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
Instituto Superior Tecnico, Universidade de Lisboa, Lisboa, Portugal
November 2018
andreflserrano@gmail.com

***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 Branch-and-Bound 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.**

***Index Terms*—Air Taffic Flow Management, Slot Allocation, Binary Integer Programming, Branch and Bound, Genetic Algorithm**

I. Introduction

Airport and airspace congestion is an inherent problem worldwide, frequently resulting in substantial flight delays, re-routings and even cancelations 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]. Consequently, a great number of conflicts between aircrafts emerge in the en-route area. To avoid such conflicts, an aircraft position forecast should be used and an Air Traffic Flow Management (ATFM) must be provided improving the safety and efficiency of air transportation. This may have significant effects on the safety and efficiency of air transportation, reducing the need for controller interventions [2]. However, sources of error in trajectories spatial and temporal prediction, span a wide range of factors such as meteorological conditions, human behavior, 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. Distributions of departure times' deviation in Europe, show that more than half of all fights deviate at least five minutes [3]. Therefore, there is a probability associated to potential conflicts, namely, conflict probabilities [4]. Allocating alternative departure timeslots to planned flights can reduce conflict probabilities but also should not increase flight take-off's delays significantly. Conflict probabilities are computed for all the possible departure time combinations. Moreover, many flights have conflicts with multiple other flights. 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 timeslots combination.

In this thesis, a Genetic Algorithm (GA) integrated in the Branch-and-Bound framework SCIP (Solving Constraint Integer Programs) is developed to solve ATFM conflict-free and conflicted problems minimising delay and conflict costs. This document is organized as follows: Section II the concept of ATFM is explained. Also, a brief analysis of delays in Europe is carried out. Section III provides a short description of the ATFM network model. The mathematical formulations for ATFM conflict-free and conflicted problems are presented. Section IV introduces SCIP with special focus on the primal heuristics incorporation in the Branch-and-Bound framework solving process. Section V describes the development of the GA scheme and its integration in SCIP. In Section V the results obtained are presented. The performance of the deterministic algorithm that SCIP employs will be compared with the stochastic method of GA, namely *atfmGA*. Section VII finishes the document, drawing some conclusions and suggesting some future work.

II. Air Traffic Flow Management in Europe

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 expeditions flow of air traffic by ensuring that the existing network elements (airspace sectors and airports) capacity are used to the maxi-

Hamburg University of Technology (TUHH), Institute for Air Transportation Systems (ILT)

mum extend possible and that the traffic volume is compatible with capacities declared by the appropriate authority with timely, accurate information for planning and execution of an economical air transport.

A. 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 Air Operators. This ATFM procedure is as known as Ground Delay Program (GDP) [5]. 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 [6]. 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.

Delays result for a large number of reasons such for a lack of resource, mechanical issues, planned processes, etc...The figure 1 depicts the causes of departure delay analysed by EUROCONTROL.

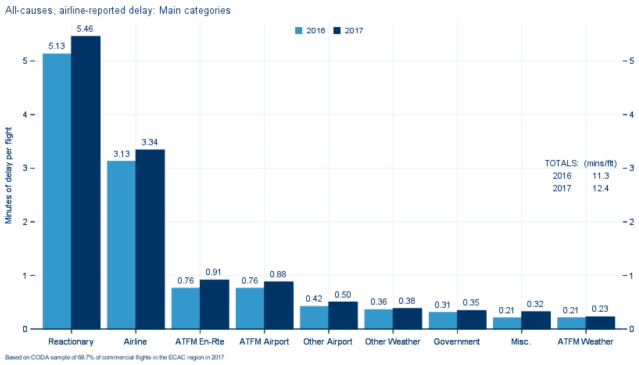


Fig. 1. All causes of delay in Europe with the respective average delay per flight in 2016 and 2017, source [7]

B. 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, [8], *i.e.* the cost of an extra minute delay for an average flight is estimated for all aircraft operating in Europe at EUR 100. 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.

III. Modelling Work-flow and Data Set

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 [9]. 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.

A. Air Traffic Flow Management Mathematical Formulation Problem

The model is based on 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, an discrete ATFM slot time is specified according to a selected time-step. Each decision variable is associated with a fixed route that defines which network elements are capacity-afflicted by a flight. The problem is characterized 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. 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' time-slots 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. 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. Next, the individual components of the linear model are presented.

1) Decision Variable: Following the work of Bertsimas and Stock Patterson (1998) [10], 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,

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

2) 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 $.

3) Start Condition: The departure constraint ensures that every flight f is assigned to exactly one departure d then we have the following start condition constraint:

$$\sum_{d \in D(f)} x_{f,d} = 1, \quad \forall f \tag{2}$$

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 timeslot 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.

5) Sector Capacity: The coefficient a assigns the flight f with delay d to the sector s in the particular timeslot t which corresponds to the Calculated Time Over (CTO) of the respective flight.

$$a_{(s,t),(f,d)} = \begin{cases} 1, & \text{if } CTO_s(f,d) = t \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

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.

$$\sum_{f \in F} \sum_{d \in D(f)} a_{(s,t)(f,d)} \cdot x_{f,d} \leq c_s, \quad \forall s,t \tag{4}$$

Airport arrival and departure capacity constraints observe the same constraint equation as 4 but with different indexing.

The ATFM problem is then formulated as a Binary Problem with binary decision variables, a cost function and linear constraints allowing the application of methods of linear optimisation.

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 5.

$$Z(x) = min\left(\sum_{f} \sum_{d} \omega_{f,d} \cdot x_{f,d} \right) \tag{5}$$

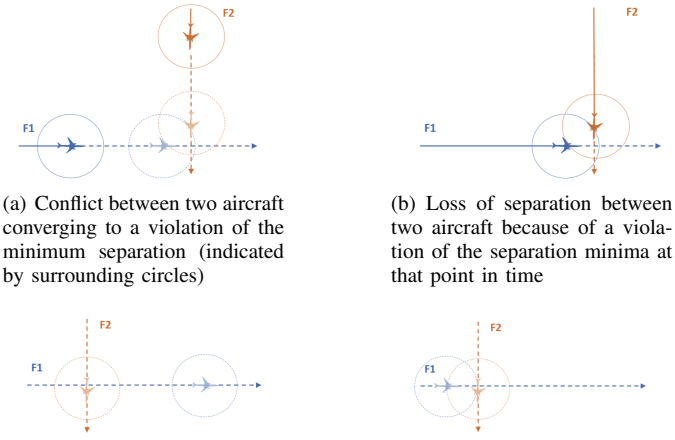
B. Air Traffic Flow Management with Strategic Deconfliction

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 [11]. 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 2 illustrates. To resolve conflicts, the ATCs provide pilots with instructions to perform airborne manoeuvres. ATCs have workload capacity which impacts sectors capacity [12]. 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 [12]. 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.

C. 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 2.b. To enable strategic deconfliction, every planned flight path point is checked against points of other flights for infringement of the separation minima [2].

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 2.c. Conversely, if flight F2 is delayed 15 minutes, there is a strategic conflict, see 2.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

Fig. 2. Strategic deconfliction

The components of the ATFM with strategic deconfliction[1] are:

1) Conflict cost: Every strategic conflict probability between two flights $f^{'}$ and $f^{''}$ with departure slots $x_{f',d'}$ and $x_{f'',d''}$ 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 6 shows.

$$k_{f',d',f'',d''} = \frac{1}{2} \cdot P_c(x_{f',d'}, x_{f'',d''}) \tag{6}$$

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 7.

$$Z(x) = min\Bigg(\sum_{f} \sum_{d} \omega_{f,d} \cdot x_{f,d} + \sum_{f'} \sum_{d'} \sum_{f''} \sum_{d''} k_{f',d',f'',d''} \cdot x_{f',d'} \cdot x_{f'',d''} \Bigg) \tag{7}$$

To linearise the quadratic conflict cost, a surrogate variable has to replace the product $x_{f',d'} \cdot x_{f'',d''}$ and so equation 8 holds:

$$y_{f',d',f'',d''} = x_{f',d'} \cdot x_{f'',d''} \tag{8}$$

2) Surrogate Constraint: The linearisation of the conflict term shown in the equation 8, provides a new constraint, namely, conflict surrogate (1),

$$x_{f',d'} + x_{f'',d''} - y_{f',d',f'',d''} \leq 1 \tag{9}$$

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:

$$Z(x) = min\Bigg(\sum_{f} \sum_{d} \omega_{f,d} \cdot x_{f,d} + \sum_{f'} \sum_{d'} \sum_{f''} \sum_{d''} k_{f',d',f'',d''} \cdot y_{f',d',f'',d''} \Bigg) \tag{10}$$

IV. Solving Binary Integer Programming by Branch-and-Bound

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 [2]. 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. 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

A. SCIP - Solving Constraint Integer Programming

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 upper and lower bound.

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 specialized 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. 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.

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 customized 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 interact with the framework through a very detailed interface.

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 illustrated in figure 3.

V. Genetic Algorithm

GAs were developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg and he has been tried on various optimization problems with a high degree of success. Genetic Algorithm (GA) is a metaheuristic inspired by the a 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. This algorithm is sufficiently randomized in nature performing much better than other search methods such as random local search [13]. Genetic Algorithms have the ability

[1] or ATFM conflicted problem

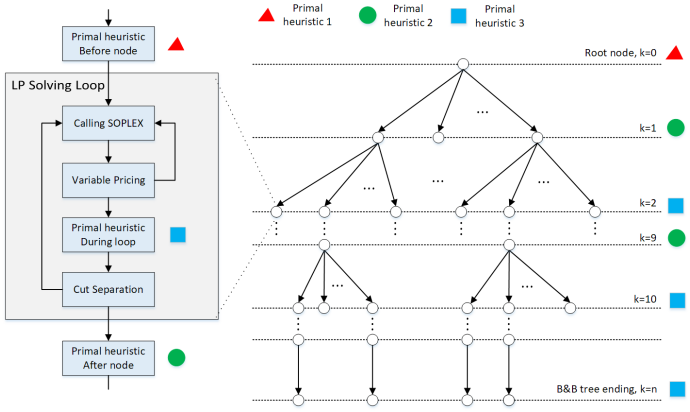


Fig. 3. Branch-and-Bound tree search and primal heuristics execution calls in the SCIP solving process

to deliver a 'good-enough' solution 'fast-enough'. This makes GAs attractive for use in solving optimisation problems.

A. Algorithm Scheme

GA starts with a initial set of solutions, each one of them represented by a chromosome[2]. 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 children. 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 repeated until GA reached the stopping criteria. GA iteratively applies the generation of new population and the replacement of a new one.

B. Fitness Landscape Analysis

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 [13]. The goal of this analysis, is to attempt to predict the behaviour of the GA' search components.

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, and so on) and the twelfth start-slot correspond 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 4 depicts a sum of the flights delay costs per start-slot. 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 5.206×10^8\$.

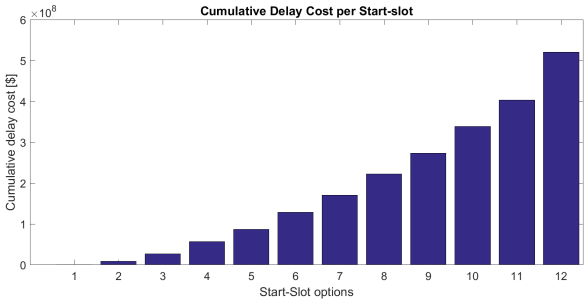


Fig. 4. Cumulative delay cost per start-slot

C. Solution Representation

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 start constraint 2, only one start-slot option can be assigned to a given flight as the table I depicts. Thus, in the encoding space, the solution representation uses the following encoding variable namely *gene*.

$$\varphi(f) = d \qquad \text{if the start-slot } d \text{ is assigned to flight } f \quad (11)$$

Each $\varphi(f)$, *i.e.* gene, has a value, namely *allele*, corresponding to the start-slot option d_k index within the discrete interval $[1, 12]$. In the table I, 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 II. 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_{10}	d_{11}	d_{12}
Flights	f_1	0	0	1		0	0	0
	f_2	0	0	0		0	0	1
	.				.			
	.				.			
	.				.			
	f_n	1	0	0		0	0	0

TABLE I
PROBLEM' SOLUTION VARIABLES

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

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

TABLE II
FLIGHT'S VECTOR

This transformation ensures that the start constraint 2 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 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 II, 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 matrix. Therefore, the flight f_2 was allocated with a cancelled flight departure slot.

D. Initial population

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.

E. 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.

1) 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 P its probability to be select is:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_i} \tag{12}$$

2) 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 illustrates. To select μ individuals, the tournament procedure is performed μ times.

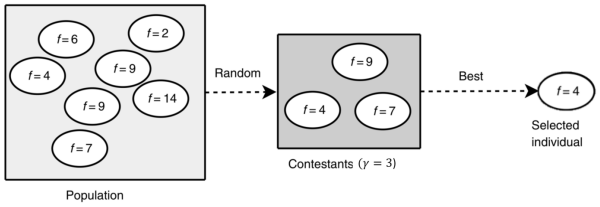


Fig. 5. Tournament selection strategy. In this example, a tournament of size $k = 3$ is carry out. Three solutions are selected randomly from the population. The best individual is then selected. Adapted from source [13]

F. 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.

1) 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. Moreover, the performance of this operator largely depends on the 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 this work, the n-point crossover and uniform crossover were implemented

2) The n-point crossover: In the 1-point crossover, a crossover point k is computed randomly which spans the chromosome's length. Then, two segments per chromosome are formed separated in the kth position and thereafter interchanged them resulting in two offspring as the figure 6 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 6 depicts.

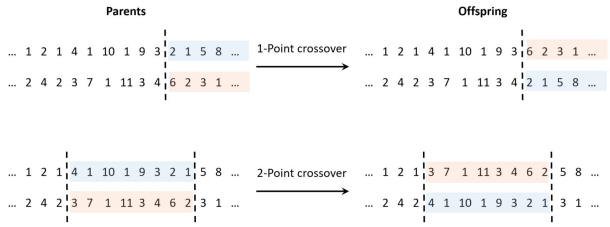


Fig. 6. n-Point crossover operator. In the upper part of image it is illustrated the 1-point crossover and in the bottom part of the figure it is illustrated the 2-point crossover

3) 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 7 shows.

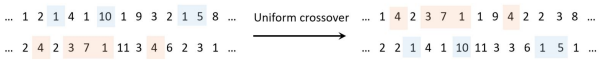


Fig. 7. The uniform crossover operator

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

For this problem, a mutation in the discrete representation is to be implemented. According to [14] 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.

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

6) 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. The figure 8 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.

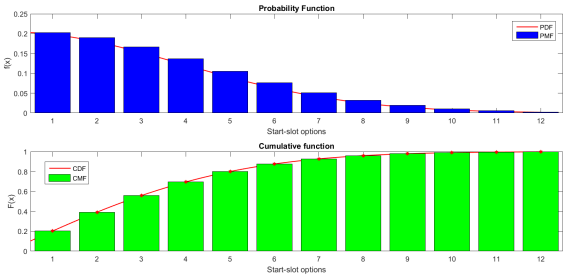


Fig. 8. Half-Normal distributions

7) 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 9 shows the normal distribution used in this operator with mean $\mu = 0$ and standard deviation $\sigma = 2.7$.

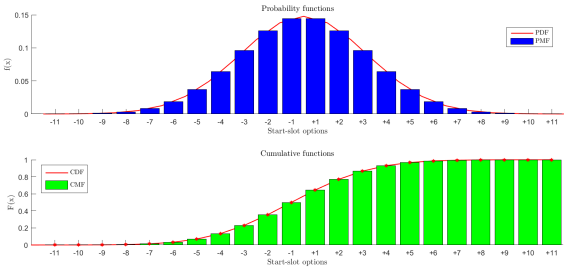


Fig. 9. Normal distributions

G. 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. The strategy implemented was the traditional generational replacement proposed by [15]. This strategy replaces the whole population of size μ. The offspring population will replace systematically the parent population. 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 10. 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.

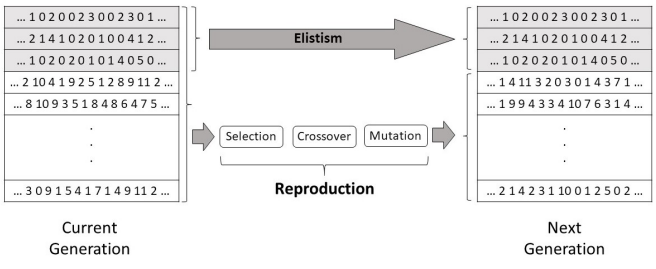


Fig. 10. Replacement strategy - Elitism

H. 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, this problem has an unknown optimal objective value. 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 return the best solutions found and its iteration statistics.

I. GA flowchart

The figure 11 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, the *Selection*, the *Reproduction*, the *Evaluation* and the *Replacement*.

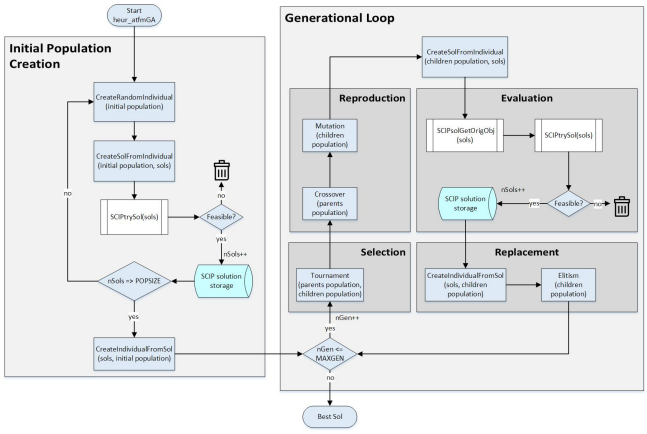


Fig. 11. GA flowchart

VI. Results

A. Solved Scenarios: Conflict-free and Conflicted

A performance assessment of the solver was carried out using two problem scenarios are used, the scenario conflict-free and the conflicted scenario. The first one was 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. 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.

The ATFM conflicted scenario represents a 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 9. 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.

B. GA Performance Assessment

The performance of GA depends on many factors[3] 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. The selected instance is the full day scenario without conflicts described in VI-A. 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 13.

$$Gap = \frac{UB - LB}{min(UB, LB)} \times 100\% \tag{13}$$

C. 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 12 shows the dynamic trade-off between exploration and exploitation using the parameters `ElitRatio`, `Pmutation` and `PXover`. The parameter `ElitRatio` dictates the number of best individuals found so far will be part of the next population. The other two are respectively the probability of mutation and crossover.

Computational tests were carried out using this strategy and an improved *atfmGA* solution was found with a cost of $\mathbf{1.075 \times 10^7}$ which reduced the gap to **138 828%**.

D. GA Performance Rate

A way of measuring the *atfmGA* performance is by analysing its successful rate of finding 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

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

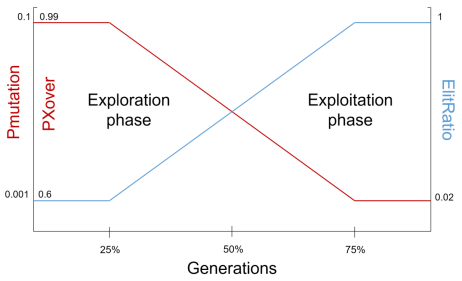


Fig. 12. Dynamic parameter values update `ElitRatio`, `Pmutation` and `PXover`.

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 14.

$$PR = \frac{\#Improvements}{\#Evaluations \times \#Runs} \tag{14}$$

With the obtained results from different computational tests, the performance rate of *atfmGA* was computed:

$$PR = \mathbf{0.63}\% \tag{15}$$

This result shows how low is the likelihood in finding better UBs using *atfmGA*. A new improved solution 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*.

E. ATFM Conflict-free Problem

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 13 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 III.

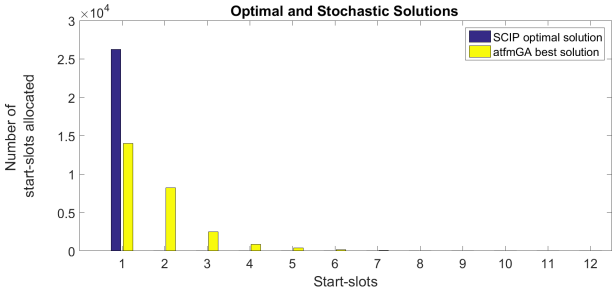


Fig. 13. Optimal result and *atfmGA* best solution found for the ATFM conflict-free problem

The figure 13 shows 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.

	Best Solution found			
	UB [$]	Gap_{best}	$t_{UB}(s)$	Regulated Flights
SCIP	7.595×10^3	0%	9.53	**0.126%**
atfmGA	$\mathbf{8.550 \times 10^6}$	112 471%	380 468	**46.496%**

TABLE III
BEST RESULTS FOUND BY SCIP AND *atfmGA* FOR THE ATFM WITHOUT CONFLICTS PROBLEM

F. ATFM conflicted problem

The comparative results of the first and best solutions found are now presented. The figure 14 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 summarized in the table IV.

		Best Solution found			
		UB [$]	Gap_{best}	$t_{UB}(s)$	Regulated Flights
SCIP	0.4	8.091×10^4	0%	25.70	0.905 %
	0.3	2.381×10^5	0%	61.50	1.775 %
	0.2	6.428×10^5	0%	429.13	3.922 %
	0.1	-	-	-	-
atfmGA	0.4	1.672×10^7	20 559%	263 689s (73.25h)	36.118%
	0.3	4.141×10^7	17 290%	242 668s (67.408h)	42.733%
	0.2	9.403×10^7	14 528%	387 232s(107.6h)	37.997%
	0.1	1.754×10^8	7 225%	20 872s (5.8h)	83.974%

TABLE IV
BEST RESULTS FOUND BY SCIP AND *atfmGA* FOR THE ATFM WITH CONFLICTS FOR THE DIFFERENT INSTANCES

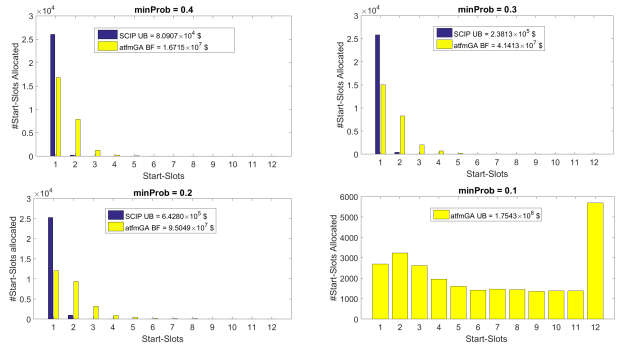


Fig. 14. Optimal result and *atfmGA* best solution found for the different ATFM with conflicts problem's instances. For the instance minProb = 0.1, only the algorithm *atfmGA* successfully found an UB.

G. GA Incorporated in the SCIP B&B

To leverage the SCIP solution process, *atfmGA* could be incorporated in the B&B tree search. SCIP's module can be called 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 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 15. A new UB was obtained of $\mathbf{1.729 \times 10^8}$ \$ and a LB of 2.395×10^6 for a runtime t = 152 136 s (42.26 hours).

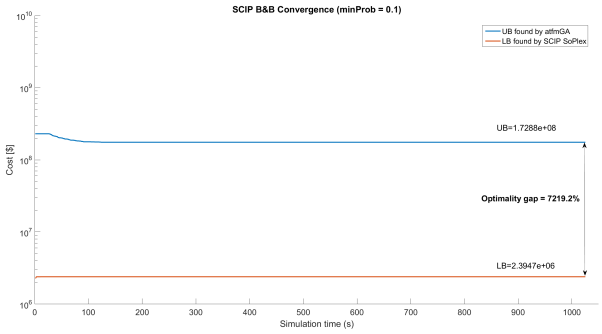


Fig. 15. Convergence result of SCIP B&B with GA for the ATFM conflicted instance `minProb`=0.1. (the y-axis is represented in a logarithmic scale to better visualize the convergence)

VII. Conclusions

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 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 [16] in which a GA was developed 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 overfitting 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 instance, 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 solution process. The Upper Bound found is 1.7288×10^8 \$.

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. 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.

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.

References

[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*, vol. 99, pp. 269–279, 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0360835216302601

[2] 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.

[3] EUROCONTROL, "Coda Digest 2015," pp. 1–34, 2016. [Online]. Available: http://www.eurocontrol.int/publications/coda-digest-2015

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

[5] FAA - Air Traffic Organization, *Traffic Flow Management in the National Airspace System*, 2009, no. October.

[6] P. R. Commission, "Evaluating the true cost to airlines of one minuete of airborne or ground delay," Eurocontrol, Tech. Rep. May, 2004.

[7] EUROCONTROL, "CODA DIGEST 2017," Tech. Rep., 2018.

[8] A. Cook, "European airline delay cost reference values," University of Westminster, London, Tech. Rep. December, 2015. [Online]. Available: http://westminsterresearch.wmin.ac.uk/19702/1/European airline delay cost reference values - updated and extended values %28V4.1%29.pdf

[9] 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.

[10] D. Bertsimas and S. S. Patterson, "The Air Traffic Flow Management Problem with Enroute Capacities," *Operation Research*, no. 1987, pp. 406–422, 1998.

[11] EUROCONTROL, *SESAR 2020 Concept Of Operations Step 1*, 01st ed., 2017.

[12] 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," p. 255, 2007.

[13] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[14] A. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 1st ed. New York: Springer-Verlag GmbH, 2003.

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

[16] J. Rios and J. Lohn, "A Comparison of Optimization Approaches for Nationwide Traffic Flow Management," *AIAA Guidance, Navigation, and Control Conference*, 2009.