage. The moment when GA is called once again in the next deep level, it doesn't have to create the initial population over again since it has at its disposal feasible solutions in SCIP storage. Thereby, from this deep level on, GA will run with the solution found by *simplerounding* which can improve other individuals and hopefully vice-versa. After almost 1000 seconds, SCIP stops its solving process because the UB and the LB met each other. Hence, the optimal solution was found (see table 6.11).

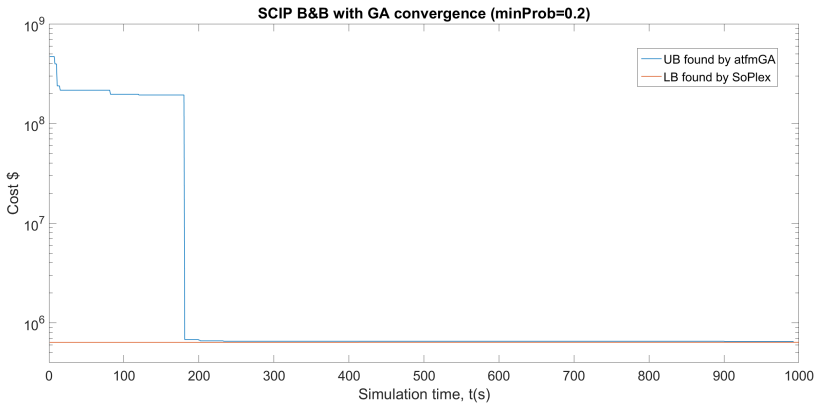


Figure 6.10: Convergence result of SCIP B&B with GA for the ATFM conflict instance `minProb`=0.2. (the y-axis is represented in a logarithmic scale to better visualise the convergence)

For the ATFM problem with `minProb=0.1`, the same properties were used. Per each depth level, *atfmGA* will run 500 generations. The result of the convergence is illustrated in the picture 6.11. A new UB was obtained of 1.729×10^8 $ and a previously obtained LB of 2.395×10^6 for a runtime t = 152 136 s (42.26 hours).

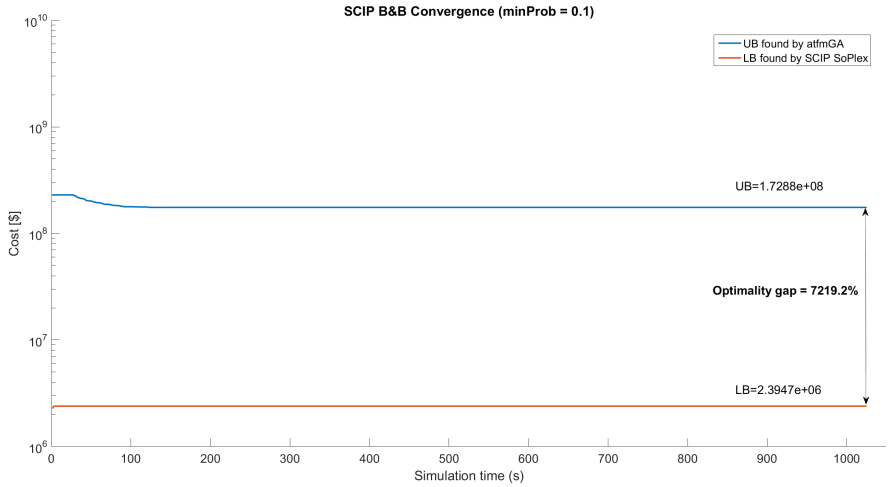


Figure 6.11: Convergence result of SCIP B&B with GA for the ATFM conflicted instance `minProb`=0.1. (the y-axis is represented in a logarithmic scale to better visualise the convergence)

6.6 Solving Performance Progress: Breaking Plateaus in the Fitness Landscape

Throughout the development of this algorithm, better solutions were incrementally found breaking *plateaus* of the fitness landscape represented in 5.4.2. The term *plateau* is a geographical metaphor used to characterize the flatness of the search space [45]. To break or escape from these flat regions, different approaches must be tested. In this work several approaches were tried. Some of them performed well, while others didn't. Looking back to the first version of *atfmGA* which had the most basic operators such as Random Resetting mutation and 1-Point crossover without elitism, a clear progress of the GA solving performance for the problem scenario conflict-free was recorded and illustrated in the figure 6.12.

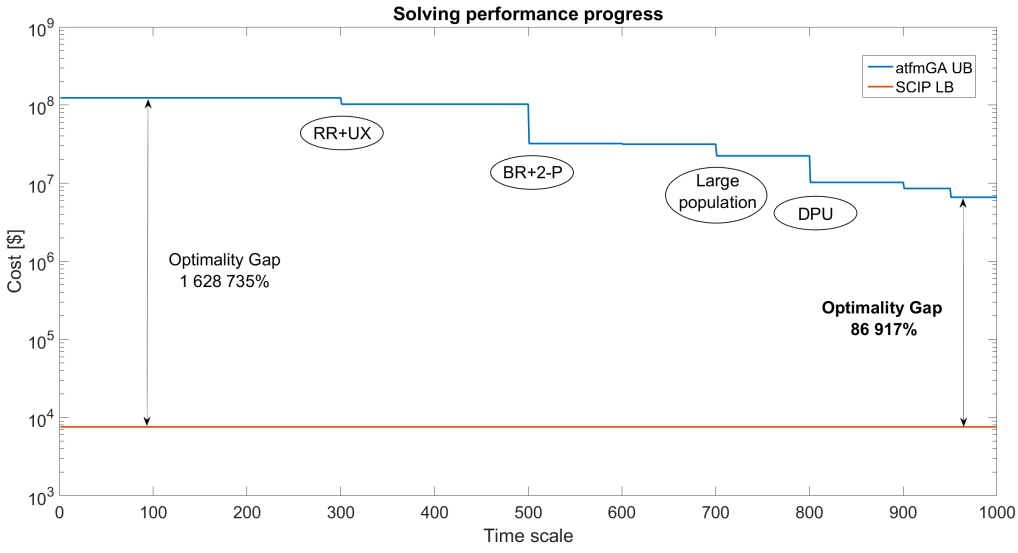


Figure 6.12: Overview of *atfmGA* UB improvements of the ATFM conflict-free problem throughout its development. (the y-axis is represented in a logarithmic scale to better visualise the convergence)

Different improvement stages of this exhaustive search for the solution with lower cost were highlighted. The first significant improvement attained with the introduction of the Uniform Crossover operator. However, another plateau was reached. The operator Bias Resetting mutation proved to be outstanding in breaking flat regions pushing alleles to start-slots with lower delays. Yet, another plateau was encountered. A sensitivity analysis for the *atfmGA* parameters was carried out. A new local minima was reached by running generations with large population. Nonetheless, due to the great computational time in using this population size, this approach was abandoned. Finally, an on-line parameters initialisation fashion, namely Dynamic Performance Update (DPU), was implemented producing excellent results. Using the latest developed idea, long computational tests were executed. To try to improve even further the solution quality, a penalty function was implement. Unfortunately, against the expectations, this approach failed in improving the known *atfmGA* best solution found.

The highlights go to the incredible reduction of the optimality gap from 1 628 735% to **86 917%**. After all these attempts, the best solutions found for each solved problem' scenarios are summarised in the table.

		Conflicted			
	Conflict-free	minProb=0.4	minProb=0.3	minProb=0.2	minProb=0.1
atfmGA [$]	6.609×10^6	1.672×10^7	4.141×10^7	9.505×10^7	1.754×10^8
SCIP [$]	7.595×10^3	8.091×10^4	2.381×10^5	6.428×10^5	-

Table 6.14: Best found solutions by SCIP and *atfmGA* for the ATFM conflict-free and conflicted and its instances

Chapter 7

Conclusions

In this thesis, a new Genetic Algorithm, namely *atfmGA*, for solving large-scale European Air Traffic Flow Management binary problems with conflict probabilities, integrated in the optimisation framework SCIP, was developed. For this purpose, the designed GA, based on problem-specific knowledge, considers the optimisation problem's non-convexity as well the economic objective in the cost function.

The existing Air Traffic Control in Europe was described in Chapter 2 as the basis of the model in Chapter 3. The airspace and airports can only handle a limited number of flights per period. If these are limited by disruptions, their capacities are reduced and security can only be guaranteed for fewer flights. In order to prevent an overload of the limited capacities by the previously scheduled flights, the latter is assigned a delay. Thus, the use of airspaces and airports is postponed by some flights to a later date, where no more congestion threatens.

The problem of assigning a delay to flights was formulated in Chapter 3 as a linear optimisation problem with a total delay to be minimised. In this formulation, every possibility of delaying a flight was represented by a binary decision variable. The *atfmGA* was implemented in the Branch-and-Bound framework SCIP. So that the solver of the latter can be executed from Matlab with the problem data, a programming interface in which SCIP is implemented namely, Matlab Executable (MEX), was provided and described in Chapter 4.

The development of the *atfmGA* is described in Chapter 5. The question of whether it would be worthy to use an Evolutionary Computing framework with implemented functions ready to be used was addressed. It was concluded that, an easy to read, already coded and well documented framework should be used to match the author's programming expertise. The *simpleGA* program was the one chosen to serve as an evolutionary computing basis structure for a more complex and robust GA.

Before the actual *atfmGA* development, an analysis of the problem's fitness landscape was performed to serve as input knowledge in the design of the GA operators. These informations proved to be valuable in the performance of the GA components such as in the generation of the initial population and the mutation operator. In order to enable mutation and crossover operations in an efficient manner, an integer representation, namely genotype, of the solution space was implemented. This representation allows to draw any value in the genotype efficiently without incurring in any violation of the start condition

constraint. Moreover, it decreases the size of the solution by 10 times consequently reducing the computational time of the variable's values drawing. Several genetic operators were designed in Chapter 5 and their performance assessed in the consequent Chapter 6.

The integration of *atfmGA* as a SCIP's module was not an easy task. Several difficulties were encountered throughout this integration specially in the implementation of the data structure and the encoding and decoding scheme. Memory management problems also arose leading to exponential increments of memory due to bad memory allocation. The problems were identified and with the help of the SCIP mailing list, which is very available and helpful, the adversities were surpassed.

In the Chapter 6, the genetic operators which better performed in the tested instance were the chosen ones to integrate the final structure of *atfmGA* followed by a sensitivity analysis to respective parameters. Additional small modifications, such as dynamic parameter update and a penalty function, in the algorithm structure were realized to improve even further the quality of solutions. The dynamic parameter values proved to be beneficial in the search for even better solutions. Yet, the penalty function didn't improve the best solution found by *atfmGA*. The performance assessment was very time consuming, more than 470 hours were spend on "tuning" the algorithm.

Using the final *atfmGA* structure, computational tests with high run-times were carried out for the two problem scenarios and for different instances in order to exploit as must as possible the performance of the developed algorithm in a feasible time-frame. The performance of these computational tests were then compared with SCIP optimal solving process. The results in regard quality of the best solution and computational time, showed that *atfmGA* performs poorly for the problem without conflicts and for instances of the problem with conflicts. This was also experienced in [17] in which a GA was to solve a large-scale ATFM problem. Because the solution space is immensely complex, with multiple local minima and plateaus a deception in solution space probably exists, *i.e.* there might exist a local minimum or group of local minima which attracts *atfmGA* way from the global optimum.

One reason that might explain the poor performance of *atfmGA* on the conflicted scenario's instances is the over-fitting of the parameters for the problem conflicts. A good combination of parameters values and/or operators for one problem, might be disastrous for others and even for the same problems instances this also applies.

However, *atfmGA* is advantageous in finding quickly solutions. For every instances, the results have shown that *atfmGA* always wins the race for the first solution. Moreover, it is able to find a great number of solutions which could be exploited to perform a trade-off between delay cost and conflict cost. Also, *atfmGA* was able to find feasible solutions for the large-scale EATFM with 10% of minimum conflict probability and clearly outperforming SCIP optimal solution process. The UB found is 1.7288×10^8 \$.

7.1 Achievements

The major achievements of the present work was:

- The novel implementation of a genetic algorithm in SCIP framework used in the European Air Traffic Flow Management problem;

- The discovery of Upper Bounds for the European Air Traffic Flow Management problem with 10% of minimum conflict probability.

7.2 Outlook

Pointing out the high importance of the developed operators performance assessment and their respective parameter sensibility analysis and analysing the portion of time spent in this regard, it truly corresponded to about 80%, indicating the great effort that this process demands. One could continue to run more computational tests, *e.g.* the distribution of the Bias Resetting operator could be ”tuned”. Therefore, ”globalise” a Genetic Algorithm scheme for solving large-scale problems creates a discussion topic that should also consider the trade-off between the algorithm performance and expertise time demand, since every problem has different features, and, as a consequence, the search operators have to observe problem-specific knowledge in order to improve their performance. Possibly, seeking the answer for this question will guide the subject to the meta-optimisation and machine learning field, in which Genetic Algorithm theory is part of. All these facts motivate further research regarding the best combination of GA search strategies for solving large problems, *e.g.* adaptive on-line parameters initialisation and exploring the advantage of machine learning and artificial intelligence. Nevertheless, one must have in mind that the stochastic un-guided method towards to the global optimum that GA employs not always produce satisfactory results being non-convex large-scale problems part of this kind of situations.

For this type of problems, deterministic algorithms such as Branch-and-Bound seem to be effective. However, for highly constrained problems with very large instances, the simplex method employed by SoPlex executed throughout the nodes of the tree search cannot find the global optimum and takes a long time to improve the Lower Bounds. To overcome this adversity, a research in deterministic methods applied for this problem needs to be carried out. Using SCIP as a framework, one can try to use different available and already implemented *plug-ins* such as branching rules, cutting plane separators, constraint handlers, node selectors, primal heuristics and so on. Using the *plug-ins* properties, the SCIP module can be called in every stage of the tree search combined with other modules to enhance the solution process and produce reports for every solving stage. This is one of the big advantages of using this framework.

In this work, a novel GA integration in the Branch-and-Bound tree search was implemented. However, this implementation is not yet fully functional and robust. Because the Branch-and-Bound algorithm fixes variables throughout the nodes, one must pay special attention to these fixed variables. If the value of the latter is changed at any time in the tree search the solution containing the same variable will be infeasible once it is evaluated. Code was written to handle this case but failed in capturing the fixed variables.

The developed SCIP module *atfmGA* can be incorporated in the simulation tool R-NEST used by EUROCONTROL to simulate the European Air Traffic Flow Management which uses a similar work-flow used in this work. This incorporation can possibly enhance the simulations carried out nowadays using CASA to help to deliver the best ATFM service to the European network in the SES.

The main drawback of computation time measure is that it depends on the computer characteristics such as the hardware (*e.g.*, processor, memories:RAM and cache), operating systems, language and compilers on which GA is executed. Also, the *atfmGA* functions which generate random numbers are not truly random, but pseudo-random. One may try to use different random numbers functions generators to assess their performance on the search for better UBs.

Finally, one last remark on using GA for such large problems. Optimisation problems are more and more complex and their resource requirement, such as the CPU and memory, are ever increasing. To solve problems of this magnitude, the use of parallel and distributed computing is strongly recommended for the following reasons: speed up the search, improve the quality of the solutions and are able to solve large-scale problems such as the EATFM considering all the conflict probabilities.

Bibliography

[1] T. Lehouillier, F. Soumis, J. Omer, and C. Allignol. Measuring the interactions between air traffic control and flow management using a simulation-based framework. *Computers & Industrial Engineering*, 99:269–279, 2016. ISSN 03608352. doi: 10.1016/j.cie.2016.07.025. URL http://linkinghub.elsevier.com/retrieve/pii/S0360835216302601.

[2] EUROCONTROL. European Aviation in 2040 - Challenges of Growth. pages 1–92, 2018. URL https://www.eurocontrol.int/sites/default/files/content/documents/official-documents/reports/challenges-of-growth-annex-1-25092018.pdf.

[3] EUROCONTROL. CODA Digest 2017. Technical report, 2018. URL https://www.eurocontrol.int/sites/default/files/publication/files/coda-digest-annual-2017.pdf.

[4] EUROCONTROL. CODA Digest 2015. pages 1–34, 2016. URL http://www.eurocontrol.int/publications/coda-digest-2015.

[5] J. Berling, A. Lau, and V. Gollnick. European Air Traffic Flow Management with Strategic Deconfliction. *International Conference of the German, Austrian and Swiss Operations Research Societies 2015*, 2015.

[6] ICAO. *Doc 4444 Air Traffic Management -Procedures for Air Navigation Services (PANS-ATM)*. 16 edition, 2016. ISBN 9789292580810. URL http://flightservicebureau.org/wp-content/uploads/2017/03/ICAO-Doc4444-Pans-Atm-16thEdition-2016-OPSGROUP.pdf.

[7] EUROCONTROL. *SESAR 2020 Concept Of Operations Step 1*. 01 edition, 2017. URL https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5b6d2b912{&}appId=PPGMS.

[8] J. Berling, A. Lau, and V. Gollnick. Strategic Conflict Probabilities in the European Air Traffic Management Network. *Deutscher Luft und Raumfahrtkongress 2016*, 2016.

[9] S. F. Hillier and G. J. Lieberman. Integer Programming. In *Introduction to Operations Research 10th Ed, McGraw-Hill Education*, pages 474–546, 2015.

[10] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*, volume 49. 1998. ISBN 0000000000. doi: 10.1057/palgrave.jors.2600987. URL http://link.springer.com/10.1057/palgrave.jors.2600987.

[11] D. Daniel, J.-m. Alliot, and M. Schoenauer. Genetic algorithms for air traffic assignment. *11th European Conference on Artificial Intelligence*, (May):1–6, 1994. URL https://www.researchgate.net/publication/2489370{_}Genetic{_}Algorithms{_}for{_}Air{_}Traffic{_}Assignment.

[12] D. Delahaye, S. Oussedik, S. Puechmorel, D. Delahaye, S. Oussedik, S. Puechmorel, D. Daniel, A. E. Belin, A. Galli, and A. E. Belin. Airspace congestion smoothing by multi-objective genetic algorithm. *20th Annual ACM Symposium on Applied Computing*, pages 907–912, 2005. URL https://hal-enac.archives-ouvertes.fr/file/index/docid/1004145/filename/Delahaye{_}SAC2005.pdf.

[13] S. Oussedik and D. Delahaye. Reduction of Air Traffic Congestion by Genetic Algorithms. *Parallel Problem Solving from Nature*, pages 855–864, 1998. doi: 10.1007/bfb0056927.

[14] N. Durand, C. Allignol, and N. Barnier. A ground holding model for aircraft deconfliction. *IEEE*, 2010. URL https://hal-enac.archives-ouvertes.fr/hal-00938499/document.

[15] X. Zhang, Y. Zhou, B. Liu, and Z. Wang. The Air Traffic Flow Management with Dynamic Capacity and Co-evolutionary Genetic Algorithm. *IEEE Intelligent Transportation Systems Conference*, pages 580–585, 2007. doi: 10.1109/itsc.2007.4357707.

[16] D. Daniel and A. Odoni. Airspace Congestion Smoothing by Stochastic Optimization. In *IN PROCEEDINGS OF THE SIXTH INTERNATIONAL CONFERENCE ON EVOLUTIONARY PROGRAMMING. NATURAL SELECTION INC*, pages 163—-176. 1997. doi: 10.1.1.23.7802. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.4143{&}rep=rep1{&}type=pdf.

[17] J. Rios and J. Lohn. A Comparison of Optimization Approaches for Nationwide Traffic Flow Management. *AIAA Guidance, Navigation, and Control Conference*, 2009. URL https://www.aviationsystems.arc.nasa.gov/publications/2009/AF2009189.pdf.

[18] G. Marceau, P. Sav, and M. Schoenauer. Strategic Planning in Air Traffic Control as a Multi-objective Stochastic Optimization Problem. 2013. URL https://arxiv.org/pdf/1309.3917.pdf.

[19] W. Tian and M. Hu. Study of Air Traffic Flow Management Optimization Model and Algorithm Based on Multi-Objective Programming. *Second International Conference on Computer Modeling and Simulation*, 2010. doi: 10.1109/ICCMS.2010.20. URL https://www.computer.org/csdl/proceedings/iccms/2010/5642/02/05421094.pdf.

[20] R. Fadil, B. Abou, E. Majd, H. Rahil, H. E. Ghazi, and N. Kaabouch. Multi-objective optimization approach for air traffic flow management. *International Workshop on Transportation and Supply Chain Engineering (IWTSCE'16)*, 00005:1–5, 2017. doi: https://doi.org/10.1051/matecconf/201710500005. URL https://www.matec-conferences.org/articles/matecconf/pdf/2017/19/matecconf{_}iwtsce2017{_}00005.pdf.

[21] D. Bertsimas and S. S. Patterson. The Air Traffic Flow Management Problem with Enroute Capacities. *Operation Research*, (1987):406–422, 1998.

[22] M. Baumgartner, A. Cook, N. Dennis, B. V. Houtte, A. Majumdar, N. Pilon, G. Tanner, and V. Williams. European Air Traffic Management: Principles, Practice and Research. page 255, 2007.

[23] W. Phillipp and F. Gainche. Air traffic flow manament in Europe. *Advanced Technologies for Air Traffic Flow Management.*, 198(Lecture Notes in Control and Information Sciences), 1994.

[24] EUROCONTROL Editorial Team. Twenty Years of European Air Traffic Flow Management. 2015. URL https://www.eurocontrol.int/central-flow-management/building-20-years-central-flow-management.

[25] E. Parliament. Minutes of Procedings of the Sitting of Monday, 13 November 1995. *Official Journal of European Communities*, 38, 1995. URL https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:C:1995:323:FULL{&}from=EN.

[26] D.-G. f. E. and Transport and T. D.-G. for Energy. Single Europe European sky Report of the high-level group. Technical Report November, European Commission, 2000. URL https://web.archive.org/web/20110519234028/http://ec.europa.eu/transport/air{_}portal/traffic{_}management/ses/doc/history/hlgreport{_}en.pdf.

[27] C. R. (EU). 15.7.2011. *Official Journal of the European Union*, (2):1–29, 2011. URL https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2011:185:0001:0029:EN:PDF.

[28] A. Lau, J. Berling, F. Linke, V. Gollnick, and K. Nachtigall. Large-Scale Network Slot Allocation with Dynamic Time Horizons. {*ATM2015*} *11th USA/Europe Air Traffic Management Research and Development Seminar*, 2015.

[29] EUROCONTROL. ATFCM users manual, 2018. URL https://www.eurocontrol.int/sites/default/files/content/documents/nm/network-operations/HANDBOOK/atfcm-users-manual-current.pdf.

[30] EUROCONTROL. ATFCM Operating Procedures for Flow Management Position, 2014. URL https://www.eurocontrol.int/sites/default/files/content/documents/nm/network-operations/HANDBOOK/atfcm-ops-procedures-fmp-current.pdf.

[31] B. Houot. Network Manager - IFPS users manual. 2018. URL https://www.eurocontrol.int/sites/default/files/content/documents/nm/network-operations/HANDBOOK/ifps-users-manual-next.pdf.

[32] N. O. Handbook. Provision of cacd data. 2017. URL https://www.eurocontrol.int/sites/default/files/content/documents/nm/network-operations/HANDBOOK/provision-CACD-data-current.pdf.

[33] M. Růžičková. *Global Flight Processing Systems*. Msc thesis, Czech Technical University in Prague, 2016. URL https://dspace.cvut.cz/bitstream/handle/10467/65890/F6-BP-2016-Ruzickova-Marcela-GlobalFlightPlanProcessingSystems.pdf?sequence=-1.

[34] EUROCONTROL. NM Area of Operations, 2018. URL http://www.nm.eurocontrol.int/STATIC/NM{_}AREA/imports/ATFM{_}AREAS.JPG.

[35] FAA - Air Traffic Organization. *Traffic Flow Management in the National Airspace System*. Number October. 2009. URL https://www.fly.faa.gov/Products/Training/Traffic{_}Management{_}for{_}Pilots/TFM{_}in{_}the{_}NAS{_}Booklet{_}ca10.pdf.

[36] P. R. Commission. Evaluating the True Cost to Airlanes of One Minute of Airborne or Ground Delay. Technical Report May, Eurocontrol, 2004. URL https://www.eurocontrol.int/sites/default/files/content/documents/sesar/business-case/evaluating{_}true{_}cost{_}of{_}delay{_}2004.pdf.

[37] T. Leggat, C. Yven, and Y. D. Wandeler. A Matter of Time : Air Traffic Delay in Europe. Technical report, EUROCONTROL, 2009. URL https://www.eurocontrol.int/sites/default/files/publication/files/tat2-air-traffic-delay-europe-2007.pdf.

[38] P. R. Commission. PRR 2017 Performance Review Report. (May):86, 2018. URL https://www.eurocontrol.int/sites/default/files/publication/files/prr-2017.pdf.

[39] A. Cook. European airline delay cost reference values. Technical Report December, University of Westminster, London, 2015. URL http://westminsterresearch.wmin.ac.uk/19702/1/Europeanairlinedelaycostreferencevalues-updatedandextendedvalues{%}28V4.1{%}29.pdf.

[40] A. Lau, R. Budde, J. Berling, and V. Gollnick. The Network Flow Environment: Slot Allocation Model Evaluation with Convective Nowcasting. *29th Congress of the Internatioal Council of the Aeronautical Sciences 2014*, 2014.

[41] A. Majumdar. The factors affecting air traffic controller workload : a multivariate analysis based upon simulation modelling of controller workload . pages 1–41, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.629.5387{&}rep=rep1{&}type=pdf.

[42] G. Tobaruela, P. Fransen, W. Schuster, W. Y. Ochieng, and A. Majumdar. Air traffic predictability framework Development, performance evaluation and application. *Journal of Air Transport Management*, 39:48–58, 2014. ISSN 0969-6997. doi: 10.1016/j.jairtraman.2014.04.001. URL http://dx.doi.org/10.1016/j.jairtraman.2014.04.001.

[43] I. Kaysi. Dynamic Network Models and Driver Information Systems. *Transportantion Research Board*, 25(5):251–266, 1991.

[44] T. Achterberg. *Constraint Integer Programming*. Phd thesis, Fakultät II – Mathematik und Naturwissenschaften der Technischen Universität Berlin, 2007.

[45] E.-G. Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009. ISBN 0470278587, 9780470278581.

[46] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994. URL https://sophia.javeriana.edu.co/{~}cbustaca/docencia/DSBP-2018-01/recursos/ErichGamma,RichardHelm,RalphJohnson,JohnM.Vlissides-DesignPatterns{_}ElementsofReusableObject-OrientedSoftware-Addison-WesleyProfessional(.pdf.

[47] A. Liefooghe, L. Jourdan, and E.-g. Talbi. A software framework based on a conceptual unified model for evolutionary multiobjective optimization : ParadisEO-MOEO. *European Journal of Operational Research*, 209(2):104–112, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2010.07.023. URL http://dx.doi.org/10.1016/j.ejor.2010.07.023.

[48] D. L. Woodruff. Optimization Software Class. pages 1–24, 2002.

[49] Olivier Mallet. GALGO 2.0 Genetic Algorithm in C++ with template metaprogramming and abstration for constraint optimization. URL https://github.com/olmallet81/GALGO-2.0.

[50] E. López-Camacho. jMetalCpp, an object-oriented C++-based framework for solving multi-objective optimization problems with metaheuristics. URL http://jmetalcpp.sourceforge.net/.

[51] E. López-Camacho, M. J. García Godoy, A. J. Nebro, and J. F. Aldana-Montes. JMetalCpp: Optimizing molecular docking problems with a C++ metaheuristic framework. *Bioinformatics*, 30(3): 437–438, 2014. ISSN 13674803. doi: 10.1093/bioinformatics/btt679.

[52] E. López-Camacho, M. J. García Godoy, J. García-Nieto, A. J. Nebro, and J. F. Aldana-Montes. Solving molecular flexible docking problems with metaheuristics: A comparative study. *Applied Soft Computing Journal*, 28:379–393, 2015. ISSN 15684946. doi: 10.1016/j.asoc.2014.10.049. URL http://dx.doi.org/10.1016/j.asoc.2014.10.049.

[53] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi. openGA , a C ++ Genetic Algorithm library. *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017. doi: 10.1109/SMC.2017.8122921.

[54] M. Wall. GAlib : A C ++ Library of Genetic Algorithm Components. (August), 1996. URL http://lancet.mit.edu/ga/dist/galibdoc.pdf.

[55] M. Barkaoui, J. Berger, and A. Boukhtouta. Customer satisfaction in dynamic vehicle routing problem with time windows. *Applied Soft Computing Journal*, pages 1–11, 2015. ISSN 1568-4946. doi: 10.1016/j.asoc.2015.06.035. URL http://dx.doi.org/10.1016/j.asoc.2015.06.035.

[56] L. Pradenas, J. Garcés, V. Parada, and J. Ferland. Genotype – phenotype heuristic approaches for a cutting stock problem with circular patterns. *Engineering Applications of Artificial Intelligence*, 26(10):2349–2355, 2013. ISSN 0952-1976. doi: 10.1016/j.engappai.2013.08.003. URL http://dx.doi.org/10.1016/j.engappai.2013.08.003.

[57] S. Yuan, B. Skinner, S. Huang, and D. Liu. A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *European Journal of Operational Research*, 228(1):72–82, 2013. ISSN 0377-2217. doi: 10.1016/j.ejor.2013.01.043. URL http://dx.doi.org/10.1016/j.ejor.2013.01.043.

[58] J. A. Fernandez-prieto, J. Canada-bago, M. A. Gadeo-martos, and J. R. Velasco. Optimisation of control parameters for genetic algorithms to test computer networks under realistic traffic loads. *Applied Soft Computing Journal*, 12(7):1875–1883, 2012. ISSN 1568-4946. doi: 10.1016/j.asoc.2012.04.018. URL http://dx.doi.org/10.1016/j.asoc.2012.04.018.

[59] John Burkardt. A Simple Genetic Algorithm. URL https://people.sc.fsu.edu/{~}jburkardt/cpp{_}src/simple{_}ga/simple{_}ga.cpp.

[60] Zbigniew Michalewicz. *Genetic Algorithms + Data-Structures = Evolution-Programs*. Springer, 1992. ISBN 3-540-60676-9.

[61] A. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag GmbH, New York, 1 edition, 2003. ISBN 9783642072857. doi: 10.1007/978-3-662-05094-1.

[62] J.H.Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[63] J. Berling. *Konzeptionierung und Implementierung einer Lösungsmethode zur Anwendung in einem europäischen taktischen Luftverkehrsflussmodell*. Msc thesis, Technischen Universität Hamburg-Harburg, 2014.

[64] C. A. Schippers and A. Eiben. On evolutionary exploration and exploitation. 35: 1–16, 1998. URL http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=B4E714BC12A043E4E6213C794594AF11?doi=10.1.1.29.4885{&}rep=rep1{&}type=pdf.

Appendix A

Vector Formulation

A.1 Variables, Network Elements and Cost Vectors

The decision variables, the delay costs, the sector and airport both arrival and departure capacities can be summarized in vectors.

- **Vector of decision variables:** For each flight f there is a column vector x_f with all possible delays $d = 1, ..., D$ per flight,

$$\mathbf{x_f} = [x_{f,1} \ ... \ x_{f,D}] \quad \in \mathbb{Z}^D \tag{A.1}$$

  The complete vector of decision variables consists in the set of vector of all possible delays of all flights $f = 1, ..., F$,

$$\mathbf{x} = [\mathbf{x_1} \ ... \ \mathbf{x_F}] \quad \in \mathbb{Z}^{F.D} \tag{A.2}$$

  Jointly with the number of flights F and the number of ATFM slots D, the decision vector $\mathbf{x}$ has the length $n = F.D$.

- **Delay cost vector:** For each flight f there is a column vector w_f with all possible delays $d = 1, ..., D$ per flight which contains:

$$\mathbf{w_f} = [w_{f,1} \ ... \ w_{f,D}]^\top \quad \in \mathbb{Z}^D \tag{A.3}$$

  The complete vector of decision variables consists in the set of vector of all possible delays of all flights $f = 1, ..., F$,

$$\mathbf{w} = [\mathbf{w_1} \ ... \ \mathbf{w_F}]^\top \quad \in \mathbb{Z}^{F.D} \tag{A.4}$$

  The vector of delay costs w is analogous to $\mathbf{x}$.

- **Sector capacity vector:**

  For each time interval t, there is a column vector $\mathbf{c_t}$ which contains the sector capacitances $\mathbf{c_{t,s}}$ of all sectors in this time interval,

$$\mathbf{c_t} = [c_{t,1} \ ... \ c_{t,S}]^\top \quad \in \mathbb{Z}^S \tag{A.5}$$

  The complete sector capacity vector consists in the sector capacity vectors for each interval $t = 1, ..., T$,

$$\mathbf{c} = [\mathbf{c_1} \ ... \ \mathbf{c_T}]^\top \quad \in \mathbb{Z}^{S.T} \tag{A.6}$$

  Jointly with the number of sectors S and the number of time intervals T, the sector capacity vector $\mathbf{c}$ has the length $m = S.T$.

- **Airport capacity vector:** For each arrival t_h and departure t_j time interval, there are column vectors $\mathbf{i_{t_h}}$ and $\mathbf{o_{t_j}}$ which contain respectively the airport arrival $\mathbf{i_{t_h,h}}$ and departure $\mathbf{o_{tj,j}}$ capacitances of all airports in this time interval,

  For arrivals:

$$\mathbf{i_{t_h}} = [\mathbf{i}_{t_h,1} \ ... \ \mathbf{i}_{t_h,H}]^\top \quad \in \mathbb{Z}^H \tag{A.7}$$

  For departures:

$$\mathbf{o_{t_j}} = [\mathbf{o}_{t_j,1} \ ... \ \mathbf{o}_{t_j,J}]^\top \quad \in \mathbb{Z}^J \tag{A.8}$$

  The complete airport capacity vector consists in the airport capacity vectors for both arrival and departure's time interval $t_h = 1, ..., T_h$ and $t_j = 1, ..., T_j$ respectively,

  For arrivals:

$$\mathbf{i} = [\mathbf{I_1} \ ... \ \mathbf{i_{T_h}}]^\top \quad \in \mathbb{Z}^{H.T_h} \tag{A.9}$$

  For departures:

$$\mathbf{o} = [\mathbf{O_1} \ ... \ \mathbf{o_{T_j}}]^\top \quad \in \mathbb{Z}^{J.T_j} \tag{A.10}$$

  Jointly with the number of airports[1] B and the number of timeslots[2] T, the airport capacity vectors for both arrival $\mathbf{i}$ and departure $\mathbf{o}$ have the length $l = B.T$.

[1] The total number of arrival airports H and departure airports J is the same and so, this number is simply represented by B

[2] The total number of arrival timeslots T_h and departure timeslots T_j is the same and so, this number is simply represented by T

A.2 Constraints

A.2.1 Start Condition

To be able to write this constrain in a vector formulation two new coefficients were introduced in the equation.

$$\sum_{d=0}^{D(f)} g_{f,(f.d)} \cdot x_{f,d} = e_f, \quad \forall s,t \tag{A.11}$$

On the left hand side of the equation the coefficient $g_{f,(f.d)}$ is always equal to one. On the right hand side the coefficient e_f is also equal to one.

The coefficients g can be written in a matrix $\mathbf{G} \in F \times (F.D)$. Every coefficient g in line of $\mathbf{G}_f$ is one for every elements of $\mathbf{x}_f$ (see definition A.1). The coefficients $\mathbf{e} \in F$. Thus, resulting in the equation A.12:

$$\mathbf{Gx} = \mathbf{e} \tag{A.12}$$

A.2.2 Sector Capacity

With the coefficients a of the matrix $A \in (S.T \times F.D)$, the decision vector $\mathbf{x}$ and the capacity vector $\mathbf{c}$, the capacity's constrain can be written in matrix notation formulate as it follows:

$$\mathbf{Ax} = \mathbf{c} \tag{A.13}$$

A.2.3 Airport Capacity

Arrivals

With the coefficients r of the matrix $R \in (H.T \times F.D)$, the decision vector $\mathbf{x}$ and the capacity vector $\mathbf{i}$, the capacity's constrain can be written in matrix notation formulate as it follows:

$$\mathbf{Rx} = \mathbf{i} \tag{A.14}$$

Departures

With the coefficients p of the matrix $P \in (A.T \times F.D)$, the decision vector $\mathbf{x}$ and the capacity vector $\mathbf{o}$, the capacity's constrain can be written in matrix notation formulate as it follows:

$$\mathbf{Px} = \mathbf{o} \tag{A.15}$$

A.2.4 Surrogate Constraints

In a vector form:

$$\mathbf{Bx} + \mathbf{My} \leq \mathbf{i} \tag{A.16}$$

Appendix B

SCIP Technical Configurations

B.1 SCIP and Matlab Data

To prepare for implementation in SCIP, this section describes the data structures of it. The problem to be solved is created in SCIP in a data structure of type SCIP. The variables and constraints are added to the problem. To associate the variables with the constraints, the variables are added with their coefficients to the constraints, *e.g.* when a start-slot variable has a flight entering a capacity time window, the variable is added to the constraint with coefficient 1. All data types contain a variety of management data, such as the elapsed time, the number of solutions or the solving status.

- **Problem's object:** The basic data object in SCIP is the `scip` structure. All variables, constraints, coefficients, settings, information about the solution process, etc are stored in the `scip` structure.

- **Variables:** The basis of the data object SCIP and the linear programming are the variables `SCIP_Var`. There are several types of variables in SCIP, but in this work only the variables of type binary are used. Each variable has a LB and an UB. Of course, for the binary variables, these bounds are zero and one respectively. Furthermore, the variables are initialized with their cost coefficients. Firstly a variable is created using the `SCIPcreateVarBasic()` method and then added to the SCIP problem using the `SCIPaddVar()` function. The functions are called with various parameters. The variables are accessed via pointers `SCIP_VAR*`.

- **Constraints:** The constraints in SCIP are managed by the constraint handlers. A constraint handler is responsible for testing for feasibility, adding new constraints to the LP, managing the dual variables, etc. [44]. In this work, the setppc[1] constraint handler is used for the start condition ($= 1$), the knapsack constraint handler for the capacity constraint ($\leq c$) and conflict surrogate constraints.

[1] Setppc stands for set partitioning (start condition = 1) set packing and set covering constraints

B.2 SCIP - MEX Interface

Matlab offers a programming interface to call a program from external code just like a normal Matlab function. The Opti-Toolbox[2] provides such an interface named Matlab executable function[3] (MEX) with integrated SCIP libraries. With MEX functionality, programs from `C/C ++` or Fortran can be called like a separate function. Also, the source code and a makefile for the SCIP-MEX are included.

B.2.1 Matlab Executable

To transfer the data to the solver, a so-called MEX file is created. The mentioned file can only be created in Matlab. To create a MEX file, the source code must include the Matlab libraries of the programming interface. To build MEX, Matlab offers a build script. The latter calls a compiler that generates the MEX file from the program code. It is also possible to integrate runtime libraries. Furthermore, a so-called gateway is needed, which processes the input and output arguments in the code. The input arguments started by MEX in Matlab are defined as a pointer array. The position in the argument list is equal to the index in the pointer array. The Matlab data is converted to data structures of the programming language used, in this case `C`, using methods of the inserted Matlab libraries.

ATFM - MEX

The solver of the ATFM problem works with the framework SCIP. An adapted interface based on Opti-Toolbox provides a basis for integrating SCIP into a MEX file. It provides a makefile of MEX, which can be built, a guide to the build process and a source code of its implementation. In order to their functionality become available for the solution process, SCIP libraries are integrated in the source code. Furthermore, runtime libraries are integrated by external programs because they are required by SCIP. The interface with Matlab is re-implemented to accept the ATFM problem. The generation of the problem with SCIP is also newly implemented. From the source code, a MEX function is built by means of an external compiler[4]. The MEX functions can be called in Matlab with the command `scip()` with the respective input arguments. To solve the linear ATFM problem, it must first be created from the NFE data. In order to be able to create it in SCIP, the required data is passed as arguments to the MEX function `scip()`. The SCIP-MEX creates a linear problem and solves it later.

NFE Data Structure

The database for the ATFM problem is provided by the NFE developed at ILT. This section describes the existing data structures from which the ATFM-LP is generated in SCIP. The NFE data consists in:

- **Time steps:** The model time is set from the earliest start time to the latest landing time of all included aircraft movements, plus the maximum possible ground holdings. For the application of

[2]OPTimization Interface Toolbox is a free MATLAB toolbox for constructing and solving linear, nonlinear, continuous and discrete optimization problems. https://www.inverseproblem.co.nz/OPTI/

[3]Matlab EXecutable: https://www.mathworks.com/help/matlab/ref/mex.html

[4]The ATFM-MEX is built with the Microsoft Visual Studio 2013 Compiler (https://visualstudio.microsoft.com/vs/)

linear optimization, discrete time segments are generated. The Matlab variable `t.mod` consists of the time steps $(1, ..., T)$ from the first start time to the last possible landing time;

- **Profile Points:** The routes of the flights are stored in so-called point profiles, *i.e.* planned trajectories. It contains all the locations where a flight overflies on its route, including flight's duration. For each flight, the departure (*ADEP*) and destination (*ADES*) aerodromes as well as the ETOT are available;

- **Calculated Time Over(CTOs):** The overflight times, in 15 minutes time step, over the sectors are calculated from the profile points. These are stored in the matrix *ft* (flight time). The matrix links the flights with the sectors in a way that the rows represent the flights and the columns the sectors. When a flight enters in a sector, the flight duration is stored in the element where the flight row and sector column match. Sectors that are not on the route are marked with `NaN`[5];

- **Capacities:** Sectors and both aerodrome departure (*D*) and arrival (*A*) capacities are stored in matrix form. The capacities indicate the maximum number of entries for each time step.

B.2.2 Data Treatment

The data from NFE must be processed in advance. The individual steps are explained in the following section.

Non-Regulated Flights

In the model, some flights are exempted. These flights do not receive an ATFM slot, but start at their scheduled departure time (ETOT). They will not be included in the SCIP linear problem. Instead, their capacity requirements are calculated in the sectors entering by the planned trajectories. Thereby, the capacity available in SCIP for scheduling is the nominal capacity minus the capacity required by the unregulated flights. There are three types of non-regulated flights:

- Flights starting outside the ATFM zone;

- Flights to which one airport can not be assigned;

- Flights that start before the scenario-relevant time period.

Selection of solution-relevant sectors and flights

Only the sectors, departure and destination aerodromes where traffic demand exists are considered. The indexing is adjusted accordingly, so that the remaining flights are mapped to the remaining capacities.

[5]NaN stands for "not a number"

Routes (assignment of variables to constraints)

To create the linear problem in SCIP, there should be a data structure that can be interpreted as quickly and conveniently as possible. Matlab's existing flight time matrix *ft* which contains `NaNs` for non-entering sectors. It is unsuitable for rapid processing in `C` because firstly all sectors (and not just those sectors of the planned trajectory) need to be checked for one flight and secondly the `NaNs` in `C` have no equivalent[6].

Two matrices are created: The *SectorProfile.Route* matrix contains all flights routes and, in turn, the indices of the respective entering sectors, the "Sector Profiles". The other matrix is the *SectorProfile.Time* contains the ETO times to the sectors.

Capacities

The existing capacities in the NFE are influenced by two factors before being handed over to SCIP. First, the capacity requirements of non-regulated but existing flights are deducted.

Delay Costs of ATFM slots

Every aircraft's take-off induce certain costs. This is implemented in the work as follows. The matrix *cost* contains the delay costs of each flight for each possible delay and respective cancellation. Consequently, the variables that inflict a certain delay to an aircraft take-off will be passed into SCIP with their respective cost coefficients.

Conflict Probability Costs of ATFM

The conflict probabilities are handled by an huge square matrix, each entry representing a conflict cost for a pair of flights in their respective time-slots delays. Because only a small set of the flights have actual conflict probabilities, to be precise 16 391 398 pair of flights points have conflict probability different than zero, the matrix *ConflictedMatrix* is a immensely sparse with a density[7] of 0.0312% and a sparsity[8] of 99.9608%. Due to memory limitations, only the upper half of matrix is treated since it is a square matrix and thus, the other half is symmetric which have the same conflicted flights. A conflict cost can be arbitrary defined to compute the conflict probability cost given by the product between conflict probability and conflict cost. Now that the conflict probabilities cost are defined, flights can be afflicted with different conflict costs depending on their respective conflict probability.

B.2.3 Parsing ATFM Problem Data from Matlab and its Initialization in SCIP

This sub-chapter describes how to pass the data of the Matlab problem to the solver's MEX and how to construct the linear ATFM problem. The input data and the generation of the linear ATFM problem from the same data is also described.

[6] Since the flight time can also be zero, the only way to handle the matrix *ft* in SCIP would be to set the flight time to the maximum value for all `NaNs`

[7] The sparse matrix density is calculated as follows $100\% \times \frac{number\ of\ non-zero\ elements}{number\ of\ elements}$

[8] The sparse matrix sparsity is calculated as follows $100\% - density$

SCIP and Matlab data

It follows the specification of the input arguments for the ATFM solver and its correspondent data in SCIP. The MEX of the solver is started in Matlab with the function `scipmex()` with the arguments exposed in tables with its respective description. The Matlab data input corresponds respectively to the data in SCIP as the tables B.1, B.2 and B.3 show.

The dataset related with the network elements consists in its available quantities and its capacities. It also contains the number of time-slots and the maximum number of sectors which one flight is allowed to enter in its route. The mentioned data is exposed in the table B.1.

Matlab	SCIP	Dimension	Description
-	`nT`	-	Number of time-slots
-	`nSECTORS`	-	Number of sectors
-	`nADR`	-	Number of aerodromes
t	`*T`	nT	Indexes of the time-slots
C	`*Csec`	$nT \times nSECTORS$	Sector capacities for all time-slots
D	`*Cdep`	$nT \times nADR$	Departure aerodrome capacities for all time-slots
A	`*Cdes`	$nT \times nADR$	Destination aerodrome capacities for all time-slots
maxEnrouteS	`MAX_SECTORS_ON_ROUTE`	-	Maximum numbers of sectors on one flight route

Table B.1: Network elements' data

The dataset related with the intrinsic flights, maps each flight position to the entering sector in a particular time-slot. It also assigns each flight to its respective departure and destination aerodrome and to the time-slot of departure/arrival. It consists in the number of flights for each scenario, its delay costs for each start-slot option and all its ETOs to the network elements for the *initialDelay* scenario. In this work, a zero initial delay scenario is considered. The number of start-slots options, i.e. ATFM departure delays, is also contemplated in this dataset. It is set to 12 and it consists in the maximum number of delays $MAX\_DELAY = 10$ plus the on time start-slot, *i.e.* no departure delay, and the cancelled flight start-slot which once allocated to a flight its departure is cancelled. The mentioned data is exposed in the table B.2.

Matlab	SCIP	Dimension	Description
-	`nFLIGHTS`	-	Number of flights
-	`nSTARTSLOTSOPTS`	-	Number of start slots options
initialDelay	`*Xzero`	$nFLIGHTS \times 1$	Start zero delays vector
depT	`*Tdep`	$nFLIGHTS \times 1$	Estimated departure times
arrT	`*Tarr`	$nFLIGHTS \times 1$	Estimated arrival times
sectorProfileRouteOptimizer	`*SectorProfileRoute`	$nFLIGHTS \times MAX\_SECTORS\_ON\_ROUTE$	All sectors en-route enters for all flights.
sectorProfileTime	`*SectorProfileTime`	$nFLIGHTS \times MAX\_SECTORS\_ON\_ROUTE$	All entry times in sectors (CTOs) en-route for all flights
adep	`*FlightDeparture`	$nFLIGHTS \times 1$	Indexes of the departure aerodromes in which flights have departed
ades	`*FlightDestination`	$nFLIGHTS \times 1$	Indexes of the destination aerodromes in which flights have arrived
cost	`*FlightSlotCost`	$nFLIGTHS \times nSTARTSLOTSOPTS$	Cost of all start-slots (delay slots) per flight

Table B.2: Flights' data

The conflicts' dataset is represented by a large square sparse matrix, namely conflicted matrix. It consists in conflict probabilities for all possible conflicts between flights for every time-slots, see table

B.3. Matlab gets the information of the non-zero elements, *i.e.* their ith row and the j column indices. Then, Matlab passes to *scipmex* through the MEX function `mxGetlr` the sparse array which contains the non-zero elements information.

Matlab	SCIP	Dimension	Description
ConflictedMatrix	`*conflictedMatrix`	$(nFLIGHTS \times (nSTARTSLOTSOPTS - 1))^2$	Conflict probability
conflictCost	`conflictCost`	-	Cost of each conflict

Table B.3: Conflicts' data

SCIP Output Arguments

SCIP outputs a solution vector *xScip*, which contains all start-slots options assigned to flights. For each flight, a start-slot option is assigned and marked with a 1.

B.2.4 Generation of the ATFM-LP from the Data

When the ATFM-MEX is called in Matlab, the programmed sequence starts running in the main source file *scipmex.cpp*. First, the main function `mexFunction()`[9] is invoked and reads out all Matlab input data listed and generates `C` data structures from them. Finally, the actual SCIP process begins.

First, the `SCIPcreate(&scip)` function is used to create a `scip` object, in which all information, constraints, variables, heuristics, etc. are stored. The solution is then prepared by loading the default plugins, creating an empty linear problem in the *scip-struct*, and allocating memory for the data structures. The generation of the actual ATFM problem is explained next.

Create and Add ATFM Variables

The variables are first created in SCIP and then added. First, all variables are created as a *struct* of type *SCIP_Var* using the `SCIPcreateVarBasic()` method. The variables are generated as binary variables and each receive their cost coefficient.

There are two types of variables in this problem:

- **Flight Variables:** The flight variables are generated as binary variables and each receive their delay cost coefficient `FlightSlotCost(f, d)`. Using the method `SCIPaddVar(f,d)` the variables are added to the problem one by one per flight. This step is accomplished by using two nested loops. The outer loop runs with index `f` over the number of flights (`nFLIGHTS`). The inner loop runs with index `d` over the number of possible start-slots $(MAX\_DELAY + 1)$[10]. In each iteration, a variable `vars(f, d)` is created using the `SCIPcreateVarBasic()` function and then added to the problem using the *SCIPaddVar()* function. When the computation is complete, the function outputs a `SCIP_OKAY`. This signals to SCIP that the function has completed correctly.

[9] Is the entry point `C/C++` MEX function built with `C` Matrix API. More information, see https://www.mathworks.com/help/matlab/apiref/mexfunction.html

[10] Matlab uses 1-based indexing, whereas C uses 0-based indexing. The index of a Matlab array is subtracted by 1, when the element is called the first time in a function-chain

- **Surrogate Variables:** The surrogate variables are generated as binary variables and each receive their conflict cost coefficient from the conflict probabilities sparce matrix and the defined conflict cost. This step is accomplished by using two nested loops over the conflict probability matrix elements. Because each element of this matrix represents pair of two variables `vars(`f_1`,`d_1`)` and `vars(`f_2`,`d_2`)`, *i.e.* two flights f_1 and f_2 with their respective departure time-slots d_1 and d_2, it is possible two check for every existing conflict. If a pair of flights is in conflict, the conflicted matrix element has an associated conflict probability, otherwise the element is a *NaN*. Thereby, the outer loop runs with the `vars(`f_1`,`d_1`)` element's index, *i.e.* runs through the columns of the conflicted matrix, over the number of elements of the same. If there is a different number of non-zero elements in the current column than in the previous column, there is at least one conflict in the current column. The inner loop runs with the conflict's row-index over the current column and get the conflicted flights and their respective conflicted departure time-slots. Only the conflicts of the upper triangle are taking into account. If `vars(`f_1`,`d_1`)` and `vars(`f_2`,`d_2`)` exist, then these variables will be used to create and add the respective conflict surrogate. In each iteration, a surrogate variable `qobj` is created using the `SCIPcreateVarBasic()` function and then added to the problem using the `SCIPaddVar()` function with respective conflict probability cost. When the computation is complete, the function outputs a `SCIP_OKAY`. This signals to SCIP that the function has completed correctly.

Create and Add ATFM Constraints

After all variables have been created and added to the *scip* problem, the constraints (of type `SCIP_Cons`) are created. For the constraints, memory is first allocated. Then the constraints are created and added to the *scip* problem.

There are three type of constraints in this problem:

- **Start Constraint:** using `SCIPcreateAndAddAllSetppcCons()` method, every flight is subject to this type of constraint;
- **Knapsack:** using `SCIPcreateAndAddAllKnapsackCons()` method, every network element for all their respective time-slots is subject to this type of constraint;
- **Surrogate:** using `createAndAddConflict()` function, every conflicted pair of flights with their respective conflicted departure time-slots is subject to this type of constraint.

Link Variables to Constraints

At the implementation time described here, all variables and constraints have been created and added to the *scip* problem. In a LP, variables are columns and the rows of the constraints are inside. In SCIP, the variables are linked to the constraints. This is realized by adding a coefficient and a variable together to a constraint. The constraint now consists of a variable with its coefficients. However, the coefficients are not stored in the variable.

As each aircraft can only be at a particular location at a specific time at one time, coefficients of the ATFM problem set to one, are added to the capacity time-slot afflicted constraint of that network element as an entry count. However, if a flight enters in more than one network element in a time interval less than the network element's time-step (15 mins), the same flight can count network element's entries for more than one network element's capacity constraint for the same start-slot d. In a loop, all coefficients are successively added to the respective constraints.

The variable will be linked to the constraints in which the flight f is linked to the delay d. When all the variables are linked to the respective capacity afflicted constraints, the creation of the ATFM problem is complete.