Hybrid metaheuristic for air traffic management with uncertainty

Supatcha Chaimatanan, Daniel Delahaye, Marcel Mongeau

To cite this version:

Supatcha Chaimatanan, Daniel Delahaye, Marcel Mongeau. Hybrid metaheuristic for air traffic management with uncertainty. Lionel Amodeo, El-Ghazali Talbi, Farouk Yalaoui. Recent Developments in Metaheuristics, 62, Springer, pp.219-251, 2019, Lecture Notes in Electrical Engineering Part of the Operations Research/Computer Science Interfaces Series book series (ORCS), 978-3-319-58252-8. 10.1007/978-3-319-58253-514 . hal-01478322

HAL Id: hal-01478322
https://enac.hal.science/hal-01478322

Submitted on 25 Jul 2017

Hybrid metaheuristic for air traffic management with uncertainty

S. Chaimatanan, D. Delahaye, and M. Mongeau

ENAC, MAIAA, F-31055 Toulouse, France
Univ de Toulouse, IMT, F-31400 Toulouse, France
{supatcha,delahaye,mongeau}@recherche.enac.fr

Abstract

To sustain the rapidly increasing air traffic demand, the future air traffic management system will rely on a concept, called Trajectory-Based Operations (TBO), that will require aircraft to follow an assigned 4D trajectory (time-constrained trajectory) with high precision. TBO involves separating aircraft via strategic (long-term) trajectory deconfliction rather than the currently-practicing *tactical* (short-term) conflict resolution. In this context, this chapter presents a strategic trajectory planning approach aiming at minimizing the number of conflicts between aircraft trajectories for a given day. The proposed methodology allocates an alternative departure time, a horizontal flight path, and a flight level to each aircraft at a nation-wide scale.

In real-life situations, aircraft may arrive at a given position with some uncertainties on its curvilinear abscissa due to external events. To ensure robustness of the strategic trajectory plan, the aircraft arrival time to any given position will be represented here by a probabilistic distribution over its nominal assigned arrival time.

The proposed approach optimizes the 4D trajectory of each aircraft so as to minimize the probability of potential conflicts between trajectories. A hybrid-metaheuristic optimization algorithm has been developed to solve this large-scale mixed-variable optimization problem. The algorithm is implemented and tested with real air traffic data taking into account uncertainty over the French airspace for which a conflict-free and robust 4D trajectory plan is produced.

1 Introduction

This section provides a brief overview of the air traffic management system and the strategic trajectory planning problem.

1.1 Air traffic management: a brief review

Air traffic management (ATM) is a system that assists and guides aircraft from a departure aerodrome to a destination aerodrome in order to ensure its safety, while minimizing delays and airspace congestion. It manages the air traffic through the management of the three following complementary systems: airspace management (ASM), air traffic flow management (ATFM), and air traffic control (ATC).

The ASM organizes the usage of airspace. Its primary objective is to maximize the utilization of available airspace by segregating the airspace among various airspace user's needs in order to prevent interference from all users and to facilitates the flow of air traffic.

The ATFM manages the air traffic flow in order to minimize delays and prevent congestion. In Europe, this system is the concern of the Central Flow Management Unit (CFMU) of Eurocontrol. Every (non-military) operation flight performing under Instrument Flight Rules (IFR) in Europe must submit a flight plan to the CFMU. The CFMU then analyzes the compatibility of the request with the overall demand. If a request is not compatible with the airspace structure or the capacity limit, the CFMU will suggest alternative flight plan. It then distributes the accepted flight plan to all local air traffic control centers in Europe overflown by that particular flight.

This work has been supported by French National Research Agency (ANR) through JCJC program (project ATOMIC n° ANR 12-JS02-009-01).

The ATC then controls the air traffic in real time to ensure separation between aircraft. For this purpose, the airspace is partitioned into different airspace *sectors*, each of which is assigned to a specific group of controllers monitoring air traffic. Within each sector, a few minutes before the aircraft enters into the sector, the controllers are responsible for predicting conflicts. Then, the controllers are in charge of monitoring the traffic, maintaining aircraft separation by issuing instructions to pilots, and ensuring coordination with the neighboring sectors.

As mentioned above, in the current air traffic management system, an aircraft traveling between airports must register a flight plan in order to inform the relevant air navigation services. This flight plan includes the following information:

- Aircraft identification number, aircraft type, and navigation equipment installed on board;
- Departure airport;
- Proposed time of departure;
- Requested cruising altitude (*flight level*[1]);
- Requested route of flight;
- Cruising airspeed, climb and descent profiles, and speed schedules;
- Destination airport.

The ATC uses this information to predict the traffic situation. It issues necessary changes to the flight plan in order to ensure aircraft separation, and to maintain the order of air traffic flow, while satisfying as much as possible the pilot's request.

Current air traffic control regulations require aircraft that operate in the en-route environment up to FL[1]410 to be vertically separated by at least $N_v = 1,000$ feet (ft), and to be horizontally separated by a minimum of $N_h = 5$ nautical miles (Nm). For aircraft operating above FL 410, the required minimum vertical separation is increased to $N_v = 2,000$ ft. Aircraft are considered to be *in conflict* when such a *minimum separation* requirement is violated. This conflict situation does not necessary leads to a collision; however, it is a situation that controllers must avoid. One can consider that at any given time, each aircraft has a bounded and closed reserved block of airspace defined by a three-dimensional cylinder, as shown on Figure 1, in which other aircraft are not allowed to enter.

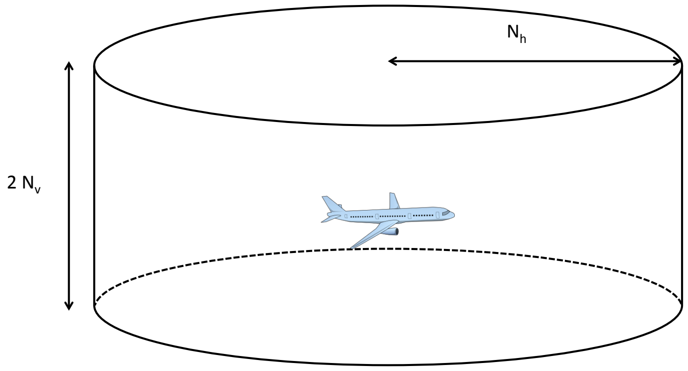

Fig. 1. The cylindrical *protection volume*

Because airspace, aircraft, ground systems, and human operators are limited resources which are very costly to extend, the usage of these resources has to be optimized through an effective planning. A good planning allows the ATM process to conform with the airspace user's requirements, and to be robust against unexpected events. Currently, the ATM process is performed through the following three planning phases:

- **Strategic planning.** This phase is performed from one year down to one week before real-time operations of the flights. This process aims at predicting the air traffic load, and at designing

[1] Flight level (FL) is a pressure altitude expressed in hundreds of feet, e.g. an altitude of 32,000 feet (ft) is referred to as FL 320.

the air-route structure in order to balance capacity and demand. During this long-term and medium-term planning phase, the air traffic is macroscopically organized.

- **Pre-tactical planning.** This phase takes place from six days down to one day before the real-time operations. The objectives are to optimize the overall ATM network performance, minimizing delay and cost by fine-tuning the strategic plan using more up-to-date information of expected traffic conditions, traffic demand, available capacity and weather forecast. During this phase, the air traffic flow is not only organized at macroscopic level, but also on each airplane. The takeoff slots[1] of each airplane is also managed.
- **Tactical planning.** This phase is carried out on the day of operations. Adjustments to the flight plans are performed based on the most up-to-date knowledge of the traffic situation and of the weather conditions. In this phase, individual aircraft departure slots are re-adjusted. Re-routings and alternative flight profiles can also be issued in order to avoid bottlenecks (congested sector) and to maximize airspace capacity according to real-time traffic demand. During this phase, the controller deals with the traffic inside a sector, and applies local changes to the aircraft trajectory in order to ensure aircraft separation within the corresponding sector.

1.2 Strategic aircraft trajectory planning

In order to accommodate the increasing air traffic demand in an already saturated airspace, the world's major ATM systems (e.g. European and U.S. ATM systems) are being modernized. The Next Generation air transportation system (NextGen) is a project aiming to transform the National Airspace System (NAS) of the United States towards a satellite-based air traffic management and control system. The Single European Sky ATM Research (SESAR) project is a major collaborative project aiming at modernizing the European air traffic management system. With the soon-coming technologies that will enable more powerful communication systems, more precise surveillance systems, and more reliance automated support tools, these new ATM systems will improve safety, reduce delay and aviation pollution emissions, while maximizing the use of airspace capacity.

The new ATM systems will rely on the concept of Trajectory Based Operations (TBO) which will focus more on adapting the airspace user's demand to the current airspace capacity. The conflict detection and resolution task load will be re-distributed to the strategic planning phase. In this new ATM paradigm, an aircraft flying through the airspace will be required to follow a negotiated *conflict-free* trajectory, accurately defined in four dimensions (three spatial dimensions and time). This will significantly reduce recourse to controller's intervention during the tactical phase, thereby enabling the controllers to manage a significant increase in traffic at any given time.

In this future ATM context, the aims of the strategic aircraft trajectory planning is to reduce the number of potential conflicts between trajectories. The objective of this chapter is to present a methodology to address such a strategic planning problem given a set of flight plans for a given day at a nation-wide scale. In real-life situations, aircraft may not be able to comply with the time constraint due to external events (wind, passenger delay, etc.). Moreover, imposing hard time constraints on the 4D trajectory may results in an increase of fuel consumption and aircraft engine workload, since the aircraft may have to adjust constantly its velocity. In order to improve robustness of the strategic trajectory plans and to relax the time constraints, uncertainties of aircraft arrival time to a given position is also taken into account in the trajectory optimization process.

More precisely, the given input of the strategic trajectory planning problem under consideration can be presented as follows:

- We consider a flight plan for a given day associated with a nation-wide scale airspace.
- The characteristics of the uncertainty of aircraft arrival time to any given position are given.
- For each flight, i, we suppose that the following elements are given:
  - a set of candidate routes;
  - a set of candidate flight levels;
  - a set of candidate departure times;

[1] A takeoff slot is an interval of time in which the take-off has to take place.

In the sequel, we shall often refer to flight i as *trajectory* i, or even *aircraft* i. The proposed strategic planning methodology consists of four main modules: a 4D-trajectory generator, a conflict-detection module, an interaction evaluation module, and a hybrid-metaheuristic optimization module (Figure 2). The 4D-trajectory generator is used to provide a 4D trajectory given an alternative route, an alternative flight level, and an alternative departure time. Then, the probabilistic conflict-detection module computes the probability of conflict involving a given 4D trajectory. After that, the interaction evaluation module will compute the level of interaction between trajectories at a nation-wide scale. The hybrid-metaheuristic optimization algorithm manages the search of an optimal set of alternative routes, alternative flight levels, and alternative departure times that minimize the potential conflicts (or the *interaction*-defined later) between trajectories.

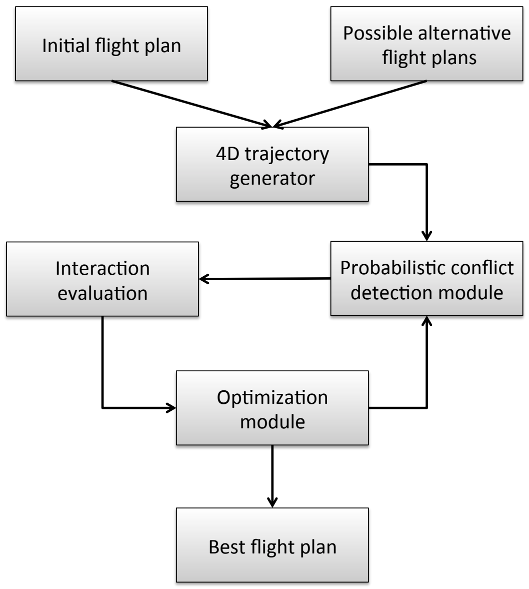

Fig. 2. Strategic trajectory planning procedure.

The optimal solution obtained is based on the following assumptions and simplifications:

– The airspace is considered as a Euclidean space. Latitudes and longitudes on the Earth's surface are transformed into (x, y) coordinates.
– The altitude, in feet, will be represented by the z coordinate.
– Each given initial route is a straight line from the departure airport to the destination airport.
– Aircraft speed is assumed to be changing only linearly between two consecutive sampling time steps.
– Uncertainty of aircraft arrival time does not grow with time.
– Wind conditions and weather forecast are not taken into account in the trajectory optimization process.

The remaining parts of this chapter are organized as follows. In Section 2, previous works related to air traffic management problems are discussed. Section 3 presents the mathematical model of the strategic trajectory planning problem. Section 4 presents a methodology to detect conflicts, and to compute conflict probability between aircraft trajectories. A hybrid-metaheuristic method designed to solve the strategic trajectory planning problem aiming at minimizing the total interaction between aircraft trajectories is presented in Section 5. Computational experiments with the proposed strategic trajectory planning methodology are presented in Section 6. Conclusions and perspectives are discussed in Section 7.

2 Previous related works

Over the last decades, numerous researches on the air traffic management problem have been conducted. We refer the reader interested by a survey on modeling and optimization in air traffic to the recent book [12]. A survey on mathematical optimization models for air traffic management problems based on different air traffic management strategies is provided in [1]. A comparison of different optimization methods (deterministic and metaheuristic optimization approaches) used for air traffic management is provided in [18].

In the strategic planning framework, aircraft trajectories can be separated in many different ways. One of the simplest and the most used method is to modify the departure time of aircraft. This is commonly referred to as *ground delay* or *ground holding*. The main idea of the ground holding strategy is to limit the number of airborne aircraft at any given time. Examples of works related to ground holding are [3], [19], and [24]. Delaying aircraft on the ground is effective since it prevents aircraft from flying extra distance to avoid congested areas or flying in a holding pattern around congested airport, which induce extra fuel consumption. However, with increasing air traffic demand, significant delays still have to be assigned to a large number of aircraft in order to meet all airspace-sector and airport capacity constraints. Besides, the ground holding strategy is more effective for the situation where congestion is likely to occur at the airports, which is not the case in Europe where most congestion occurs in the airspace sectors.

In [8] and [11], another idea to separate trajectories is presented based on speed regulations. Speed regulations introduce additional degrees of freedom to manage the flow of air traffic. However, it is effective at the fine-grain level which is irrelevant in the strategic trajectory planning context where there remain a high level of uncertainty. Furthermore, it requires numerous extensive and fine-tuned computations, which is not viable for a large-scale problem.

Other commonly-used strategies consider diverting the flight (re-routing), or modifying the flight levels, or a combination of the above-mentioned methods. To simplify the problem, several works rely on a flow-based air traffic model, where aircraft trajectories are grouped into several flows. For instance, [15] addresses large-scale (one day traffic over France) air traffic flow problems via a flow-based trajectory allocation, where the optimal separated 3D trajectory are obtained using an A* algorithm or using a genetic algorithm (GA) global search strategy . In [4], a ground holding is assigned to each aircraft and an optimal flight level is subsequently allocated to each flow of aircraft using constraint programming. Their results show that such rerouting and flight level re-allocation yield decrease in delays. In spite of the fact that the flow-based air traffic model has advantages in terms of reduced computation time, it cannot separate aircraft that belong to the same flow of trajectories.

To consider now each flight *individually*, air traffic flow models can can rely on a collection of subgraphs, whose the nodes represent the airports and waypoints overflown by each flight, and whose arcs connect the nodes for each flight. For instance, in [7], the authors show that the departure-time and alternative-route allocation problem is NP hard. Their optimal ground-holding times and alternative routes are obtained by solving a 0-1 integer model taking into account airspace sector capacity. Their models were implemented and tested with realistic datasets consisting of 2 to 6 airports. In [5, 6], integer optimization approaches are used in order to allocate ground delays and rerouting options to trajectories taking into account airspace sector capacity constraints. Thus, [5–7] propose improvements of air traffic at the airspace sector level but do not manage conflicts.

The authors of [2], introduce a mixed-integer programming model to minimize traveling time, operating/fuel cost, air/sound pollutions under separation and technical constraints. The optimal arcs and nodes (in a 3D-mesh network), speeds, and departure/arrival times for each flight are obtained by an exact deterministic method. However, the approach was tested on instances limited to problems involving 10 flights.

Reference [20, 21], the authors focus on managing each individual trajectory in large problems. Congestion in the airspace sectors is minimized by allocating to each flight optimal departure times and alternative routes (based on route-beacons navigation) using genetic algorithms (GA). Their results show that GA is very efficient in solving highly complex problems. Nevertheless, GA is not well adapted for the large-scale 4D trajectory planning problems that we are considering, due to excessive memory requirement intrinsic to population-based optimization algorithms whose performance depends on the size of population.

In the future ATM context, aircraft trajectory can be represented by a time sequence of 4D coordinates. In [9, 10], preliminary studies on the optimization of individual 4D trajectories are presented. In these papers, optimal (conflict-free) 4D trajectories for individual flights are allocated by solving a combinatorial optimization problem using a non-population-based hybrid-metaheuristic optimization method. The numerical results presented in [10] show advantages of the hybrid-metaheuristic optimization approach on ATFM problems. However, the discretization of the search domain (candidate departure times and trajectories) induces high combinatorics.

Uncertainties of aircraft position were taken into account in the conflict detection and resolution problem addressed in the work presented, for example, in [14,16,22]. Aircraft positions are modeled as a probabilistic distribution, then the predicted aircraft positions are computed over a certain time window using a dynamic model of aircraft, and conflict probabilities are evaluated. These methods are suitable for mid-term and short-term conflict-detection and resolution problems involving a small number of aircraft. However, they are not suitable for the large-scale problems that we are attempting to address in this work, due to the heavy computational burden implicated in predicting aircraft positions using aircraft dynamic models. We refer the reader interested by a review of conflict-detection and resolution modeling methods to [17].

In this chapter, we put forward the work presented in [10], by relaxing the solution space and proposing an alternative, mixed-integer programming formulation of the problem. Moreover, we introduce a methodology to take into account uncertainty of aircraft trajectories in the strategic trajectory planning problem. We also propose new intensification local-search steps, and we describe a computationally-efficient hash-table based method for detecting and evaluating probabilities of conflict between trajectories. Finally, we prove the viability of the overall methodology on large-scale air traffic data on the French airspace.

3 Mathematical model

This section set the mathematical framework of the proposed strategic trajectory planning methodology. First, the assumed uncertainty on aircraft trajectory is characterized. Then, methods that are used to separate the aircraft trajectories are described. Finally, a concept of interaction between trajectories, and a mathematical formulation of the strategic trajectory planning problem under the form of a mixed-integer optimization problem are presented.

3.1 Uncertainty

Conflict detection methods can be roughly classified into three categories [16]: nominal, worst-case, and probabilistic conflict detections, according to the assumptions made on the predicted aircraft trajectory. The *nominal* conflict detection does not take into account deviation of aircraft from its assigned (nominal) trajectory. The *worst-case* conflict detection identifies the conflict as a situation in which the distance between the *envelopes* of the predicted trajectories (the set of all possible trajectories) is less than the minimum separation requirements. The *probabilistic* conflict detection method involves computing probability of conflict between aircraft whose trajectories are described with probability density functions. In other words, it computes the probability that two aircraft will penetrate into the (cylindrical) protection volume of one another. It is suitable for assessing the air traffic condition in a large-scale traffic scenario with high level of uncertainties, for example in strategic trajectory planning.

We shall use in the remaining of this chapter the notation $P_i = (x_{P_i}, y_{P_i}, z_{P_i}, t_{P_i})$ to designate a 4D point on trajectory i. We shall call its forth coordinate, t_{P_i}, the *assigned arrival time* of aircraft i at the point $(x_{P_i}, y_{P_i}, z_{P_i})$. Let us consider two trajectories, A and B, and let us first consider the case where time uncertainty in *not* taken into account. In the absence of time uncertainty, when the horizontal separation, $d_h = \sqrt{(x_{P_A} - x_{P_B})^2 + (y_{P_A} - y_{P_B})^2}$, is less than 5 Nm, and when the vertical separation, $d_v = |z_{P_A} - z_{P_B}|$, is less than 1,000 ft, the arrival times of both aircraft must be separated in time, i.e. $d_t = |t_{P_A} - t_{P_B}|$, strictly greater than zero.

In general, an aircraft is able to follow a given flight profile with very high accuracy thanks to the flight management system (FMS)[1]. The residue uncertainty of aircraft position is more likely

[1] Flight management system (FMS) is an on-board computer system that determines the aircraft exact position and calculates the lateral and horizontal guidance for the aircraft

to occur in the time domain. The aircraft may arrive at a given position with a time error due to, for example, wind conditions, external temperature, aircraft weight estimation errors, passenger delay, etc.

Consider now the case with time uncertainty: let t_ϵ be the *maximum time error* (defined by the user). The *predicted* arrival time of an aircraft at a position P under uncertainty therefore lies in the interval:

$$[t_P - t_\epsilon, t_P + t_\epsilon].$$

For the purpose of potential conflict detection, we assume that the predicted aircraft arrival time can be modeled as a random variable with the following triangular distribution defined over the interval $[t_P - t_\epsilon, t_P + t_\epsilon]$. Given a lower limit $t_P - t_\epsilon$, an upper limit $t_P + t_\epsilon$, the *predicted* arrival time, $\hat{t}_P$, to the position P is given by the probability density function:

$$\hat{t}_P(t) = \mathcal{T}_{P,t_\epsilon}(t),$$

where $\mathcal{T}_{P,t_\epsilon}(t)$ denotes the triangular distribution:

$$\mathcal{T}_{P,t_\epsilon}(t) = \begin{cases} 0 & \text{for } t < t_P - t_\epsilon, \\ \frac{(t - t_P + t_\epsilon)}{t_\epsilon^2} & \text{for } t_P - t_\epsilon \le t \le t_P, \\ \frac{(t_P + t_\epsilon - t)}{t_\epsilon^2} & \text{for } t_P < t \le t_P + t_\epsilon, \\ 0 & \text{for } t_P + t_\epsilon < t. \end{cases}$$

To explain the process to detect conflicts between two aircraft trajectories in our triangular distribution case, let us first consider two trajectories A and B illustrated in Figure 3. For simplicity, let us first assume that trajectories A and B are defined by continuous functions, and let P_A and P_B be a pair of any points on the trajectories A and B respectively. To identify conflict between these two trajectories, we must check the minimum separation between all possible pairs of points P_A and P_B (pair-wise comparison). The predicted arrival time, $\hat{t}_{P_A}$, of aircraft A to the given point

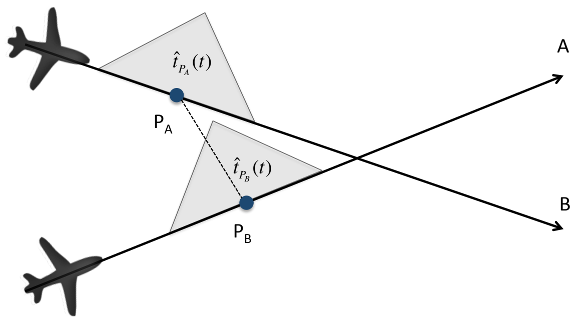

Fig. 3. Measuring conflict between two continuous trajectories A and B.

P_A, and the *predicted* arrival time, $\hat{t}_{P_B}$, to the given point P_B are given by:

$$\hat{t}_{P_A}(t) = \mathcal{T}_{t_{P_A},t_\epsilon}(t),$$

and

$$\hat{t}_{P_B}(t) = \mathcal{T}_{t_{P_B},t_\epsilon}(t).$$

A *potential conflict* between trajectories A and B occurs when there exists a pair of points, P_A and P_B, from each trajectory such that $d_h = \sqrt{(x_{P_A} - x_{P_B})^2 + (y_{P_A} - y_{P_B})^2}$ is less than 5 Nm, $d_v = |z_{P_A} - z_{P_B}|$ is less than 1,000 ft, and the intersection between intervals $[t_{P_A} - t_\epsilon, t_{P_A} + t_\epsilon]$ and $[t_{P_B} - t_\epsilon, t_{P_B} + t_\epsilon]$ is not empty. The *conflict probability*, denoted $\mathcal{P}_C(P_A, P_B)$, between the point P_A and point P_B can be computed from:

$$\mathcal{P}_C(P_A, P_B) = \int_{t_{start}}^{t_{end}} \hat{t}_{P_A} \hat{t}_{P_B} dt, \tag{1}$$

where t_{start} and t_{end} are respectively the lower and upper bounds of the interval $[t_{P_A} - t_\epsilon, t_{P_A} + t_\epsilon] \cap [t_{P_B} - t_\epsilon, t_{P_B} + t_\epsilon]$.

To implement our conflict detection algorithm, we discretize the 4D trajectories. The sampling time step, t_s, must be set (by the user) sufficiently small to guarantee that any conflict occurring between two consecutive sampling steps will be detected. Doing so, trajectories A and B can be represented respectively, by the time sequences of 4D coordinates $\{P_{A,k_A}\}_{k_A=1}^{K_A}$, and $\{P_{B,k_B}\}_{k_B=1}^{K_B}$, where K_A and K_B are the number of sampling points corresponding to trajectories A and B respectively (see Figure 4). In order to detect the conflicts between the two trajectories, we must verify the minimum separation constraint between every possible pair of sampled points P_{A,k_A} and P_{B,k_B}.

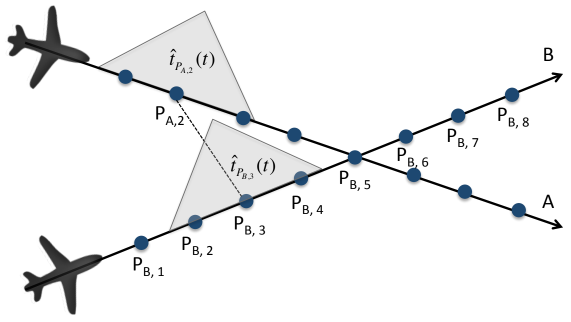

Fig. 4. Measuring conflict between two discretized trajectories A and B.

The conflict probability, $\mathcal{P}_C(P_{A,k_A}, P_{B,k_B})$, associated to every pair of sample points P_{A,k_A} and P_{B,k_B} of trajectories A and B. This can be computed using equation 1. However, this pair-wise comparison is time consuming. It requires prohibitive time in a large-scale application context as the one considered in this study. A fast algorithm to detect such a probabilistic violation of the minimum separation requirements (i.e . to compute the conflict probabilities associated to all pairs of sampled trajectory points between large-scale aircraft trajectories) will be presented in Section 4.

3.2 Trajectory separation methods

In this subsection, we describe the three possible trajectory separation methods we are considering in order to avoid conflicts:

- shifting the departure time,
- changing the flight level,
- modifying the route (horizontal flight profile).

The alternative departure time, alternative flight level, and alternative route to be allocated to each flight are modeled as follows.

Alternative departure time. The departure time of each flight, i, can be shifted by a positive (delay) or a negative (advance) time shift denote δ_i. The departure time, t_i, of flight i is therefore

$$t_i = t_{i,0} + \delta_i,$$

where $t_{i,0}$ is the initially-planned departure time of flight i. Following common practice in airports, the set of possible values for δ_i will be discrete.

Alternative flight level. To separate the trajectories in the vertical plane, we define another decision variable associated to each flight i: a flight level shift $l_i \in \mathbb{Z}$. Therefore, the flight level of each flight i is given by:

$$FL_i = FL_{i,0} + l_i,$$

where $FL_{i,0}$ is the initially-planned flight level of flight i. Figure 5 shows a trajectory with two alternative flight levels.

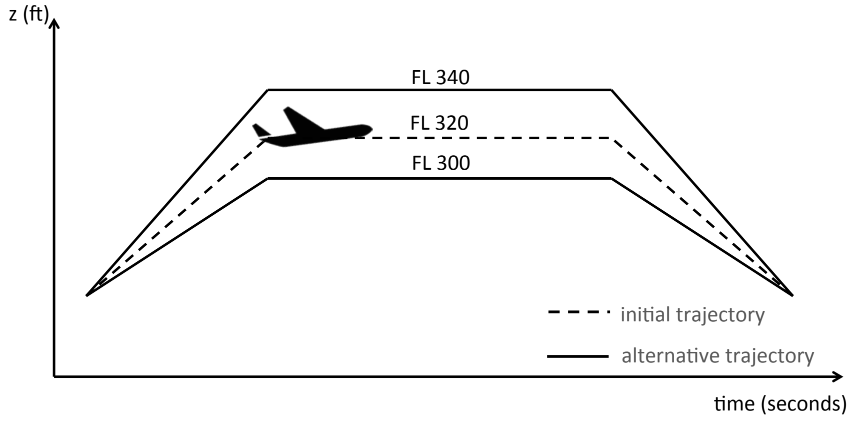

Fig. 5. Two alternative vertical profiles of a trajectory (two alternative flight levels).

Alternative route design. An alternative route should not deviate too much from the nominal route. It should also be computed in a short computation time. To generate an alternative route, we modify the given initial *horizontal flight profile* of a trajectory, i, by placing a set of virtual waypoints near the initial horizontal flight profile of flight i, and then by reconnecting the successive waypoints with straight-line segments.

We call *longitudinal axis* (x') the axis that is tangent to the initial en-route segment, and the *lateral axis* (y') is the axis that is perpendicular to the longitudinal axis. The position of each waypoint will be defined using these relative $x'y'$-reference axes.

We define, for each flight i, a vector, w_i, of virtual waypoints (optimization variables) used to control the trajectory shape of flight i: $w_i = (w_i^1, w_i^2, \ldots, w_i^m)$, where M denotes the number of virtual waypoints that the user is allowed to introduce, and where $w_i^m = (w_{ix'}^m, w_{iy'}^m)$ is the m^{th} virtual waypoint of trajectory i, where $w_{ix'}^m$ and $w_{iy'}^m$ are the longitudinal and lateral components of w_i^m respectively. Figure 6, illustrates possible alternative horizontal profiles for a given trajectory constructed with $M = 2$ virtual waypoints.

Remark that such an alternative trajectory is likely to yield an increase in flight duration when compared with the initial trajectory. To compensate this increased flight duration, the altitude profile will be updated to avoid a premature descent. Let T_{ext} be the increased flight duration of flight i. In the case of a regional flight, whose flight phases are all carried out in the same (current) airspace sector, the altitude profile is updated by extending the cruise phase at the top of descent for a duration of T_{ext} as illustrated in Figure 7.

On the other hand, for a flight whose origin or destination airports are outside of the current airspace, the top of descent of such flight may not be in the current airspace sector. Therefore, we update the altitude profile by extending the flight at maximum altitude (in the current airspace) for a duration T_{ext}. In this case, the vertical profile is updated according to six possible cases according to whether the origin/destination airports are in the current airspace or not and to whether the initial trajectory has a cruise (constant-level) phase or not, as illustrated in Figure 8.

3.3 Optimization formulation

In this subsection, we present an optimization formulation of the strategic 4D trajectory planning problem. The strategic 4D trajectory planning methodology using route / flight-level / departure-time allocation can be formulated as an optimization problem attempting at minimizing the interaction between trajectories.

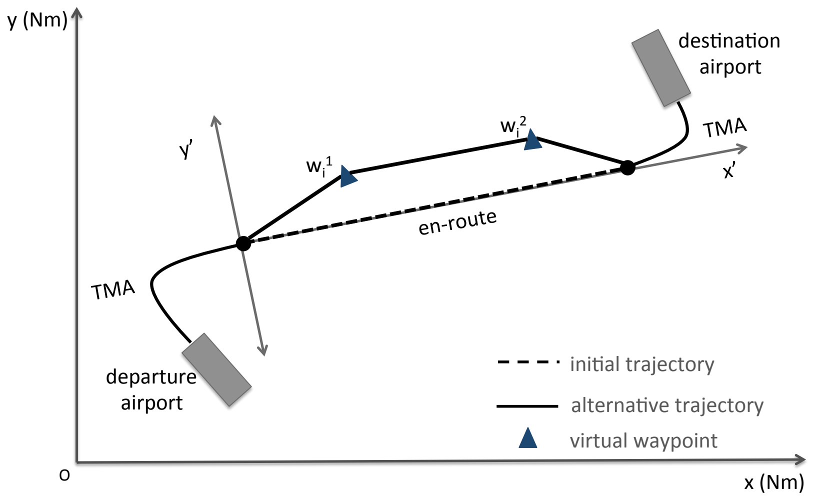

Fig. 6. An alternative horizontal profiles for a given trajectory, i, constructed with $M = 2$ virtual waypoints.

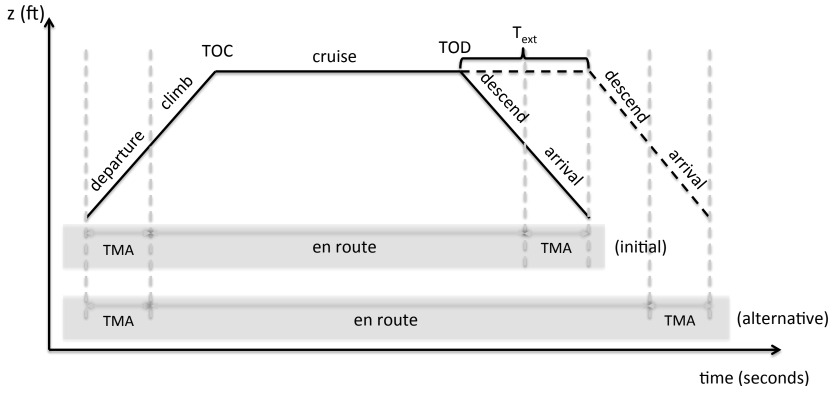

Fig. 7. Altitude-profile update: extending cruise phase at the top of descent (TOD).

Given data: A problem instance is given by:

– A set of N initial (nominal) discretized 4D trajectories;
– The maximal time error, t_ϵ;
– The sampling time step: t_s;
– The interpolation sampling time step: t_{interp};
– The number of allowed virtual waypoints: M;
– The discretization time step for the possible delay / advance departure-time shift interval: δ_s;
– For each flight i, for $i = 1, \ldots, N$:
  • The initial planned departure time: $t_{i,0}$;
  • The maximum allowed advance departure time shift: $\delta_a^i < 0$;
  • The maximum allowed delay departure time shift: $\delta_d^i > 0$;
  • The initial planned flight level: $FL_{i,0}$;
  • The maximum allowed flight level shift: $l_{i,max}$;
  • The length of the initial planned route: $L_{i,0}$;
  • The maximum allowed route length extension coefficient: $0 \leq d_i \leq 1$;
  • The user-defined parameters controlling the dimensions of the feasible domains for placing the virtual waypoints: a_i and b_i.

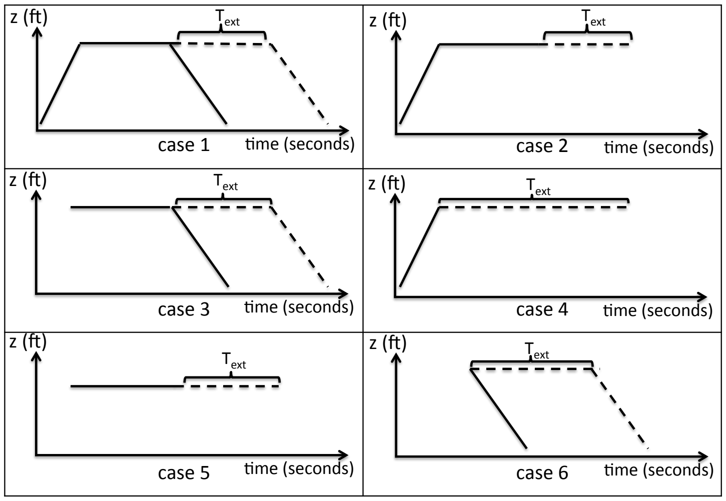

Fig. 8. Altitude profile update: six possible ways to extend the trajectory at maximum altitude.

Decision variables: As mentioned above, we consider three ways to separate trajectories. In the time domain, one can use a departure-time shift δ_i is associated to each flight, i. In the 3D space, one can relay on a vector, w_i, of virtual waypoint locations, $w_i = (w_i^1, w_i^2, \ldots, w_i^M)$ associated to each flight, i, where M is the number of virtual waypoints. Finally, in the vertical plane, a flight-level shift, l_i may be applied to each flight i.

Let us set the compact vector notation:

$$\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_N),$$
$$\mathbf{l} = (l_1, l_2, \ldots, l_N).$$

and

$$\mathbf{w} = (w_1, w_2, \ldots, w_N).$$

Therefore, the decision variables of our route / departure-time allocation problem can be represented by the vector:

$$u := (\boldsymbol{\delta}, \mathbf{l}, \mathbf{w}).$$

We shall denote by u_i the components of u. It is a vector whose components are related to the modification of the i^{th} trajectory, thereby:

$$u_i := (\delta_i, l_i, w_i)$$

Constraints: The above optimization variables must satisfy the following constraints:

Allowed departure time shift. The departure time of flight i is given by an auxiliary optimization variable, t_i, which is directly linked to the above decision variables as follows:

$$t_i = t_{i,0} + \delta_i,$$

where $t_{i,0}$ is the initial planned departure time of flight i.

In practical problems, passengers may have to transfer from one flight to another in order to get to their final destination. This generates precedence constraints stipulating that certain flights must arrive at the airport before the departure of others. In addition, each aircraft may fly several flights a day. This raises a constraint of minimum *rotation time* between flights (time required to disembark the passengers, to service the aircraft, and to embark passengers for the next flight). These constraints are not taken into account in this work; however they can easily be handled by pre-processing the set of feasible time shifts of each flight.

In order to prevent excessive delay (or advance) of departure time, the departure-time shift δ_i is limited to lie in the interval

$$[\delta_a^i, \delta_d^i].$$

However, common practice in airports conducted us to rely on a discretization of this time interval. Given a user-defined departure time shift step-size δ_s (to be set by the user), this yields $N_a^i := \frac{-\delta_a^i}{\delta_s}$ possible advance slots and $N_d^i := \frac{\delta_d^i}{\delta_s}$ possible delay slots of flight i. Parameters are to be set by the user so that both δ_a^i and δ_d^i are multiples of δ_s. Therefore, the set, Δ_i, of all possible departure time shifts of flight i:y

$$\Delta_i := \{-N_a^i.\delta_s, -(N_a^i - 1).\delta_s, \ldots, -\delta_s, 0, \delta_s, \ldots, (N_d^i - 1).\delta_s, N_d^i.\delta_s\}. \tag{2}$$

Maximum allowed flight-level changes. In order to limit the change of flight levels, the flight level shift is also bounded. The set, ΔFL_i, of all possible flight-level shifts of flight i is given by:

$$\Delta FL_i = [FL_{i,0} - l_{i,max}, \ldots, FL_{i,0} - 1, 0, FL_{i,0} + 1, \ldots, FL_{i,0} + l_{i,max}]. \tag{3}$$

Maximal route length extension. The alternative trajectory induces route length extension which causes an increase of fuel consumption. Therefore, it should be limited so that it remains acceptable by the airline. Let $0 \leq d_i \leq 1$ be the maximum allowed route length extension coefficient of flight i (to be set by the user). The alternative en-route profile of flight i must satisfy:

$$L_i(w_i) \leq (1 + d_i)L_{i,0}, \tag{4}$$

where $L_i(w_i)$ is the length of the alternative en-route profile determined by w_i. This constraint can be satisfied a priori simply by restricting the set of possible waypoint locations (as will be described below).

Allowed waypoint locations. To limit the search space, to prevent undesirable sharp turns, and to restrain the route length extension, we bound the possible location of each virtual waypoint. For simplicity, for each trajectory i, and for each waypoint w_i^m, its longitudinal component, $w_{ix'}^m$, is set to lie in the interval:

$$W_{ix'}^m := \left[\left(\frac{m}{1+M} - b_i\right) L_{i,0}, \left(\frac{m}{1+M} + b_i\right) L_{i,0}\right], \tag{5}$$

for $m = 1, \ldots, M$; and $i = 1, \ldots, N$, and where $b_i \geq 0$ is a user-defined coefficient controlling the length of the interval $W_{ix'}^m$.

Furthermore, to avoid sharp turns, the longitudinal position of the virtual waypoints should not be too close to each other. To obtain a regular trajectory, the longitudinal component of two adjacent waypoints must not overlap, i.e.

$$\left(\frac{m}{1+M} + b_i\right) < \left(\frac{m+1}{1+M} - b_i\right),$$

and hence the user should choose parameter b_i so that

$$b_i < \frac{1}{2(M+1)}.$$

Similarly, the lateral component, $w_{iy'}^m$, is restricted to lie in the interval:

$$W_{iy'}^m := [-a_i.L_{i,0}, a_i.L_{i,0}], \tag{6}$$

where $a_i \leq 0$ is a user-defined coefficient. The box-size parameters $(a_i, b_i)_{i=1}^N$ should be chosen so that the maximal route length extension constraint (4) is satisfied for all possible locations of the m waypoints in the 2D boxes $\{W_{ix'}^m \times W_{iy'}^m\}_{m=1}^M$, for every trajectory i ($i = 1, 2, \ldots, N$). Figure 9 illustrates for a trajectory i the 2D boxes of possible locations for $M = 2$ virtual waypoints, an example of positions for w_i^1 and w_i^2, and the resulting alternative trajectory. More precisely, for each trajectory i, a_i and b_i must be chosen a priori so that:

$$\max_{w_i}\{L_i(w_i) | w_i \in W_{ix'} \times W_{iy'}\} \leq (1 + d_i)L_{i,0}$$

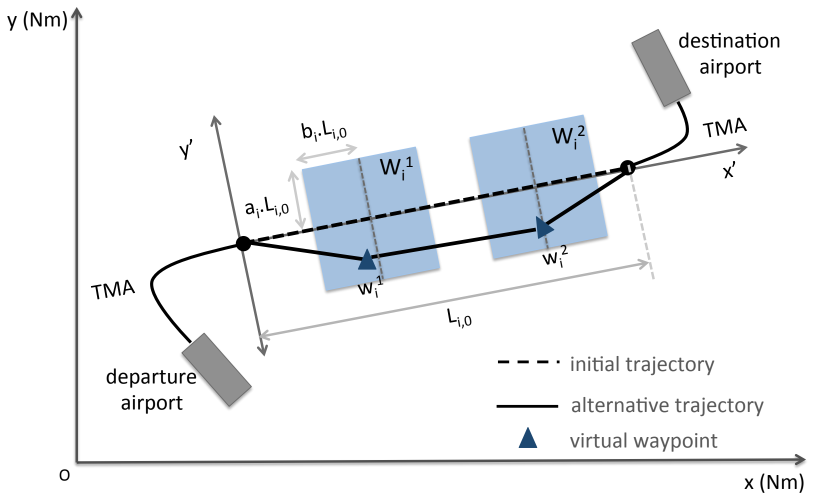

Fig. 9. 2D boxes of possible locations for $M = 2$ virtual waypoints, a proposition of location of the virtual waypoints, and the corresponding alternative trajectory.

Objective Function. In the *strategic* trajectory planning, where uncertainty is too large to fine-tune the trajectories, we focus on separating roughly aircraft trajectories rather than on solving precisely each conflict locally. Therefore, we introduce here the concept of *interaction between trajectories* to define a situation that occurs in the planning phase, when more than one trajectory compete for the "same space" at the "same period of time".

For given values of the decision variables $u = (\boldsymbol{\delta}, \mathbf{l}, \mathbf{w})$, one must first discretize each of the N resulting alternative trajectories into a sequence of 4D points: $\{P_{i,k}(u_i)\}_{k=1}^{K_i}$, $i = 1, 2, \ldots, N$. Each of these points depends only on the i^{th} component of u.

Let us define an *interaction at a point* $P_{i,k}(u_i)$ to be the sum of all the conflict probabilities associated to point $P_{i,k}(u_i)$; we denote it $\Phi_{i,k}(u)$. Remark that it depends also on the other trajectories $j \neq i$. Hence,

$$\Phi_{i,k}(u) = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{k=1}^{K_i} \sum_{l=1}^{K_j} \mathcal{P}_C(P_{i,k}(u_i) P_{j,l}(u_j)),$$

where K_i and K_j are the number of sampling points for trajectory i and j respectively.

The *interaction associated with trajectory* i, denoted Φ_i(u), is defined as follows:

$$\Phi_i(u) = \sum_{k=1}^{K_i} \Phi_{i,k}(u).$$

Finally, the *total interaction between trajectories*, $\Phi_{tot}(u)$, for a whole N-aircraft traffic situation is simply defined as:

$$\Phi_{tot}(u) = \sum_{i=1}^{N} \Phi_i(u) = \sum_{i=1}^{N} \sum_{k=1}^{K_i} \Phi_{i,k}(u). \tag{7}$$

One wishes to determine values for the optimization variables δ_i, l_i, and w_i for each flight $i = 1, 2, \ldots, N$ so as to minimize the total interaction, $\Phi_{tot}(u)$, between the N given trajectories.

To summarize, the strategic trajectory planning problem with uncertainty can be represented by an interaction minimization problem formulated as a mixed-integer optimization problem as follows:

$$\begin{aligned}
&\min_u \Phi_{tot}(u) \\
\text{subject to}\quad &\\
&\delta_i \in \Delta_i, \quad i = 1, 2, \ldots, N \qquad\qquad \text{(P1)}\\
&l_i \in \Delta FL_i, \quad i = 1, 2, \ldots, N \\
&w_i^m \in W_{ix'}^m \times W_{iy'}^m, \quad m = 1, 2, \ldots, M,\ i = 1, 2, \ldots, N,
\end{aligned}$$

where $\Phi_{tot}(u)$ is defined by (7), and Δ_i, ΔFL_i, $W_{ix'}^m$, and $W_{iy'}^m$ are defined by (2), (3), (5), and (6) respectively.

The optimization formulation (P1) involves mixed-integer variables introducing high combinatorics to the search space. We have $W_{ix'}^m \times W_{iy'}^m \subseteq \mathbb{R}^2$ for $m = 1, 2, \ldots, M$ and $i = 1, 2, \ldots, N$, and therefore $\mathbf{w} \in \mathbb{R}^{2MN}$. Each discrete departure-time shift-variable (δ_i) feasible set has cardinality $|\Delta_i| = \frac{|\delta_a^i| + |\delta_d^i|}{\delta_s} + 1$, which implies $\boldsymbol{\delta} \in \mathbb{Z}^{(N\frac{|\delta_a^i|+|\delta_d^i|}{\delta_s}+1)}$. Finally, each discrete flight-level shift variable (l_i) feasible set has cardinality $2l_{i,max} + 1$; therefore we have $\mathbf{l} \in \mathbb{Z}^{N(2l_{i,max}+1)}$.

We emphasize the fact that one evaluation of the objective function Φ_{tot} for one proposition of the decision variables $u = (\boldsymbol{\delta}, \mathbf{l}, \mathbf{w})$ involves discretizing each of the N resulting candidate trajectories. Remark also that, the objective function of problem (P1) is non-separable, because each term $\Phi_{i,k}(u)$ does not depend solely on the variable u_i; it is also affected by neighboring trajectories. The evaluation of the objective function involves a heavy computational burden in practice, as this will be seen in the sequel of this chapter where we consider a real-world problem at the nation-wide scale. Besides, the objective function may feature several local optima (the objective function may easily be shown to be multimodal). This route / flight-level / departure-time assignment problem with uncertainty is therefore sufficiently difficult to motivate recourse to a stochastic method of optimization.

4 Interaction detection module

To evaluate the objective function, Φ_{tot}, at a candidate solution, u, one needs to compute the interaction at each possible pair of sampled trajectory points involved in N aircraft trajectories. To avoid the exhaustive $\frac{N(N-1)}{2}$ time-consuming pair-wise comparisons, which is prohibitive in our large-scale application context, we propose the following grid-based interaction-detection scheme.

First, we define a four-dimensional (3D space + time) *grid*. The dimension of this 4D grid must be large enough to include the N given trajectories (and all its possible modifications through our decision variables). For instance, the time dimension of the grid must span enough to include the earliest and the latest flights on a given operational day taking into account all candidate departure-time shift options.

The 4D grid is partitioned into *cells* (see Figure 10). To detect interactions, the idea is to store the N trajectories in each corresponding cell in the 4D grid. Then for each trajectory i, and for each cell (I_x, I_y, I_z, I_t) corresponding to each sampling point $P_{i,k} := (x_{P_{i,k}}, y_{P_{i,k}}, z_{P_{i,k}}, t_{P_{i,k}})$, we simply need to check all the surrounding cells corresponding to the time period $[t_{P_{i,k}} - 2t_\epsilon, P_{i,k} + 2t_\epsilon]$. If these neighboring cells are occupied by other aircraft, for instance j, we then note $j \in (I_x, I_y, I_z, I_t)$, and then the conflict probabilities between point $P_{i,k}$ and the sample point corresponding to those aircraft are computed (otherwise, it is null).

In order to optimize the required computation memory, we implement the interaction-detection scheme using a so-called *harsh table*, which is a data structure that maps *keys* to values or *entries*. It allows us to store information in an array without the need to define a priori the size of the array. Moreover, the hash table only stores data as it is created; therefore it does not use memory for the (very numerous) empty cells in the array.

In order not to underestimate interaction (missing the loss of spatial separation occurring between two successive sampling time steps), trajectories must be discretized with a sufficiently-small sampling time step, t_s, which depends on the maximum possible aircraft horizontal and vertical speeds. Figure 11 illustrates an undetected violation of the horizontal minimum separation occurring between two successive time steps. As stated in [4], the worst-case scenario for interaction detection in the horizontal plane occurs when two aircraft follow parallel trajectories that are separated by a distance, D, less than or equal to the horizontal separation norm, $N_h = 5$ Nm, at

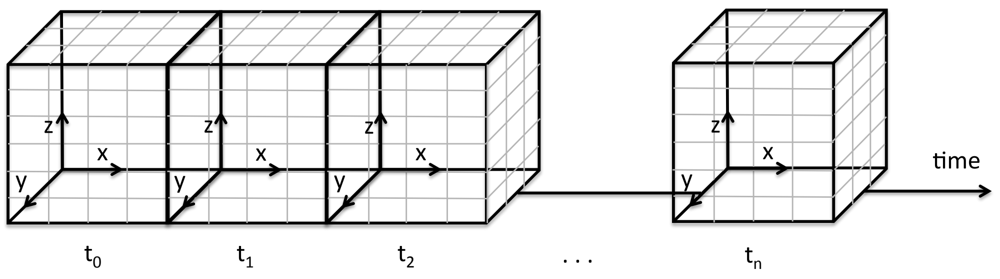

Fig. 10. Four dimension (space - time) grid.

maximum horizontal speed, $V_{h_{max}}$, with headings in opposite directions. Hence, in the horizontal plane, undetected interaction can occur when:

$$t_s > \frac{N_h}{V_{h_{max}}} \cos\left(\arcsin\left(\frac{D}{N_h}\right)\right).$$

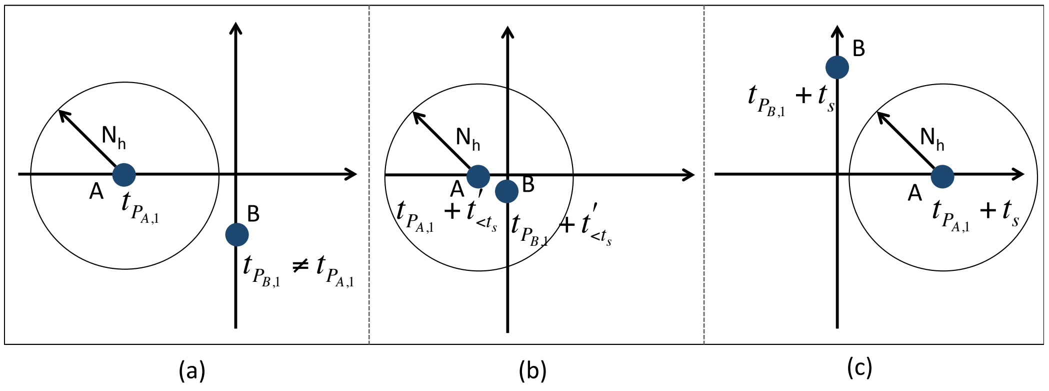

Fig. 11. Undetected violation of horizontal separation when the sampling step, t_s, is too large. In figures (a) and (c) the horizontal spatial distances between A and B are larger than $N_h = 5$ Nm, therefore the violation of the horizontal minimum separation, which occurs in the meantime (figure (b)) cannot be detected.

In the vertical plane, the worst-case scenario occurs when one aircraft is climbing at a maximum rate of climb, RoC_{max}, and another is descending at maximum rate of descent, RoD_{max} (see Figure 12). Thus, in the vertical plane, in an analogical way as what was done in [4] for the horizontal plane, we can easily show that undetected interaction can occur when:

$$t_s > \frac{N_v}{(RoC_{max} + RoD_{max})},$$

where $N_v := 1000$ ft. is the vertical separation norm.

In order to avoid such undetected conflicts, one can therefore simply choose a sufficiently small value for the (user-provided) sampling time step, t_s. However, using too small sampling time step leads to a large number of trajectory sample points, which in turn requires more computation time and memory. Instead, we propose an *inner-loop* algorithm, called interp, detecting the violation of minimum separation requirements between two sampling times, t and $t + t_s$, by *interpolating* aircraft positions with a sufficiently small interpolation step size, t_{interp}. This t_{interp} value must be set by the user so as to guarantee that no interaction remains undetected. Then, one checks

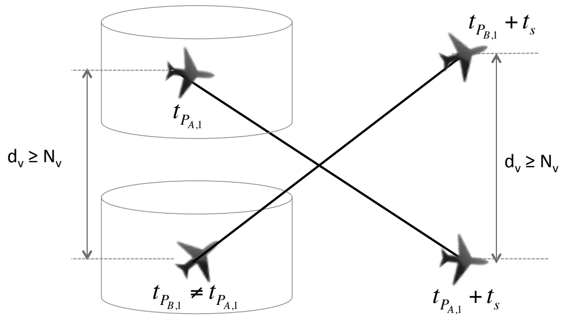

Fig. 12. Undetected violation of the vertical minimum separation between two aircraft when the sampling step, t_s, is too large.

each pair of these interpolated points. The algorithm stops when a violation of the minimum separation requirements is identified or when every pair of points have been checked. The inner-loop interpolation algorithm called `interp` is described in Figure 13. The algorithm to compute the total interaction between the N trajectories, Φ_{tot}(u), is described in detail in Figure 14.

```
Algorithm Interp
Require: $P_{i,k}$, $P_{j,l}$
1: Discretize, using time step $t_{interp}$, the trajectory segment $[P_{i,k}, P_{i,k+1}]$ and $[P_{j,l}, P_{j,l+1}]$
   as $\{P_\alpha\}_{\alpha=1}^{K}$ and $\{Q_\beta\}_{\beta=1}^{K}$ respectively;
2: for $k = 0 \to K$ do                    ▷ for each pair of interpolated points
3:     Initialize $\mathcal{P}_C := 0$;
4:     Compute conflict probability, $\mathcal{P}_C := \mathcal{P}_C(P_k, Q_k)$ using (1);
5:     if $\mathcal{P}_C > 0$ then
6:         Return $\mathcal{P}_C$;
7:         End;
8:     end if
9: end for
10: Return $\mathcal{P}_C$;
```

Fig. 13. Inner-loop interpolation algorithm.

5 Hybrid-metaheuristic for strategic trajectory planning

In our country-wide air traffic scale application context, the evaluation of the objective function value relies on a black-box simulation through the interaction detection scheme introduced in the previous section. For such a large-scale problem, this simulation requires a very large computation memory. Therefore, to solve the interaction minimization problem (P1), we rely on a non-population based hybrid-metaheuristic optimization method that combines the advantages of simulated annealing and of an iterative-improvement local search.

After presenting brief overviews of simulated annealing and iterative-improvement local search, this section presents the optimization methodology we are proposing.

```
Algorithm Interaction detection
Require: value of the decision variables $u = (\boldsymbol{\delta}, \mathbf{l}, \mathbf{w})$ Initialize $\Phi_{tot} := 0$;
  for i = 1 to N do                                          ▷ (for each trajectory $i$)
    Discretize the alternate trajectory $i$ defined by $u_i$ into a sequence $\{P_{i,k}\}_{k=1}^{K_i}$;
    Initialize $\Phi_i = 0$;
    for k = 1 to $K_i$ do                         ▷ (for each point $P_{i,k}$ of trajectory $i$)
      Initialize $\Phi_{i,k} := 0$;
      Compute the cell $I_x, I_y, I_z, I_t$ corresponding to $P_{i,k}$;
      Compute $\Phi_{i,k}(u)$:
      for $i_x = I_x - 1$ to $I_x + 1$ do
        for $i_y = I_y - 1$ to $I_y + 1$ do
          for $i_z = I_z - 1$ to $I_z + 1$ do
            for $i_t = I_t - \frac{2t_\epsilon}{t_s}$ to $I_t + \frac{2t_\epsilon}{t_s}$ do
              if $\exists j \neq i$ such that $j \in (i_x, i_y, i_z, i_t)$ then
                $L$:= list of all trajectory sample point $P_j$ in $(i_x, i_y, i_z, i_t)$;
                for $l = 1$ to length($L$) do
                  $P := L(l)$;
                  compute conflict probability, $\mathcal{P}_C = \mathcal{P}_C(P_{i,k}, P)$ using (1);
                  if $\mathcal{P}_C = 0$ then
                    $\mathcal{P}_C :=$ interp(P_{i, k}, P);
                  end if
                  $\Phi_{i,k} := \Phi_{i,k} + \mathcal{P}_C$;
                end for
              end if
            end for
          end for
        end for
      end for
      $\Phi_i := \Phi_i + \Phi_{i,k}$;
    end for
    $\Phi_{tot} := \Phi_{tot} + \Phi_i$;
  end for
  Return $\Phi_{tot}$.
```

Fig. 14. Interaction detection algorithm.

5.1 Simulated annealing: a brief overview

Simulated annealing is inspired by the annealing process in metallurgy where the state of material can be modified by controlling the cooling temperature. The physical annealing process consists in heating up a material to bring it to a high energy state. Then, it is slowly cooled down, keeping each given temperature stage for a sufficient duration until a thermodynamic balance is reached. The temperature is reduced according to a pre-described temperature reduction schedule, until the material reaches a global-minimum energy state and forms a crystallized solid. Decreasing too rapidly the temperature can however yield a non-desirable local minimum energy state.

In the simulated annealing optimization algorithm, the objective function to be minimized is analogical to the energy of the physical problem, while the decision variables of the problem correspond to the coordinates of the material's particles. A control parameter, T, that decreases as the number of iteration grows, plays the role of the temperature schedule, and a number of iterations, N_I, at each temperature step plays the role of time duration.

For a physical system, when the system reaches the thermodynamic balance at a given temperature, T, the energy, E, of its particles is distributed according to the *Boltzmann distribution*: $e^{\frac{-E}{k_B T}}$, where k_B is the Boltzmann constant. To simulate this evolution of the physical system towards the thermal equilibrium, the *Metropolis algorithm* is used. For a given temperature, T, starting from a current configuration, the state space of the simulated system is subjected to a transformation (e.g. apply a local change to one decision variable). If this transformation improves the objective function value, then it is accepted. Otherwise, it is accepted with a probability

$$P_{accept} := e^{\frac{\Delta E}{T}},$$

where ΔE is the degradation of the objective function value.

Once the number of iterations, N_I is reached, the temperature is decreased according to a pre-defined cooling schedule. As the temperature decreases, the probability, P_{accept}, to accept a degrading solution also decreases. The algorithm will eventually converge to a local optimum whose value is close to that of the global optimum if the temperature was decreased sufficiently slowly. The simulated annealing algorithm used here is summarized in Figure 15, where S_c represents a current solution, and S_N represents a neighboring solution generated by a *neighborhood function*, which will be presented in the following section.

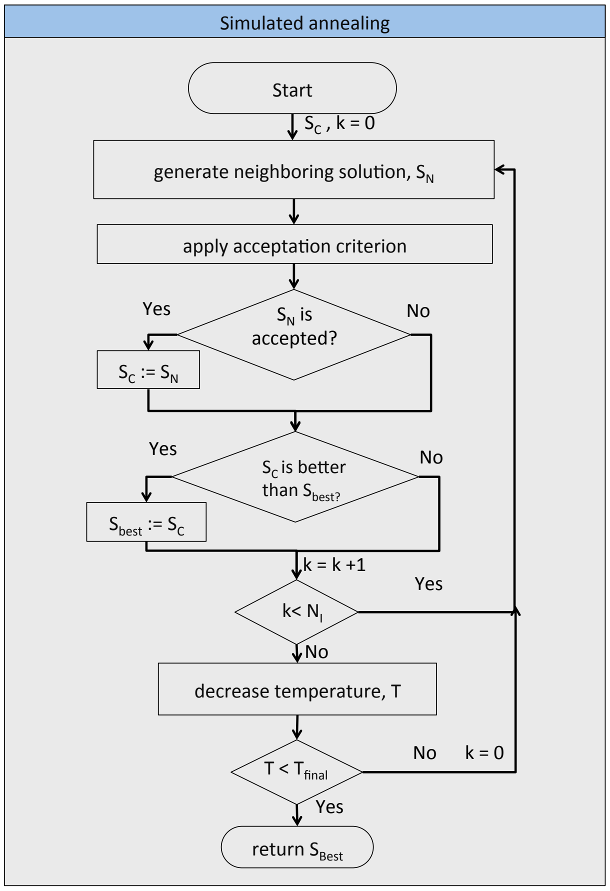

Fig. 15. Simulated annealing algorithm.

5.2 Iterative-improvement local search: a brief overview

An iterative-improvement local search is an algorithm that starts from a given initial solution, and then iteratively replaces the current solution with a better solution chosen in a pre-defined neighborhood. Given an initial solution, the iterative-improvement local search generates a neighborhood solution, and then accepts this new solution only if it yields an improvement of the objective-function value. The algorithm stops when a (pre-defined) maximum number of iteration,

N_{Loc}, is reached. The quality of the solution found by the local search depends on the initial solution, and on the definition of the neighborhood structure.

Let us denote by S_c the current solution, and let i be a given flight. In this work, we introduce a local-search module that relies on two different search strategies:

1. **Intensifying the search on one Particular Trajectory (PT).** This state-exploitation step focuses on improving S_c by applying a local change from a neighborhood structure involving solely flight i, so that only the decision variables $u_i = (\delta_i, l_i, w_i)$ can be modified.
2. **Intensifying the search on the Interacting Trajectories (IT).** This state-exploitation step applies a local change, from a neighborhood structure involving *several* flights: the flights that are currently interacting with flight i in solution S_c. Modifications to each of these *neighboring flights* are made sequentially in a greedy manner. For instance, suppose that trajectory i interacts with trajectories p, q, and r. Changes are sequentially applied to the decision variables u_p, u_q, and u_r. Obviously, the order in which the flights are considered may affect the quality of the resulting IT step.

5.3 Hybrid simulated-annealing / iterative-improvement local search

To implement the hybrid metaheuristic, we have to determine a structure to control the **level of hybridization** between each metaheuristic algorithm. According to [23], we may classify the level of hybridization into two levels:

– The *low-level hybridization* addresses an integration of metaheuristic algorithms, where each algorithm is strongly coupled with each other. In this case, an individual component in each metaheuristic may be replaced by, or exchanged, with a component from another metaheuristic algorithm. For instance, one may consider using a greedy heuristic as a crossover operator in a genetic algorithm.

– For the *high-level hybridization*, each algorithm retains its own characteristics without direct interactions between the internal functions of the algorithm. Information are exchanged between each self-contained metaheuristic algorithm through a well-defined interface. For instance, the best solution obtained from one metaheuristic algorithm can be used as an initial solution for another metaheuristic algorithm.

Another property to consider is the **order of carrying out** each metaheuristic algorithm. In general, the algorithms can be run either in a sequential, an interleaved, or a parallel manners.

Finally, one has to determine the **values** of the various user-provided **parameters** specifying the metaheuristics. These parameters control, for example, the balance between exploration and exploitation of the solution space and must be fine-tuned with care as they have a strong impact on the quality of the solution obtained.

For the sake of simplicity in this preliminary implementation, the simulated annealing and the iterative-improvement local search are hybridized in a self-contained (high-level) manner where each algorithm is sequentially run. The iterative local search is integrated as an *inner loop* in the simulated annealing (SA) algorithm so that the local search is considered as one search step of the SA. The initial solution of the local search is provided by the current solution of the simulated annealing. The solution found by the local search is returned to the simulated annealing, where an acceptance condition will be systematically applied.

The order of carrying out each metaheuristic is given as follows.

– At each iteration of the hybrid algorithm, one flight is randomly chosen among all flights featuring a certain, pre-defined level of interaction. More precisely, let Φ_τ be a pre-defined interaction threshold value (provided by the user). The hybrid algorithm chooses randomly one flight in the set $\{i \in \{1, 2, \ldots, N\} : \Phi_i \geq \Phi_\tau\}$. Let i denote the selected flight.

– Then, the hybrid algorithm determines whether to perform a classical SA step, or to trigger the iterative-improvement local search, or to perform both search strategies successively. This decision is taken according to a specific (user-defined) probability that depends upon the control temperature, T, and the value of the term, Φ_i of the objective function corresponding to flight i.

The **probability to carry out an SA** step, P_{SA}, is:

$$P_{SA}(T) = P_{SA,min} + (P_{SA,max} - P_{SA,min}) \cdot \frac{T_0 - T}{T_0}, \tag{8}$$

where $P_{SA,max}$ and $P_{SA,min}$ are the (user-provided) maximum and minimum allowed probabilities to perform SA (pre-defined by the user).

The **probability of running the iterative-improvement local search** module, P_{Loc}, is given by:

$$P_{Loc}(T) = P_{Loc,min} + (P_{Loc,max} - P_{Loc,min}) \cdot \frac{T_0 - T}{T_0}, \tag{9}$$

where, similarly, $P_{Loc,max}$ and $P_{Loc,min}$ are the (user-provided) maximum and minimum probabilities to perform the local search.

Finally, the **probability of carrying out both** SA and the local search (successively), P_{SL}, is simply:

$$P_{SL}(T) = 1 - (P_{SA}(T) + P_{Loc}(T)) \tag{10}$$

A key factor in tuning this hybrid algorithm is to reach a good trade-off between exploration (diversification) and exploitation (intensification) of the solution space, i.e. a compromise between fine convergence towards local minima, and the computation time invested in exploring the whole search space in order not to miss a global optimum.

The proposed hybrid algorithm is detailed in Figure 16, where T_{init} and T_{final} are respectively the initial and the final temperatures of the (user-provided) cooling schedule, and where N_I is the maximal number of iterations at each temperature step (also set by the user).

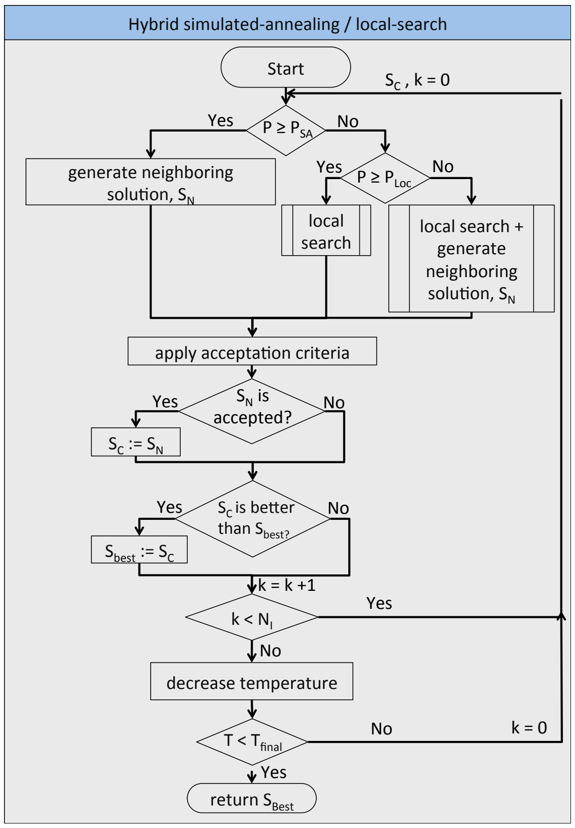

Fig. 16. Hybrid simulated-annealing / iterative-improvement local search algorithm.

5.4 Neighborhood function.

A neighboring solution is generated by applying a so-called *neighborhood function* (or transformation operator) that generates a local change to the current solution. This change should be computed rapidly, but should not involve a drastic change in the current solution. Otherwise, the characteristics of the SA will become those of a pure random search.

To generate a neighboring solution, first a flight, i, to be modified is chosen. Then, one has to determine whether to modify the location of waypoints, or to modify its departure time, or its flight level. In general, searching for a neighboring solution in the time domain would be more preferable, since it does not induce extra fuel consumption. However, empirical tests show that limiting the search to only that degree of freedom results in prohibitive computational times before reaching a reasonably good solution.

Therefore, we introduce further user-defined parameters: $0 \leq P_w \leq 1$, $0 \leq P_{FL} \leq 1$, and P_δ to control respectively the probabilities of modifying: the location of the waypoints, the flight level, and the departure time of the chosen flight i. Once P_w and P_{FL} are chosen, P_δ is simply defined to satisfy $P_w + P_{FL} + P_\delta = 1$. These parameters allow the user to implement his preferences for the resulting conflict-free 4D trajectories.

6 Computational results

The proposed strategic 4D trajectory planning methodology is implemented with the programming language Java on an AMD Opteron 2 GHz processor with 128 Gb RAM.

The methodology is tested on air traffic data representing a full-day en-route air traffic over the French airspace. It consists of N = 8,836 trajectories. Figure 17 and 18 illustrate the initial given trajectory sampling with $t_s = 60$ seconds in both the horizontal and the vertical planes (the dense area located at the coordinate point $(0; 5 \times 10^6)$ on Figure 17 corresponds to Paris).

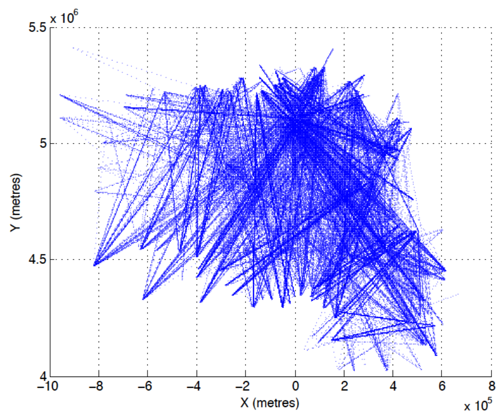

Fig. 17. The initial given trajectories consisting of a full-day traffic over the French airspace in the horizontal plane.

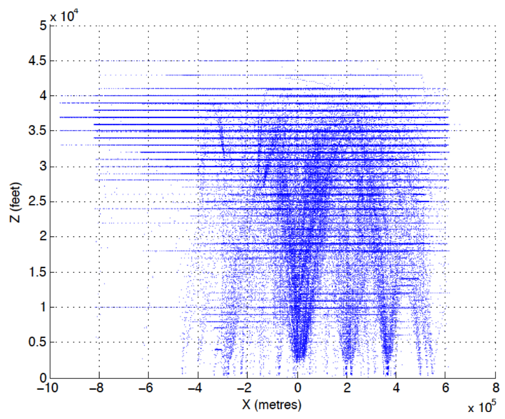

Fig. 18. The initial given trajectories consisting of a full-day traffic over the French airspace in the vertical plane.

The parameter values that specify the problem under consideration are given in Table 1. Simply to give an idea of the complexity of the computation of the objective function of this problem instance; when using the sampling time-step value $t_s = 20$ seconds, the N trajectories are discretized into between 1,388,080 and 2,175,928 sample 4D points according to the location of waypoints used to modify the shape of trajectories. With regard to the dimension of the search space, remark that our optimization problem involves for this instance:

– $2MN$ = 53,016 (continuous) waypoint variables ($\mathbf{w}$);
– $N\frac{|\delta_a^i|+|\delta_d^i|}{\delta_s} + 1$ = 3,189,796 (discrete) departure-time shifts variables ($\boldsymbol{\delta}$);
– $N(2l_{i,max} + 1)$= 44,180 (discrete) flight-level shifts variables ($\mathbf{l}$);

for a total of 53,016 continuous variables and 3,233,976 discrete variables. Empirical tests lead us to set the values of the parameters of the hybrid simulated-annealing / local-search as presented in Table 2.

Table 1. (User-defined) parameter values of the problem

parameters	value
Sampling time step, t_s	20 seconds
Inner-loop interpolation sampling time step, t_{interp}	5 seconds
Maximum departure time shift, $\delta_a^i = \delta_d^i$	60 minutes
Discretization time step for possible delay / advance departure-time shift	20 seconds
Maximum allowed route length extension, d_i	0.20
Maximum number of flight level shifts, $l_{i,max}$	2
Maximum number of waypoints, M	3
Probability to modify horizontal flight profile, P_w	1/3
Probability to modify flight level, P_{FL}	1/3
Probability to modify departure time, P_δ	1/3

Table 2. Empirically-set (user-defined) parameter values of the hybrid algorithm.

Parameter	value
Number of iterations at each temperature step, N_I	200
Number of iterations of the inner loop local search, N_{Loc}	5
Geometrical temperature reduction coefficient, β	0.99
Final temperature, T_{final}	$(1/500).T_{init}$
Probability to carry out simulated annealing, P_{SA}	$0.8 + 0.1\ T/T_{init}$
Probability to carry out the inner-loop local search, P_{Loc}	$0.4 + 0.2\ T/T_{init}$

The initial temperature, T_{init}, is calculated using an algorithm proposed in [13, pages 44 − 45]. It is computed by initiating 100 deteriorating disturbances at random; evaluating the average variation ($\Delta\Phi_{avg}$) of the objective-function value; and then deducing T_{init} from the relation: $e^{\frac{-\Delta\Phi_{avg}}{T_{init}}} = \tau_0$, where τ_0 is the initial rate of accepting degrading solutions whose value depends on the assumed quality of the initial configuration. Empirical tests leads us to set $\tau_0 = 0.3$. Each temperature reduction is performed using a geometric reduction coefficient, β, whose value is provided by the user: $T_{k+1} := \beta T_k$.

The simulations are performed considering successively aircraft time uncertainty of 2 and 3 minutes, respectively. The proposed strategic trajectory planning methodology is able to find interaction-free trajectory plans for both cases. The required computation time for each problem is presented in Table 3. When considering higher level of time uncertainty (3 minutes), the solution space becomes more constrained and therefore the algorithm requires more computation time to converge.

The number of modified flight plans for both cases is compared in Figure 19. The average changes made to the initial flight plan is shown in Table 4. The proposed algorithm is able to ensure separation of these nation-wide scale aircraft trajectories by modifying roughly 50% of the initial flight plans, yielding an average of 5% route length extension, 1.5 flight level shifts, and 30 minutes departure-time shifts.

Table 3. Numerical results considering aircraft time uncertainty of 2 and 3 minutes.

time uncertainty interval ($2t_\epsilon$) (minutes)	initial Φ_{tot}	resulting Φ_{tot}	computation time (minutes)	No. of iterations performed
2	217,441.37	0.0	188.07	653,970
3	274,953.55	0.0	2210.42	5,583,192

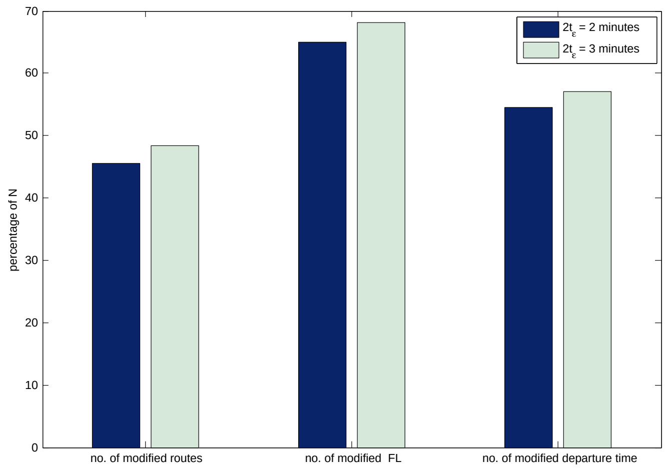

Fig. 19. Number of modified flight plans considering time uncertainty of 2 and 3 minutes.

Table 4. Average modifications applied to the initial flight plan.

time uncertainty interval ($2t_\epsilon$) (minutes)	avg. route length extension ($\%L_{i,0}$)	avg. FL shifts	avg. departure-time shifts (minutes)
2	5.43	1.55	30.37
3	5.66	1.55	30.15

7 Conclusions

We introduced an efficient methodology to address the strategic trajectory planning problem in the framework of future trajectory-based ATM operations assuming time uncertainty on the position of the aircraft along its 4D trajectory. The proposed methodology minimizes interaction (a sum of conflict probabilities) between trajectories and results in conflict-free flight plans on a nation-wide scale data instance. Uncertainty of aircraft arrival time at a given position is taken into account by modeling the predicted aircraft arrival time as a triangular probability distribution over the uncertainty time period. The proposed trajectory planning approach relies on a route / flight-level / departure-time allocation technique to separate the aircraft trajectories. The problem was modeled mathematically under the form of a mixed-integer optimization problem aiming at minimizing total interaction between trajectories.

The problems we aim at addressing involve a full-day of traffic at the nation-wide scale involving more than 8,000 trajectories. In order to ensure separation between trajectories under time uncertainty, we developed an efficient grid-based conflict detection module. To reduce the number of sampling points needed while minimizing further the computation time, this interaction-detection method interpolates the aircraft position between two suspected sampling points instead of refining the sampling-time step.

To find an optimal route, a flight level, and a departure time for each flight, we rely on a hybrid-metaheuristic optimization algorithm that combines the advantages of simulated annealing and of an iterative-improvement local-search method. The simulated annealing part ensures diversity of the candidate solutions considered, while the local-search module intensifies the search in promising regions of the feasible domain in order to accelerate convergence.

Computational experiments on a day instance of en-route air traffic over the French airspace with different levels of time uncertainty show that the proposed methodology is able to find interaction-free (i.e. conflict-free) trajectory plans, and to ensure separation between aircraft trajectories in presence of time uncertainty, within a computation time viable for the strategic planning level.

Further research should concentrate on reducing the extra route length, and the number of flight level shifts, and of departure-time shifts, instead of being content with (possibly costly) interaction-free solutions. Another challenging research track is to take into account equity between airlines in the trajectory optimization process when deciding which trajectory is to be modified in order to avoid a conflict.

References

1. A. Agustín, A. Alonso-Ayuso, L.F. Escudero, and C. Pizarro. Mathematical optimization models for air traffic flow management: A review. In *A. Bui, I. Tseveendorkj (Eds.), Combinatorial Optimization in Practice*, volume 8, pages 141–184. Hermann Informatique, Studia Informatica Universalis, 2010.
2. A. Akgunduz, B. Jaumard, and G. Moeini. Non-time indexed modeling for en-route flight planning with speed-fuel consumption trade-off. In 2nd *International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Toulouse (France), 2013.
3. N. Barnier and C. Allignol. 4D - trajectory deconfliction through departure time adjustment. In 8th *USA/Europe Air Traffic Management Research and Development Seminar (ATM 2009)*, Napa (California), 2009.
4. N. Barnier and C. Allignol. Combining flight level allocation with ground holding to optimize 4D-deconfliction. In *USA/Europe ATM Seminar: Air Traffic Management Research and Development Seminar*, Berlin, 2011.
5. D. Bertsimas, G. Lulli, and A. Odoni. The air traffic flow management problem: An integer optimization approach. In *Integer Programming and Combinatorial Optimization*, volume 5035 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2008.
6. D. Bertsimas, G. Lulli, and A. Odoni. An integer optimization approach to large-scale air traffic flow management. *Operations Research*, 59(2):211–227, 2011.
7. D. Bertsimas and S. Patterson. The air traffic flow management problem with en-route capacities. *Operations Research*, 46(3):406–422, 1998.
8. S. Cafieri and N. Durand. Aircraft deconfliction with speed regulation: New models from mixed-integer optimization. *Journal of Global Optimization*, 2013.

9. S. Chaimatanan, D. Delahaye, and M. Mongeau. A methodology for strategic planning of aircraft trajectories using simulated annealing. In *International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Daytona beach (Florida), 2012.
10. S. Chaimatanan, D. Delahaye, and M. Mongeau. Strategic deconfliction of aircraft trajectories. In 2^{nd} *International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Toulouse (France), 2013.
11. S. Constans, B. Fontaine, and R. Fondacci. Minimizing potential conflict quantity with speed control. In *4th Eurocontrol Innovative Research Workshop And Exhibition*, pages 265–274, Bretigny-sur-Orge (France), 2005.
12. D. Delahaye and S. Puechmorel. *Modeling and Optimization of Air Traffic.* Wiley-ISTE, 2013.
13. J. Dreo, A. Petrowski, P. Siarry, and E. Taillard. *Metaheuristics for hard optimization.* Springer, 2006.
14. N. Durand and J.B. Gotteland. Genetic algorithms applied to air traffic management. In *Metaheuristics for Hard Optimization*, pages 277–306. Springer, 2006.
15. D. Gianazza and N. Durand. Separating air traffic flows by allocating 3D-trajectories. In *DASC 04. The 23rd Digital Avionics Systems Conference, 2004*, volume 1, pages 2.D.4–21–13 Vol.1, 2004.
16. I. Hwang and C.E. Seah. Intent-based probabilistic conflict detection for the next generation air transportation system. *Proceedings of the IEEE*, 96(12):2040–2059, 2008.
17. J.K. Kuchar and L.C. Yang. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, 1:179–189, 2000.
18. J. Lohn and J. Rios. A comparison of optimization approaches for nationwide traffic flow management. In *AIAA Guidance, Navigation and Control Conference*, Chicago, 2009.
19. A.R. Odoni. The flow management problem in air traffic control. In *Odoni, A.R., Bianco, L., Szego, G. (eds.), Flow Control of Congested Networks*, pages 269–288. Springer, 1987.
20. S. Oussedik. *Application de l'Évolution artificielle aux problèmes de congestion du trafic aérien.* PhD thesis, École Polytechnique, Palaiseau (France), 2000.
21. S. Oussedik and D. Delahaye. Reduction of air traffic congestion by genetic algorithms. In *PSSN V: Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*, pages 855–864. Springer, 1998.
22. M. Prandini, J. Hu, J. Lygeros, and S. Sastry. A probabilistic approach to aircraft conflict detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):199–220, Dec 2000.
23. E-G. Talbi. *Metaheuristics: From Design to Implementation.* Wiley, 2009.
24. M. Terrab and A.R. Odoni. Strategic flow management for air traffic control. *Operations Research*, 41(1):138–152, 1993.