TOWARDS AN AUTOMATIC AIR TRAFFIC CONTROLLER

by

Sepanta Sekhavat and Shankar Sastry

Memorandum No. UCB/ERL M98/63

10 November 1998

cover page

TOWARDS AN AUTOMATIC AIR TRAFFIC CONTROLLER

by

Sepanta Sekhavat and Shankar Sastry

Memorandum No. UCB/ERL M98/63

10 November 1998

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Towards an Automatic Air Traffic Controller*

Sepanta Sekhavat and Shankar Sastry
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720
{sepanta,sastry}@robotics.eecs.berkeley.edu

Abstract

The current Air Traffic Management System (ATMS) relies mainly upon human Air Traffic Controllers (ATCs) to solve conflicts between aircraft plans. Due to the increasing traffic, the workload of centralized ATCs will be soon too heavy to handle. A new generation of ATMS is urged to guarantee the safety of future travelers and to reduce financial loss due to ATC-caused delays. Thanks to the new technological advances such as on board computational power, localization and communication features, conception of radically different ATMS including more automation can be envisaged. The ATMS presented in this paper is based on a Distributed Automatic Air Traffic Controller (DAATC). It solves automatically a large number of conflicts for the global system by synchronizing in a near optimal way, independant solutions computed for different parts of the system. Safety is permanently guaranteed while the system solves most of its conflicts (including *free flight* attempts) in a decentralized way, including efficiency and fairness as criteria.

1 Introduction

Air traffic management requires coordination and control of a large number of aircraft. According to the Federal Airspace Administration (FAA), this number is expected to grow for at least the next 15 years (3 to 5 percent per year for the U.S. rate). The number of control decisions that have to be made and the complexity of the resulting process imply that the current centralized management turns to be more and more **inefficient** and even **unsafe**.

This is mainly due to two linked factors which are centralization and under-utilization of resources. Currently, at the beginning of the day, aircraft have their flight schedules which can rarely be fully respected. This is due to various reasons including airport delays and changing weather conditions. Therefore, unpredicted conflicts may appear at the execution time. Currently, the airspace is divided into several parts. For each of them, an Air Traffic Control (ATC) group on the ground maintains safe spacing between aircraft while guiding them to their destinations. Thus, the resolution of unpredicted appearing conflicts becomes a complex centralized problem left to arbitrary human judgment. Therefore, there is no real guarantee of safety. The increase of air traffic worsens this situation. In other respects, in order to simplify the complexity due to centralization, the airspace is very rigidly structured and aircraft are forced to fly along predetermined *jetways*.

*Research supported by the Army Research Office MURI program on "An Integrated Approach to Intelligent Systems", under grant DAAH04-96-1-0341.

This under-utilization of the airspace leads naturally to trajectories that may be far from optimal with respect to the local weather conditions (direction of the wind, weather hazards, etc).

Future generations of ATMS will be expected to integrate more automation, on the ground as well as on board to reduce the human undeterministic intervention, to evolve through a decentralized ATMS and eventually guarantee the safety of the system. Furthermore, integration of efficiency criteria in the automatic conflict resolution procedures and the use of the whole 3D airspace can improve the performances of the current ATMS in terms of time, fuel consumption, etc. These are the motivations of this work for a Distributed Automatic ATC (DAATC), guaranteeing safety in a rigorous way, allowing the use of the whole airspace and optimizing the aircraft scheduling.

2 General presentation

In order to reach a better ATMS, different decision support tools have endeavored to reduce the ground controllers workload [6, 4]. The current trends in ATMS is to reduce even more the controllers intervention by moving towards decentralization and *free flight* capabilities [14]. This paper aims at a fully automatic decentralized controller allowing for free flight.

The ATC has to solve spatio-temporal conflicts. Indeed, different aircraft must not be at the same time in the same neighborhood. The ideal way of solving the problem is then to consider it directly at the spatio-temporal level. However, algorithms for time-extended configuration space [8] prove to be computationally hard. With additional velocity bounds, the multi-agent motion planning problem becomes NP-complete [5]. Thus, designing a general method of conflict detection and resolution for n aircraft is a difficult task. Actually, the existing safe methods deal with a relatively few number (2,3) of aircraft [15, 11, 10, 17]. The methods considering larger number of aircraft may be safe only under some strong restrictions on the type of conflict that may occur [9, 10, 13]. Also, most of these methods consider only coplanar conflict resolution and do not fully take into account 3D motions.

Although a general conflict resolution control strategy for n aircraft does not exist, there are increasing number of proposed Conflict Resolution Procedures (CRPs) for specific situations. The ATMS presented in this paper integrates these CRPs to solve the maximum of conflicts in the *space-time* configuration space. However, this operation is not a straight application of CRPs. Indeed, imagine that the ATC uses one of these procedures to solve a conflict between $k < n$ aircraft. The suggested solution assumed that the k aircraft are alone in the airspace. Applying this solution in the n aircraft context may lead to new conflicts with m other aircraft. A new conflict resolution procedure has then to be applied to the $k + m$ aircraft. After some iterations a solution may be found and will then be executed. However one can also imagine cases where after some iterations, the number of aircraft affected by the conflict lies beyond the power of available CRPs. Actually, these remaining conflicts are the ones that are too complex to be solved at the spatio-temporal level, with respect to the current state of the art. The main idea here is to solve them at the temporal level. As we will see, aircraft paths will be *roughly* considered as fixed and DAATC establishes some priority order between them to cross the conflict points. Notice that solving conflicts at a temporal level is what we have to face anyway, close to the airports, where the airspace becomes scarce. To summarize, the global problem is solved by synchronizing solutions computed for different parts of the system considered separately. Most of the conflicts can be solved in this way, even with elementary CRPs. However, in absence of a general n-aircraft planner, exceptional human helps remain necessary not for safety but for completeness of this ATMS. Thus, as we will see, the human controller becomes simply a part of DAATC for exceptional cases. Yet, his workload is considerably reduced. Above all guaranteeing safety is no longer of his responsibility. He will disappear in long term with the progress of conflict resolution procedures.

In the current ATMS, human designed plans are modified and synchronized by human controllers to avoid conflicts. In the ATMS presented in this paper, these operations are automatic. Indeed, the basic trajectories are automatically computed by planners solving as many conflict as possible using different available CRPs. The automatic controller takes care of merging these trajectories in the global n aircraft context in a way that guarantees **safety** and promotes **efficiency**. This conflict resolution scheme is distributed and different conflicts may be solved in parallel. In other respects, DAATC imposes no constraints on the type of aircraft trajectories which are allowed to use the whole airspace.

The main constraints that an ATMS has to respect are safety and efficiency. Concerning safety, the airspace around aircraft paths are dynamically partitioned in cells. These cells are considered as unsharable resources between aircraft. This partition is done in a local and distributed way (Section 3). It takes into account the imprecision on aircraft positions and is such that each aircraft can remain in safety in a given cell. By establishing consistent priority orders, The DAATC guarantees that an aircraft gets the authorization to transit to the next cell only once this later is safe (Section 4). In this way the safety of the whole system is permanently guaranteed. The efficiency of an ATMS consists mainly in respecting arrival times and reducing fuel consumption. These points and other criteria are taken into account at several level. We have seen that some of the conflicts are solved using partial conflict resolution procedures. By iterative computation, the system choose the best procedure adapted to the conflict. For the remaining conflicts, DAATC aims to optimize the priority order for cell transitions in order to globally reduce fuel consumption and possibly eliminate waiting times for cell transitions(Section 5). A corresponding schedule is computed and used by the aircraft. Note that this schedule is indicatory and cannot change the priority orders for crossing conflict zones. Indeed, these priority orders guarantee safety and feasibility of the global plan. Thus DAATC remains robust not only to positions uncertainty but also to execution time delays.

3 Safety

The major problem to guarantee aircraft safety comes from their dynamics. Indeed, a late detected conflict may be unavoidable. A way of warranting safety in any situation is to make sure that at each moment, each aircraft has enough free airspace at its disposal. This airspace will be reserved for its *individual use*. It will be big enough so that in case of problem, the aircraft can remains in safety inside and its dynamics do not force it to exit. Thus, we define along each aircraft path a succession of cells which will be reserved one after the other by the aircraft before transiting in. If two cells of two different aircraft intersect, they cannot be reserved at the same time. In this case, DAATC determines which aircraft has the priority for reservation. A real conflict may or may not happen at the execution time. In the worst case, one of the aircraft will have to wait some time in the cell preceding the conflictual one. Notice that making aircraft wait in holding patterns is what human ATC do currently to solve many conflicts. Our goal is just to make this operation automatic and methodic in order to guarantee safety. However, we will see in Section 5 how to reduce the risk of waiting periods to improve also efficiency.

Designing aircraft paths, detecting conflicts and establishing priority orders will be dealt with in the Section 4. In this section, we assume that some aircraft paths have been already chosen. We take an interest in how to build the corresponding cells. More precisely :

- We have to find a good compromise between the small number of cells and small size of cells. Indeed, cells are unsharable resources. Therefore, in case of conflict, it is interesting to have them the smallest possible. In this way, we constrain the less possible the other aircraft plans. However, too many small cells will needlessly make their management difficult. Furthermore, in a given cell, an aircraft is free to choose the variations of its velocity. We will see in section 5 that for optimization reasons, it is interesting to have big cells.
- As we will see in the next section, different aircraft paths are designed and validated in the global context by small pieces, during the flight. Therefore, the cells construction must be done dynamically and updated dynamically when new conflicts appear.

3.1 The elementary cell :

Let us start by defining the *Elementary Cell* $EC(M)$ which is the minimal airspace that we have to reserve for an aircraft at a given point M. As its main characteristic, EC must be able to include a *holding pattern*. That means a stationary motion along which the aircraft remains inside EC.

Notice that FAA imposes already a protected zone around each aircraft. The protected zone is a virtual *hockey puck* centered on the aircraft, whose radius and height are respectively 2.5 nautical miles and 2,000 ft. If we consider for example a circular holding pattern for an aircraft with a velocity of 300 mph, a roll angle of 20 degrees allows a radius of curvature of 3.13 miles. This will subject the passengers to an apparent weight of 1.06g in the seats. If we want to conserve a protected zone of 2.5 miles radius along this holding pattern, EC should have a radius that is just 2.25 times the one of the original protected zone. Therefore, we are not really adding a new geometric constraint.

In a general way, the shape and the size of EC may vary along the path and depend on several factors such as :

- The shape and the size of the protected zone.
- The shape of the holding pattern.
- The uncertainty margin with respect to the nominal paths.
- Sensors uncertainty, ...

which in turn may depend on the type and the velocity of the aircraft. However, for a given portion of a path, we can find an appropriate fixed size and shape of the EC.

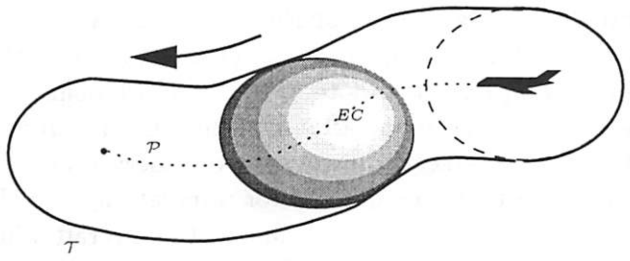


Figure 1: The tube $\mathcal{T}$ around the path $\mathcal{P}$

3.2 Cell decomposition of the tubes

Given a path $\mathcal{P}(s)$ of an aircraft in airspace parameterized by the curvilinear abscissa $s \in [s_{start}, s_{end}]$, let us define a general cell :

$$C(s_0, s_1) = \bigcup_{s \in [s_0, s_1]} EC(\mathcal{P}(s))$$

which is the swept volume of EC centered on the path $\mathcal{P}$, swept between $\mathcal{P}(s_0)$ and $\mathcal{P}(s_1)$. We denote :

$$\mathcal{T} = C(s_{start}, s_{end})$$

the tube around $\mathcal{P}$ which is also the biggest cell (see Figure 1). Notice that a cell has a minimum volume corresponding to $EC = C(s, s)$.

Now, consider a set of aircraft $\mathcal{A}^1, \ldots, \mathcal{A}^n$, the corresponding paths $\mathcal{P}^1, \ldots, \mathcal{P}^n$ and the corresponding tubes $\mathcal{T}^1, \ldots, \mathcal{T}^n$ around these paths. Given a new path $\mathcal{P}^0$, any intersection of $\mathcal{T}^0$ with other tubes leads to a decomposition of $\mathcal{T}^0$ in new cells. Each cell is characterized by its use by some given set of airplanes. Roughly, one cell is split in two if the lists of aircraft using each of them are not equal. Let us call $\mathcal{L}(s)$, the list of the aircraft whose tubes intersect the $C(s, s)$ of $\mathcal{P}^0$. The following algorithm proposes one cell decomposition of $\mathcal{T}^0 = C(s^0_{start}, s^0_{end})$ given $ListTubes = \{\mathcal{T}^1, \ldots, \mathcal{T}^n\}$:

```
CellDecomposition($\mathcal{T}^0$, ListTubes)

$k := 0$
$s_{start} = s^0_{start}$
do{
    $\mathcal{L}_k := \mathcal{L}(s_{start})$
    $s_{end} := min\{u \mid s_{start} \le u \le s^0_{end}, \mathcal{L}(u) \ne \mathcal{L}_k\}$
    if $s_{end} = s_{start}$ {
                    $s_{end} := s^0_{end}$
                    $\mathcal{C}_k := C(s_{start}, s_{end})$
                    break
    }
    $\mathcal{C}_k := C(s_{start}, s_{end})$
    $s_{start} := s_{end}$
    $k := k + 1$
}
```

At the end of the algorithm, $\mathcal{T}^0 = \bigcup_k \mathcal{C}_k$ and for all k, $\mathcal{L}_k$ is the list of the airplanes using (at least partly) $\mathcal{C}^k$. This algorithm gives a satisfactory cell decomposition with respect to our compromise : unshared cells are as large as possible, shared cells are just big enough and there is no useless cell. See Figure 2

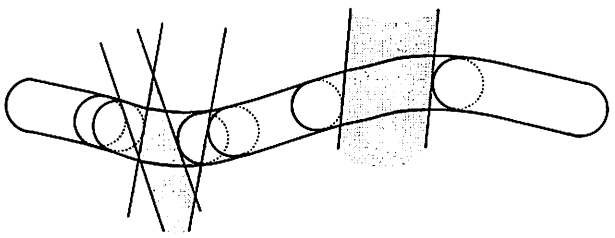


Figure 2: Cell decomposition of the tube $\mathcal{T}^0$.

The intersections of the new tube $\mathcal{T}^0$ with pre-existent tubes lead not only to $\mathcal{T}^0$ cell decomposition but also to update the one of the previous tubes. Indeed, let us imagine that $\mathcal{T}^0$ crosses a long cell $\mathcal{C}^i$ of another tube $\mathcal{T}^i$. Instead of forbidding the whole cell to the aircraft $\mathcal{A}^i$ while $\mathcal{A}^0$ is crossing, we can divide $\mathcal{C}^i$ into several cells and optimize the use of the space. For each new tube $\mathcal{T}^0$, this updating concerns only the airplanes listed in one of the elements of $\{\mathcal{L}_1^0, \ldots, \mathcal{L}_p^0\}$. The updating corresponds to a local application of the same cell decomposition algorithm.

Notice that the cell construction is a distributed operation. Each aircraft asks its neighbors about the position of their tube and designs its cells on its own.

By using cells and reserving them beforehand, our approach will be safe by construction. Now the question is : how successful this approach is to solve conflicts. In order to answer, we have to explain more precisely how the paths are computed and the priority orders established.

4 The coordination scheme

The cells along a path are used to associate a plan to the path tracking.

Definition 1 *Given an aircraft $\mathcal{A}$ tracking a path $\mathcal{P}$ and given C_i the cells built along $\mathcal{P}$. We define the aircraft* **individual plan** *as a sequence of* **states** *$\mathcal{S}_i$ defined with respect to the cells $C_i = C(s_i^{start}, s_i^{end})$ (see figure 3). The aircraft $\mathcal{A}$ is in the state $\mathcal{S}_i$ if its nominal position s along $\mathcal{P}$ is in $[s_i^{start}, s_i^{end}[$. Which implies that its real position is somewhere inside C_i. The* **transition** *T^i (see Figure 3) corresponds to the execution events of transiting from the state $\mathcal{S}^i$ to $\mathcal{S}^{i+1}$.*

Notice that when $\mathcal{A}$ executes a holding pattern, its nominal position is frozen at the abscissa at which it starts the holding pattern. For two aircraft $\mathcal{A}^1$ and $\mathcal{A}^2$ and the respective cells C_i^1 and C_j^2, if $C_i^1 \cap C_j^2 \neq \emptyset$, a possibility of conflict exists between the states $\mathcal{S}_i^1$ and $\mathcal{S}_j^2$. As mentioned before, in order to guarantee safety a priority order is set to make sure that $\mathcal{S}_i^1$ and $\mathcal{S}_j^2$ will not be occupied at the same time. These constraints are represented by arrows on the Figure 3. An arrow means that the event at the bottom of the arrow must happen before the event to which the arrow points. Setting the priority constraints validate the individual plan in the global context. This operation is called in the sequel the **plan merging operation** *PMO*.

Definition 2 *The* **coordination plan** *of an aircraft is its individual plan plus the set of affecting priority constraints resulted from the merging of its individual plan into the global context.*

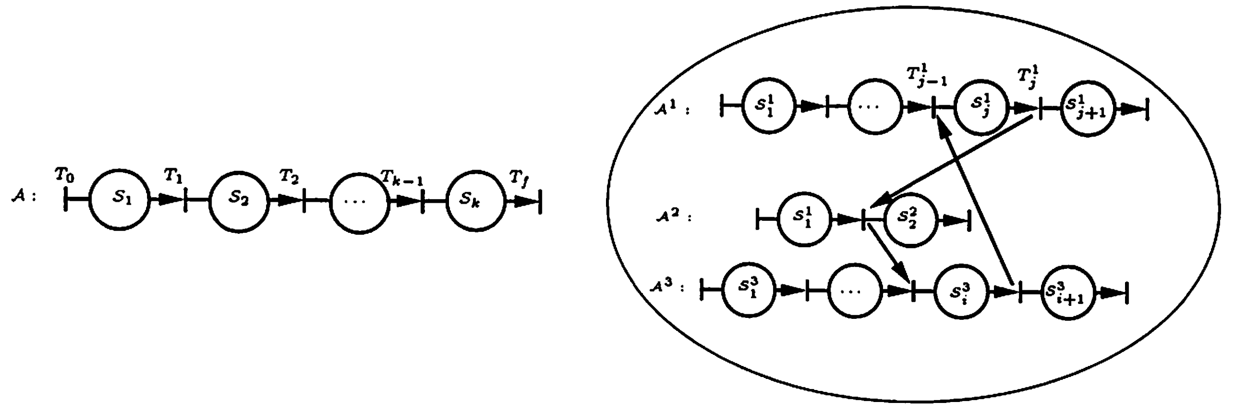


Figure 3: An individual aircraft plan (on the left) and a subsystem plan (on the right).

Actually, the notions of aircraft individual and coordination plans apply more generally to a subsystem with respect to the rest of the system. Only coordination plans are executed by aircraft. Tracking a path consists then in transiting through the sequence of states defining the coordination plan. An aircraft can remain in safety in a given state and choose the transition time by its own. Before each transition, the aircraft just has to check that the possible priority constraints are respected. This is done through local communication between aircraft.

The advantage of working with plans rather than trajectories is that the dynamics of the system disappear in plans. Synchronizing *continuous* trajectories with *complex dynamics* is a difficult task and the result will not be robust. We work instead with *discretized* plans with *no dynamics*. In this way we can synchronize path tracking by setting precedence between *execution events* rather than using clock time, which is much more robust.

4.1 The protocol

The protocol that each aircraft follows to create its plan and to establish priority orders is presented by the automaton of Figure 4 . By multiple aspects, this protocol is inspired by the *Plan Merging Paradigm PMP* [3, 2]. Indeed, the formulation of our problem in terms of dynamics free plans allows the application of PMP. However as we will see, our protocol differs definitely from PMP in several points. The major difference is in the very basic idea of PMP concerning the way the priorities are distributed. the plan merging rule in PMP can *roughly* be stated as :"The last one to merge has the lowest priority". This rule does not include any efficiency consideration and even imposes some limitations. We propose another plan merging rule in our protocol aiming at an optimized distribution of priorities (see Section 5).

However inspired from PMP, planning is done in a decentralized way and in parallel by different aircraft. Each aircraft builds its own individual plan which allows it for example the possibility of planning a *free-flight*. This plan has then to be merged in the global context. Therefore at each step, only a partial plan for a short range of time is built, so as to avoid over constraining the other aircraft plans. In practice, to build and merge a plan, an aircraft $\mathcal{A}^*$ first designs a partial path and the tube around it. Then $\mathcal{A}^*$ asks for the unsharable right to process a PMO. Once it gets this right, $\mathcal{A}^*$ collects the neighbor aircraft tubes and cell decompositions. The *CellDecomposition* algorithm is applied to $\mathcal{A}^*$ tube and possibly to some other tubes for some local refinement. The $\mathcal{A}^*$ plan is directly deduced from the cell decomposition and priorities are chosen with respect to some rules exposed later.

Notice that planning and execution can be done most of the time in parallel. An aircraft executes its current plan while preparing its future partial plan. Also different aircraft can plan in parallel. However in order to guarantee the coherence of the protocol, at each time only one aircraft is allowed to execute a PMO. In this way we make sure that the global context in which we try to merge safely a new plan, will not change during the PMO. Thus the PMO will not miss a new conflict appearing in parallel. The exclusive right to PMO can be handled in a decentralized way using a distributed mutual exclusion protocol (see for example [12]). However, if two conflicts appear in two disconnected subsystems, even the plan merging can be done in parallel. Notice also that there is a connection between the communication range d_c among aircraft and the maximum length of the trajectory d_t under execution for any aircraft of the system[1]. This relation is of the type : $d_c > 2d_t$. In this case, we can be sure that if an aircraft has not received the broadcast, it is further than $2d_t$ from the planning aircraft so its current trajectory (shorter than d_t) cannot

[1]More precisely, the interesting distance d_t is the largest distance between any to points of the trajectory under execution.

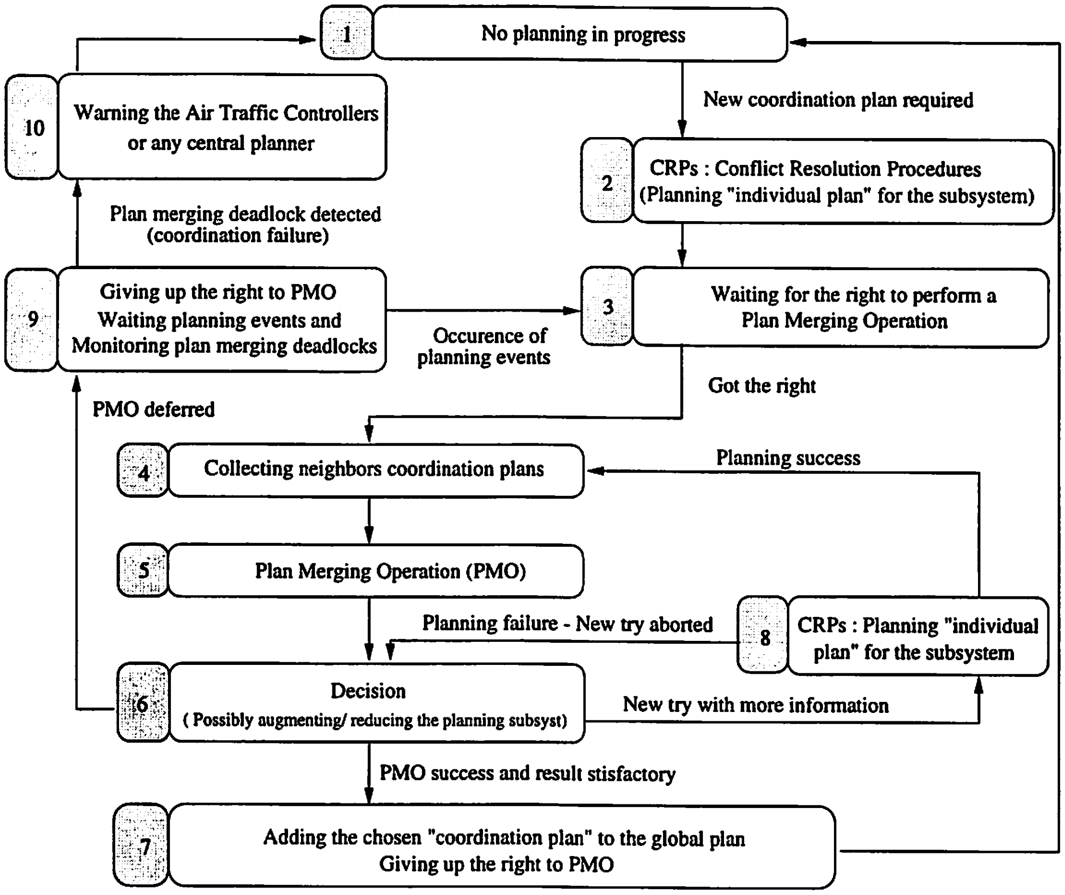

1
No planning in progress
New coordination plan required
2
CRPs : Conflict Resolution Procedures
(Planning "individual plan" for the subsystem)
10
Warning the Air Traffic Controllers
or any central planner
Plan merging deadlock detected
(coordination failure)
9
Giving up the right to PMO
Waiting planning events and
Monitoring plan merging deadlocks
Occurence of
planning events
3
Waiting for the right to perform a
Plan Merging Operation
Got the right
PMO deferred
4
Collecting neighbors coordination plans
Planning success
5
Plan Merging Operation (PMO)
Planning failure - New try aborted
8
CRPs : Planning "individual
plan" for the subsystem
6
Decision
(Possibly augmenting/ reducing the planning subsyst)
New try with more information
PMO success and result stisfactory
7
Adding the chosen "coordination plan" to the global plan
Giving up the right to PMO

Figure 4: Coordination protocol from a subsystem point of view.

intersect the trajectory of the planning aircraft (also shorter than d_t).

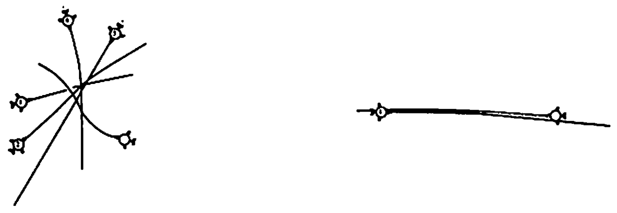


Figure 5: A solvable conflict (on the left) and an unsolvable one (on the right) for PMP.

The DAATC inherits the interesting properties of PMP which is a distributed, local and incremental scheme (see [2] for more details). However the question is : which proportion of conflicts can be solved by PMP ? Multiple complex spatio-temporal conflicts, as the left example of Figure 5, may be solved by plan merging (see also [1]). However, many spatio-temporal conflict can obviously not be solved without changing the geometry of the trajectories. The conflict on the right example of Figure 5 cannot be solved by any priority establishment. Clearly, at least one of the paths has to be modified. That is one of the reasons why we want DAATC to try first solving conflicts at the spatio-temporal level, using all the degrees of freedom of the system to apply the available conflict resolution procedures. Therefore, regarding the protocol, the basic PMP is completed by the addition of the loop including the states $\{4, 5, 6, 8\}$ of the automaton (see Figure 4). At each iteration, informations obtained on the type of conflicts that may appear can be used to adapt the choice of the conflict resolution procedure. As a consequence, the planning aircraft may evolve to different planning *subsystems* before a satisfactory plan is merged. This is not contradictory with the exclusive right to PMO since planning operation is still unique. Notice that the conflict resolution procedures can be indifferently on any of the affected aircraft or even on ground.

However, as long as a general n aircraft conflict resolution procedure does not exist, there will always be cases that DAATC will not be able to solve without a human help. In the same way as PMP, these cases are detected automatically and the system can wait in safety for an external solution. The idea here is that an individual plan is merged into the global one only if we have the guarantee that it can be executed until its end. Therefore, cases that cannot be solved automatically will come out as plan merging deadlocks rather than execution failures. Typically, a plan $\mathcal{P}^0$ that cannot be merged is a plan that is constrained by the last state $\mathcal{S}^1_{end}$ of another plan $\mathcal{P}^1$. Indeed, since the state following $\mathcal{S}^1_{end}$ has not been planned yet, we are not sure that the aircraft $\mathcal{A}^1$ will be able to leave $\mathcal{S}^1_{end}$. Therefore we are not sure that $\mathcal{P}^0$ can be executed completely. In this case $\mathcal{P}^0$ is not merged and the aircraft $\mathcal{A}^0$ wait for $\mathcal{A}^1$ to plan further than $\mathcal{S}^1_{end}$ before trying again a PMO. Later on, $\mathcal{A}^1$ may be also forced to wait for a planning operation of another aircraft. In normal cases, $\mathcal{A}^0$ will be eventually able to merge its plan. However, in problematic cases $\mathcal{A}^0$ detects that it is indirectly waiting for itself. The famous PMO deadlock is then reached and an external help is asked by the DAATC. These operations correspond to the states $\{9, 10\}$ of the protocol automaton (see Figure 4).

4.2 The consistency of the global plan

In the decentralized protocol presented above, each aircraft (or subsystem) designs and stores its own coordination plan. The global plan of the system which is the union of all coordination plans is never built. A legitimate question is how to be sure that the set of all priority constraints is a coherent one ?

Definition 3 *To each global plan, one can associate a* **scheduling graph** *which is an oriented graph whose nodes are the transitions of the plan and whose edges correspond to the priority constraints between the transitions.*

Definition 4 *A plan is called a* **consistent plan**, *if its scheduling graph is a Directed Acyclic Graph (DAG).*

For example, the left plan on Figure 3 is inconsistent. It is easy to check that the constraints lead to a cycle of the type : "T^1_j must happen before T^1_j". This plan runs into a execution deadlock. In order to prevent the apparition of cycles, PMP solution is to always give the priority to pre-existent plans. The basic PMP consider only individual plan merging. We have seen that in our protocol, due to the integration of conflict resolution procedures, plan merging can be generalized to a subsystem. However an approach similar to the one of PMP can still be applied.

Indeed, imagine that a Subsystem individual Plan (SP) has to be merged into the Global Plan (GP). The subsystem is composed of aircraft and SP includes the individual plans of these aircraft. For each individual plan, consider the transition corresponding to the leaving of its last state. Consider now F_s as the set of these final transitions of SP. In the same way, F_g designates the set of the final transitions of GP.

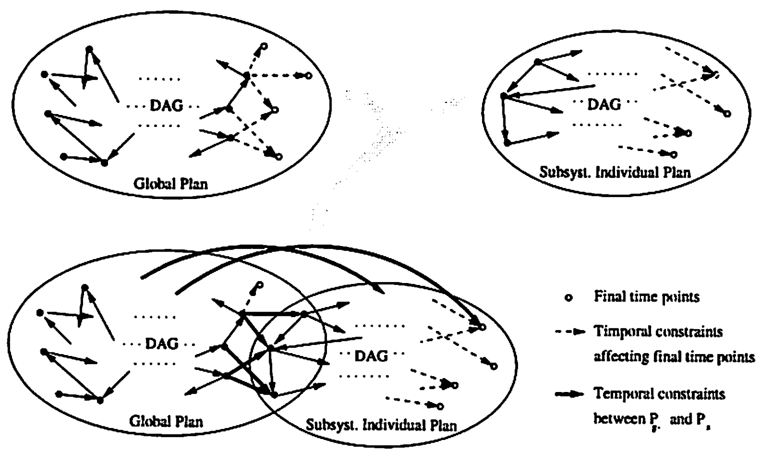


Figure 6: Unidirectional edges between two disjoint DAG create no directed cycle

Proposition 1 *If our coordination protocol represented in Figure 4 respects the following rules while merging a subsystem plan into the global plan then, the current global plan of the system is always consistent. These rules are :*

1. *A subsystem plan is considered as unacceptable if it contains any priority constraints stemming from F_s. Obviously, it has also to be consistent.*
2. *If a priority order is required for the merging of the subsystem plans, the precedence is given to the global plan. In other words, in the scheduling graph, edges should be oriented "from" the transitions of the previous global plan "towards" the ones of the subsystem plan.*
3. *A given subsystem plan cannot be merged to the global one as long as it implies giving a priority to a point of F_g.*

Proof : Let us prove this result by recurrence. The first global plan of the system is simply the first acceptable individual plan designed for one of its subsystems. The rule (1) implies that the first global plan of the system respects the two assumptions of recurrence, namely : the global plan is a DAG and no temporal constraint stems from F_g. Suppose that at a given step, the global plan verifies these two assumptions . Let us merge an acceptable subsystem individual plan SP to this global plan GP, while respecting the rules. Let us call P_s (resp. P_g) the set of the transitions of SP (resp. GP) (see Figure 6). The common points of P_s and P_g are the transitions between some of the last states of GP aircraft plans and the first states of SP ones. Therefore, $P_i = P_s \cap P_g \subset F_g$. Let us denote $P_g- = P_g - P_i$, then $P_g \cup P_s = P_g- \cup P_s$ and $P_g- \cap P_s = \emptyset$. In other words, the union of the transitions of SP and GP can be partitioned in two disjoint sets P_g- and P_s. In the scheduling graph, the edges between these two components are between P_g- and $P_g \cap P_s \subset F_g$. From the recurrence assumption on F_g, all these edges are oriented from P_g- to P_s. Thus, the scheduling graph corresponding to the union of SP and GP has two parts which are DAG by assumption, and the edges between these two parts have all the same orientation. Therefore, the whole graph is a DAG. Let us now add the edges due to the plan merging operation. If SP is merged following the rule (2) and (3), then all the edges corresponding to PMO stem from P_g- towards P_s. Then again, the final scheduling graph is a DAG. Furthermore, the set of final transitions of the new global plan is :

$$F_g := F_s \cup (F_g - P_s)$$

The rule (1) and the assumption on the original global plan imply that no temporal constraints stem from F_s nor from $(F_g - P_s)$. Therefore, the assumptions of recurrence are verified by the new global plan : it is a DAG and no temporal constraints stem from its F_g.□

In other respects more than the basic PMP, we want to be able to change some parts of the current global plan. This can be done in two steps by cancelling the final part of some aircraft plans and reprocessing a PMO for the substituting parts. The following proposition gives the rule to respect in order to maintain the consistency of the global plan.

Proposition 2 *Given a previously merged coordination plan $\mathcal{P}$ of an aircraft $\mathcal{A}$, a final portion of $\mathcal{P}$ can be removed from the global plan (possibly for a substitution) if no priority constraint stems from the exiting transition of the last remaining state of $\mathcal{P}$.*

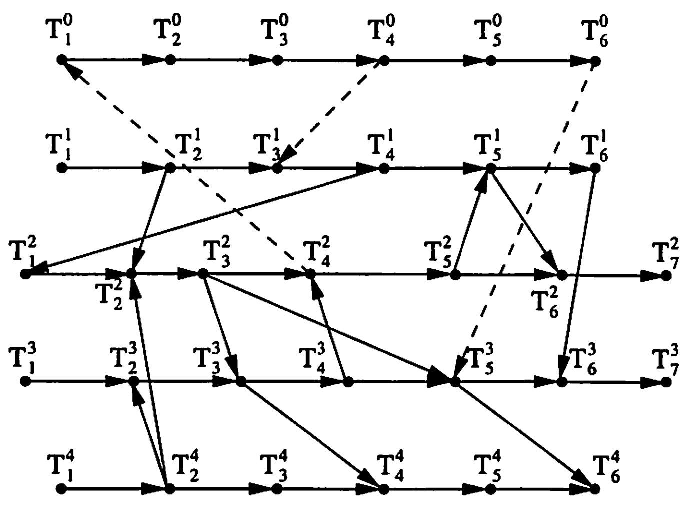
The proposition 1 guarantees that there exists an easy way to establish consistent priority orders for plan merging. If proposition 1 allows us to design safe and feasible plans, it does not include efficiency considerations and even imposes some limitations regarding the efficiency. The next section deals with efficiency issues. New rules for plan merging are presented which are computationally more complex but lead to more efficient plans.

5 Efficiency issues

In the scheme presented so far, DAATC tries first to solve conflicts at the spatio-temporal level using available conflict resolution procedures. During this phase, efficiency is taken into consideration by evolving iteratively towards the most efficient available planner (see Figure 4). However, the remaining conflicts are solved by plan merging, so far devoid of any efficiency considerations. This section aims to lead to an efficient scheduling method.

Consider the case in which an aircraft $\mathcal{A}^1$ tries to merge its best new partial plan which is in conflict with the current plan of aircraft $\mathcal{A}^2$. Imagine that this conflict is due to the fact that $\mathcal{A}^1$ path intersects at its beginning the end of $\mathcal{A}^2$ path. According to Proposition 1, $\mathcal{A}^2$ has precedence over $\mathcal{A}^1$ which has to wait in a holding pattern at the beginning of its path, the passage of $\mathcal{A}^2$, whereas it could have time to pass before. Clearly in this case, it is more efficient to leave the precedence to $\mathcal{A}^1$ but then, we do not respect Proposition 1 and therefore there is no more guarantee to maintain the global plan consistent. That means we have to check that after the merging of $\mathcal{A}^1$ plan, the global plan is still a DAG. The main advantage of a protocol respecting Proposition 1 is that the absence of DAG is automatic and therefore agents simply merge their plans without having to build or store the whole global plan. Each aircraft only has to know its own plan and the set of priority constraints directly linking it with other plans. Plan merging is then a fully local and distributed operation. Now if for sake of efficiency, we allow the protocol to accept priority constraints in both directions between the global and the merging plan, the consistency checking will no longer be a local operation. If we want it to remain distributed, more informations have to be stored and updated by each agent. We obviously want to avoid too much communication between agents. Therefore the extreme case in which each agent has to store and update the whole global plan (large communication for systematic updating) as well as the case of each agent storing only its individual plan and the corresponding constraints (large communication for checking the DAGs) are to be avoided. In Appendix A a compromise is presented by defining the distributed Data Structure that should be stored by each agent in order to detect directed cycles in the global plan.

Thus, DAATC has more allowed combinations of priority orders. If this helps to design more efficient schedules, we still have to find a way to do it. For this purpose, we formalize the general



$\mathcal{D}_4^1 = \{T_4^2\}; \mathcal{D}_6^1 = \{T_6^3\}$
$\mathcal{I}_4^1 = \{T_3^2\}; \mathcal{I}_1^1 = \{T_1^4\}$

$\mathcal{D}_2^2 = \{T_2^3, T_4^4\}; \mathcal{D}_3^2 = \{T_4^1, T_6^3\}$
$\mathcal{I}_5^2 = \{T_3^3, T_2^4\}; \mathcal{I}_4^2 = \{T_4^1, T_1^4\}$

$\mathcal{D}_2^3 = \{T_4^4\}; \mathcal{D}_3^3 = \{T_5^2\}; \mathcal{D}_4^3 = \{T_6^4\}$
$\mathcal{I}_6^3 = \{T_6^1, T_3^2\}; \mathcal{I}_4^3 = \{T_2^2\}; \mathcal{I}_2^3 = \{T_2^2\};$
$\mathcal{I}_1^3 = \{T_2^4\}; \mathcal{I}_0^3 = \{T_1^4\}$
$\mathcal{D}_1^4 = \{T_0^3, T_1^1, T_4^2\}; \mathcal{D}_2^4 = \{T_1^3, T_5^2\}$
$\mathcal{I}_6^4 = \{T_4^3, T_2^2\}; \mathcal{I}_4^4 = \{T_2^3, T_2^2\}$

Figure 7: An example of the scheduling graph $\mathcal{G}$ and the associated Data Structure explained in Appendix A.

efficiency problem as an optimization problem and suggest an approach to improve efficiency by defining a cost function.

5.1 Formalizing the efficiency problem

Given a global plan, let us call $\mathcal{G}$ the associated scheduling graph (see Figure 7). Recall that the nodes T_i^k of the graph $\mathcal{G}$ characterize so far the occurrence of an event (transition of the aircraft $\mathcal{A}^k$ from state $\mathcal{S}_i^k$ to $\mathcal{S}_{i+1}^k$) rather than an explicit clock instant. The constraints represented by the edges of $\mathcal{G}$ correspond to a set of precedences existing between the occurrence of these events. The "horizontal" constraints (see Figure 7) of $\mathcal{G}$ are rather hard since they correspond to the individual paths followed by aircraft and at the plan merging step, these paths are supposed fixed. These constraints are of the type :

$$\forall k, \ \ i < j \Longrightarrow 0 < T_i^k < T_j^k \tag{1}$$

The "vertical" constraints are the ones established during the plan merging between conflictual states. For these constraints, there is a flexibility on the choice of the priority holder , as long as the global plan remains consistent. Imagine that two states $\mathcal{S}_i^k$ and $\mathcal{S}_j^l$ are conflictual. The interval of time spent by the aircraft $\mathcal{A}^k$ in $\mathcal{S}_i^k$ is $[T_i^k, T_{i+1}^k]$. The conflict between states leads to a constraint of the type :

$$[T_i^k, T_{i+1}^k] \cap [T_j^l, T_{j+1}^l] = \emptyset \iff (\ T_{i+1}^k < T_j^l \ \ or \ \ T_{j+1}^l < T_i^k\ ) \tag{2}$$

However, these constraints guaranteeing safety allow a large freedom in the choice of the exact values of T_i^k. Yet, the performances of the system are closely dependent on the values of T_i^k. Actually, there exist other quantitative constraints on T_i^k. Indeed, the time spent by each aircraft in a state $\mathcal{S}_i^k$ (i.e. physically in the airspace cell $\mathcal{C}_i^k$) has a lower bound δ_i^k imposed by the maximal permissible velocity and the length of $\mathcal{C}_i^k$, weather condition, etc and an upper bound Δ_i^k imposed by the fuel! These constraints are of the type :

$$\forall k, i \quad \delta_i^k < T_{i+1}^k - T_i^k < \Delta_i^k \tag{3}$$

The performances of the system depend on a combination of factors. The most important of them are energy consumption and the respect of the departure and arrival time. These factors are directly related to the time spent by aircraft in crossing different cells. The performance of the synchronization scheme can be measured by a cost, only function of T_i^k :

$$J(T_0^0, \cdots, T_i^k, \cdots, T_p^n)$$

Indeed, recall that at this stage of our scheme, we assume that the aircraft paths are the best that we were able to find with the available planners and therefore they are fixed. Improving performances in this case corresponds to playing with the last degree of freedom which is time. That means optimizing the time parameterization of these paths while avoiding conflicts. Assuming that we know the optimal time parameterization of the path crossing a cell C_i^k in a given time interval $[T_i^k, T_{i+1}^k]$, the optimal performances of the system correspond to the optimal scheduling (choices of the time values of T_i^k) obtained by minimizing the cost function J under the constraints (1),(2) and (3).

5.2 Optimization strategy

Solving the general optimization problem above presents a major difficulty which is the nonconvexity of the solution space (the space defined by the constraints). However, if we fix the vertical constraints of our graph for a given instance of the problem (i.e eliminating *or* in constraint (2)), we obtain a convex solution set. Imagine that a partial plan $\mathcal{P}$ has to be merged in the global plan. Imagine that $\mathcal{P}$ has m states which are in conflict with other individual plans. To each conflict corresponds one constraint of the type (2). Some number m' of these states correspond to potential conflicts between trajectories computed by a CRP. Therefore the choice of the priority holder is imposed by the CRP. Notice that in execution time, these m' states should not meet any conflict. The corresponding vertical constraints are just set to guarantee safety. By choosing the priority holder for each of the remaining $m - m'$ conflicts and maintaining it unchanged for the pre-existent conflicts, we will have $2^{m-m'}$ versions for the graph $\mathcal{G}$. Some of these versions may not be valid. Indeed, the graph may not be DAG. These cases can be detected easily in advance and eliminated, using the data structure introduced in Appendix A. For each of the remaining versions of $\mathcal{G}$, the constraints of the corresponding optimization problem define a convex set. Solving the *elementary optimization problem* for each version and comparing the results, leads to the optimal way of merging $\mathcal{P}$. However, the practical solvability of each of the elementary optimization problems depends on the expression of the cost function J. In the following sections, we propose such an expression and show that our elementary optimization problem is equivalent to a Quadratic Programming problem which is known to be solvable using some variation of the Simplex algorithm.

Notice that if the rule 2 of Proposition 1 has disappeared, we maintain the two other rules. In this way we keep detecting deadlock situations at the planning level. Furthermore, in non deadlock situations, we have the guarantee to possess at least one consistent way of merging the last plan. Indeed, it suffices to merge it by respecting also the rule 2 of Proposition 1.

In other respect, in our scheme, a plan is merged incrementally through a sequence of partial plans computed for a reasonably short horizon. Therefore, the number m of conflicts mentioned in the above paragraph (and thus the number of elementary optimization problems) remains reasonable from a computational point of view. Also, the more problems solved at the spatio-temporal level (m'), the less elementary optimization problems to consider. Another interesting aspect of the incremental plan merging is that weather conditions can be assumed predictable along the short partial path. This knowledge can be used to define a cost function in which weather conditions are not needed to be taken as disturbance anymore.

Finally, notice that the problem described above is not exactly the general one since we have neglected the flexibility of the vertical branches[2] of the graph pre-existing the merging of the new branch. In other words, the precedences chosen to solve previous set of conflicts were maintained fixed. From a theoretical point of view, the neglected flexibilities can also be taken into account in order to find the real optimal scheduling. However, due to the combinatory aspect of our approach the general problem has a higher computational cost. Indeed, taking into account the flexibility of a vertical branch may double the number of elementary optimization problems that we have to solve. Therefore, according to the computational power, one has to chose a depth level in the graph around the new merging branch, after which the vertical branches are assumed fixed. This level can vary from the merging partial plan (as the example of the first paragraph) to the whole global plan. In the following we present the case of the general problem.

5.3 The cost function

A plan of an aircraft $\mathcal{A}^k$ consists in transiting through a sequence of cells $\mathcal{C}_i^k$. The entering instant and the exit instant of $\mathcal{C}_i^k$ are respectively given by T_i^k and T_{i+1}^k in $\mathcal{A}^k$ plan. Let us call τ_i^k the time spent by $\mathcal{A}^k$ to cross $\mathcal{C}_i^k$. Then :

$$\tau_i^k = T_{i+1}^k - T_i^k \text{ and } \delta_i^k < \tau_i^k < \Delta_i^k$$

see constraint (3). We can reasonably assume that the energy cost of crossing $\mathcal{C}_i^k$ is a function of τ_i^k. Indeed, the path is fixed and the aircraft trajectories are rather smooth. Therefore the transit time gives an estimation of the average velocity which gives an estimation of the energy consumption. The weather conditions are assumed known since the incremental planning leads to reasonably short time considerations. The energy consumption graph can be represented as on Figure 8. The expression of this function depends on the path and the local weather conditions but in general, the cost decreases as τ_i^k increases (i.e the average velocity decreases). After some point, the cost increases again since the aircraft cannot slowdown eternally and have to start holding patterns. We denote $c_i^k(\tau_i^k)$ a piecewise-linear convex approximation of the energy cost of crossing the cell $\mathcal{C}_i^k$. The energy cost of the aircraft $\mathcal{A}^k$ executing a plan $\mathcal{P}^k$ is then :

$$E^k(\cdots, \tau_i^k, \cdots) = \sum_i c_i^k(\tau_i^k)$$

Where i indexes the cells that remain to be crossed by $\mathcal{A}^k$. The energy cost of the system executing a global plan is then the sum of the individual costs of the n aircraft composing the system :

$$E(\cdots, \tau_i^1, \cdots, \tau_i^k, \cdots, \tau_i^n, \cdots) = \sum_k E^k(\cdots, \tau_i^k, \cdots) = \sum_{i,k} c_i^k(\tau_i^k)$$

Notice that the performances of an ATMS cannot be measured only in terms of energy consumption. Actually, c_i^k can integrate more information than just energy consumption (respecting the planned cell crossing time, ergonomy, aerodynamic constraints on the airplane body,...). Each aircraft can make its own compromise of different criteria in order to find the optimal crossing time

[2]The *or* in constraint (2).

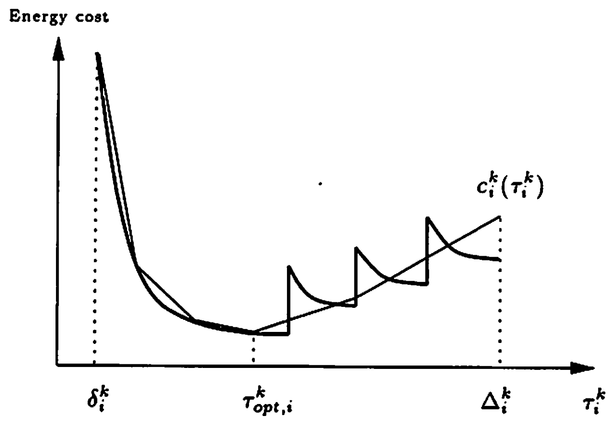


Figure 8: The basic cost function $c_i^k(\tau_i^k)$.

$\tau_{opt,i}^k$ of a given cell C_i^k. Afterwards, a cost function is built around this minimum and c_i^k will be a piecewise linear approximation of this function. The only assumption that we impose on c_i^k is to be convex. The following optimization can still be applied if this constraint is relaxed but then, we can only guarantee to find a local minimum.

If we express one of our elementary optimization problems of Section 5.2 in terms of τ_i^k rather than T_i^k and using the objective function E, we get :

$$\text{Minimize} \quad E(\cdots,\tau_i^1,\cdots,\tau_i^k,\cdots,\tau_i^n,\cdots) = \sum_{i,k} c_i^k(\tau_i^k)$$

Under the constraints :

$$\begin{aligned} \delta_i^k &< \tau_i^k < \Delta_i^k & \forall k,i \quad (4)\\ T_0^k + \tau_0^k + \cdots + \tau_i^k &< T_0^l + \tau_0^l + \cdots + \tau_j^l & \text{for some } k,l,i,j \quad (5) \end{aligned}$$

The constraint (5) concerns the conflictual states of different individual plans (see constraint (2)). We use T_0^k (the departure time of $\mathcal{A}^k$) in constraint (5) but more generally this constant of the problem is the last reached time point of $\mathcal{A}^k$ at the moment of optimization.

If most of the performance criteria can be taken into account in this way, a major one is missing. Indeed, if an aircraft is delayed with respect to its nominal trajectory because of a conflict avoidance, our optimization will not tend to catch up this delay later (which is normal if we consider for example that we minimize the energy consumption). Therefore, we complete the objective function with a new term to get :

$$F(\cdots,\tau_i^1,\cdots,\tau_i^k,\cdots,\tau_i^n,\cdots) = \underbrace{\lambda\sum_k (T_0^k + \sum_i \tau_i^k - T_f^k)^2}_{\textit{delay cost}} + \underbrace{\sum_{i,k} c_i^k(\tau_i^k)}_{\textit{energy and other costs}}$$

Where T_f^k is the desired arrival time at the end of the current partial plan of the aircraft $\mathcal{A}^k$. The constant λ is the converting factor between the cost in delay and other costs.

Now, let us note $p_{i,k}$ the number of linear pieces of c_i^k and $d_1^{i,k},\cdots,d_{p_{k,i}}^{k,i}$ the linear expressions of each piece. Since c_i^k is convex :

$$c_i^k(\tau_i^k) = \max\left(d_1^{i,k}(\tau_i^k),\cdots,d_{p_{k,i}}^{k,i}(\tau_i^k)\right)$$

Therefore, our optimization problem becomes :

$$\text{Minimize} \quad F(\cdots,\tau_i^1,\cdots,\tau_i^k,\cdots,\tau_i^n,\cdots) = \lambda\sum_k (T_0^k+\sum_i \tau_i^k - T_f^k)^2 + \sum_{i,k} \max\left(d_1^{i,k}(\tau_i^k),\cdots,d_{p_{k,i}}^{k,i}(\tau_i^k)\right)$$

Under the constraints (4) and (5). Now, one can prove that this problem is equivalent to the following one :

$$\text{Minimize} \quad F(\cdots,\tau_i^1,\cdots,\tau_i^k,\cdots,\tau_i^n,\cdots) = \lambda\sum_k (T_0^k+\sum_i \tau_i^k - T_f^k)^2 \ + \ \sum_{i,k} y_i^k$$

Under the constraints :

$$\begin{array}{rcll}
\delta_i^k & < & \tau_i^k < \Delta_i^k & \forall k,i \qquad (6)\\
T_0^k+\tau_0^k+\cdots+\tau_i^k & < & T_0^l+\tau_0^l+\cdots+\tau_j^l & \text{for some } k,l,i,j \qquad (7)\\
y_i^k & \geq & d_j^{k,i}(\tau_i^k) & \forall k,i \text{ and } \forall j, 1<j<p_{k,i} \qquad (8)
\end{array}$$

The interesting point here is that the last expression of our optimization problem corresponds to a Quadratic Programming problem. Indeed, the constraints are linear and the objective function can be transformed into $A^t x - \frac{1}{2}x^t B x$ where x is the variables vector and B is a symmetric semi-definite fixed matrix. Thanks to an algorithm proposed by Wolfe [16] and its generalization [7], a modified version of the powerful simplex algorithm (able to deal with thousands of variables) can solve this problem in a finite number of steps.

6 Fairness issues

One of the specifities of the ATM problem is that different agents are not aiming to achieve a common goal. They are rather in competition for the resources to achieve their individual goals. A good ATC must be not only safe and efficient but also *fair* in the sense that it should not penalize one agent (or one air company) more than the others. Therefore, an optimal solution (in terms of energy, time, etc) where all the cost is borne by the same agent is less acceptable than a near optimal solution where the cost is equally shared between agents. In order to achieve fairness, we introduce some weight coefficients μ^k in the objective function :

$$F(\cdots,\tau_i^1,\cdots,\tau_i^k,\cdots,\tau_i^n,\cdots) = \lambda\sum_k \mu^k(T_0^k+\sum_i \tau_i^k - T_f^k)^2 \ + \ \sum_k \mu^k E^k(\cdots,\tau_i^k,\cdots)$$

A relative increase in the coefficient μ^k will decrease the aircraft $\mathcal{A}^k$ risks to be penalized. We can consider an aircraft $\mathcal{A}^k$ as an individual and modify the values of μ^k for successive plan merging of its partial plans, with respect to the past penalties. If the fairness is not respected at an individual level[3], we can also consider $\mathcal{A}^k$ as a member of a family (the aircraft of the same air company) and modify μ^k in order to establish fairness between companies at the end of the day.

7 Recapitulation and conclusion

The Distributed Automatic Air Traffic Controller DAATC presented in this paper has the main advantage of guaranteeing the safety of the whole air fleet in a systematic way, independently on the

[3]Actually, this corresponds to the fairness for passengers.

shape of the aircraft trajectories. The aircraft plans are designed and validated incrementally, by small pieces. Planning the future partial plan is done in parallel with the execution of the current one. First, available conflict resolution procedures are used to solve as many conflicts as possible in an efficient way. The DAATC can integrate any type of current or future planner. Notice that at this step, even incomplete planners can be used for a try. Then, DAATC verifies the validity of these partial solutions in presence of other aircraft of the system. The remaining conflicts are solved by establishing priority rules between different conflictual aircraft. The efficiency at this level consists in finding the best distribution of priorities and the best scheduling between aircraft. For this purpose, the results of a family of elementary optimization problems are compared. In the same way as human controllers, DAATC may lead some aircraft to execute some holding pattern. However, with DAATC optimized scheduling, each aircraft knows when it has to be at each via point in order to reduce the waiting periods and to optimize the execution of the current global plan. Fairness issues stem also from these optimizations. Notice that safety has precedence on efficiency and is guaranteed by the respect of the distribution of priorities. In other words, the scheduling has only an indicatory value. Each aircraft tries to respect its own. However, if execution time delays induce contradiction between the priority rules and the scheduling, the earliest prevails and guarantees the safety and the complete execution of the global plan. Indeed, even if the scheduling is not respected, the cell decomposition allows to maintain priority orders by the means of holding patterns. Thus DAATC is robust with respect to execution delays.

Numerous conflicts can be solved automatically by DAATC, using simple planners and synchronizing their designed paths between each other. The ability of this scheme in helping to solve air traffic conflicts can be partly justified by the fact that even if we do not possess powerful planners, we have a large maneuvering airspace at our disposal. However, integration of human help in DAATC is still necessary. Indeed, even if safety is fully handled by DAATC, human interventions are required to guarantee the completeness of the controller, in case that available planners and plan merging fail in solving a conflict. Hopefully, this situation will become more and more rare with the progress of conflict resolution procedures.

Aknowledgements

This work was supported by the Army Research Office under grant DAAH 04-96-1-0341.

The authors would like to thank George J. Pappas, Claire Tomlin, Thierry Vidal and John Lygeros for their stimulating discussion.

References

[1] L. Aguilar, R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. "Ten Autonomous Mobile Robots (and even more) in a route network like environment". In Proceedings of *IEEE Intelligent Rob. and Sys.*, IROS'95, Pittsburgh, Pennsylvania, 1995.

[2] R. Alami, F. Ingrand, and S. Qutub. "Plan Coordination and Execution in Multi-Robots Environment". In Proceedings of *Int. Conf. Advanced Robotics* ICAR'97, Monterey, 1997.

[3] R. Alami, F. Robert, F.F. Ingrand and S. Suzuki. "A paradigm for plan-merging and its use for multi-robot cooperation". In *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, Texas, 1994.

[4] D.J. Brudnicki and A.L. McFarland. "User Request Evaluation Tool (URET) conflict probe performance and benefits assessment". In Proceedings of the U.S.A./Europe ATM Seminar, Eurocontrol, Paris, 1997.

[5] J. Canny and J. Reif. "New lower bound techniques for robot motion planning problems". In Proceedings of the *28th Annual IEEE Symposium on Foundations of Computer Science*, pages 49-60, 1987.

[6] H. Erzberger, T.J. Davis, and S. Green. "Design of Center-TRACON Automation System". In Proceedings of the *AGARD Guidance and Control Symposium on Machine Intelligence in Air Traffic Management*, pages 11.1-11.12, Berlin, Germany, 1993.

[7] J.D. Canon, C.D. Cullum and E. Polak,"Theory of Optimal Control and Mathematical Programming", McGrawHill, 1970.

[8] M. Erdman and T. Lozano-Perez. "On multiple moving objects". *Algorithmica, 2:477-595*, 1987.

[9] J. Košecká, C. Tomlin, G.J. Pappas, and S. Sastry. "Generation of Conflict Resolution Maneuvers for Air Traffic Management". In Proceedings of *Intelligent Robots and Systems*, IROS'97, Grenobe, France, Sept. 1997.

[10] J. Krozel and M. Peters. "Conflict Detection and Resolution for Free-Flight". In *Air Traffic Control Quarterly Journal*, Special Issue on Free Flight, 1997.

[11] J. Krozel, T. Mueller, and G. Hunter. "Free flight conflict detection and resolution analysis". In Proceedings of the *AIAA Guidance, Navigation and Control Conference, AIAA-96-3763*, San Diego, CA, August 1996.

[12] M. Naimi, M. Trehel and A. Arnold. "A log(n) distributed mutual exclusion algorithm based on path reversal". *Journal of Parallel and Distributed Computing*, 34, 1996.

[13] J.-H Oh and E. Feron. "Fast detection and resolution of multiple conflicts for 3-Dimensional free flight". In Proceedings of the *IEEE Conference on Decision and Control.* San Diego, CA, 1997.

[14] Radio Technical Commission for Aeronautics. "Final report of RTCS Task Force 3: Free flight implementation". *Technical report*, RTCS, Washington DC, October 1995.

[15] C. Tomlin, G.J. Pappas, and S. Sastry. "Noncooperative Conflict Resolution". In Proceedings of *IEEE Int. Conf. on Decision and Control*, San Diego, California, Dec.1997.

[16] P. Wolfe, "The Simplex Method for Quadratic Programming", In Econometria, 27, 1959.

[17] Y. Zhao and R. Schultz. "Deterministic resolution of two aircraft conflict in free flight." In Proceedings of the *AIAA Guidance, Navigation and Control Conference, AIAA-97-3547*, New Orleans, LA, August 1997.

Appendix A : Distributed data structure for DAG verification

Given $\mathcal{G}$ the scheduling plan of the system. As for many other multi-agent systems, the graph $\mathcal{G}$ has a special structure due to the fact that individual plans corresponds to sequences of states (or actions). Therefore, posing :

Definition 5 *: T_i^k denotes the i-th time point in the individual plan of the aircraft $\mathcal{A}^k$*

we have :

$$\forall i, j, k \;\; i < j \implies T_i^k < T_j^k \tag{9}$$

Thus, $\mathcal{G}$ looks like the graph depicted on Figure 7. The "horizontal" branches of $\mathcal{G}$ (characterized by the relation (9) above) are implicitly known by all aircraft. So, we should mainly store the "vertical" branches. However as we will see, if our goal is only to detect directed cycles in $\mathcal{G}$, all these branches are not of interest for a given aircraft.

Imagine that for each node T_i^k of the horizontal branch k (corresponding to $\mathcal{A}^k$ plan), we wish to determine all the nodes of $\mathcal{G}$ that may be reached from T_i^k. Notice that :

- If T_j^l can be reached, then all T_h^l for $h > j$ can also be reached.
- All the nodes reachable from T_j^k with $j > i$ are also reachable from T_i^k.

Therefore, a good data structure should avoid these redundancies. Let us define the elementary brick of this structure :

Definition 6 *: Given a node T_i^k, $\mathcal{D}_i^k$ denotes the set of time points T_j^h (one at most for each horizontal branch $h \neq k$) fulfilling the following conditions :*

- *T_j^h is the least time point of the horizontal branch h, reachable from T_i^k,*
- *T_j^h is not reachable by other time points $T_m^k > T_i^k$. That means, there is no $m > i$ and $p \leq j$ such that $T_p^h \in \mathcal{D}_m^k$.*

Notice that if no temporal constraint stems from T_i^k towards other horizontal branches, then $\mathcal{D}_i^k = \emptyset$. It is easy to verify that for a given horizontal branch k, the set of $\mathcal{D}_i^k$ $\forall i$, allows to determine the set of reachable nodes from any node T_j^k of the branch. Indeed :

$$Reach(T_j^k) = \{T_h^l \mid \exists T_p^l \in \cup_{i \geq j} \mathcal{D}_i^k \text{ with } p \leq h\} \;\cup\; \{T_i^k,\ i \geq j\}$$

Therefore, the set of $\mathcal{D}_i^k$ $\forall i$ are also enough for the aircraft $\mathcal{A}^k$ to detect by its own (in a decentralized way), any DAG including one of the time points of its plan. However, the data structure stored by each aircraft needs to be completed in order to allow the distributed updating of the set of $\mathcal{D}_i^k$.

Definition 7 *: Given a node T_i^k, $\mathcal{I}_i^k$ denotes the set of time points T_j^h (one at most for each horizontal branch $h \neq k$) fulfilling the following conditions :*

- *T_j^h is the greatest time point of the horizontal branch h, from which T_i^k can be reached,*
- *No other time points $T_m^k < T_i^k$ of the branch k can be reached from T_j^h. That means, there is no $m < i$ and $p \geq j$ such that $T_p^h \in \mathcal{I}_m^k$.*

In the same way as the set of $\mathcal{D}_i^k$ determine for the branch k, the nodes of $\mathcal{G}$ that are reachable *Directly*, the set of $\mathcal{I}_i^k$ determine the nodes that are reachable following the *Inverse* direction. In other words the nodes from which the branch k is reachable.

7.1 Generating and updating the data structure

As we have seen in ATCS, aircraft plans are built incrementally and merged to the global plan. It is easy to see that thanks to the indexing of time points, addition of horizontal branches to $\mathcal{G}$ does not require storage of any further information to our data structure. However, each vertical branch requires the generation of a new $\mathcal{D}_i^k$ or $\mathcal{I}_i^k$ and imply updating some of the old ones.

Let us explain through the example shown on Figure 7, the 3 steps of this procedure presented below. Let us imagine that the current global plan is composed of the branches $H^1, \cdots, H^4$ and the constraints between them. Each aircraft $\mathcal{A}^k$ knows its coordination plan and also the set of its $\mathcal{D}_i^k$ and $\mathcal{I}_i^k$ (see Figure 7). Now, the aircraft $\mathcal{A}^0$ starts planning (H^0 on the figure) and tries to merge its plan (dashed lines on the figure). The corresponding vertical branches can be added one after the other in any order. Let us start by the branch between T_4^0 and T_3^1 :

1. The set $\mathcal{D}_4^0$ is no longer empty and needs to be computed. From the definition of $\mathcal{D}_4^0$:

$$\mathcal{D}_4^0 \subset \cup_{i \geq 3} \mathcal{D}_i^1 \bigcup \{T_3^1\} = \{T_3^1, T_1^2, T_3^3, T_4^4, T_6^2, T_6^3\}$$

   This set can be used as such for our goal but in order to respect the definition of $\mathcal{D}_i^k$, if several time points of the same horizontal branch are present, we should only keep the least. We should also remove from it any point that can be reached from another time point of H^0 greater than T_4^0 (which can be deduced from $\cup_{i>4}\mathcal{D}_i^0$, see Definition 6). Thus :

$$\mathcal{D}_4^0 = \{T_3^1, T_1^2, T_3^3, T_4^4\}$$

2. The new branch implies that the new subset of $\mathcal{G}$ points that can now be reached from T_4^0, can be reached by all points that could reach T_4^0. The useful information is stored in a distributed way by updating some of the $\mathcal{I}_i^k$. The list of these $\mathcal{I}_i^k$ representing the reachable set from T_4^0 is directly deduced from $\mathcal{D}_4^0$ (in our case : $\mathcal{I}_3^1$, $\mathcal{I}_1^2$, $\mathcal{I}_3^3$ and $\mathcal{I}_4^4$). The time points (which must be added to these $\mathcal{I}_i^k$), representing the set of points from which T_4^0 is reachable is deduced from $\cup_{i \leq 4}\mathcal{I}_i^0$ (in our case, simply $\{T_4^0\}$). That means, $\mathcal{I}_3^1 = \{T_4^0\}$ has to be created. The set $\mathcal{I}_1^2$ has to be completed[4] by T_4^0. In the same way, T_4^0 has to be added to $\mathcal{I}_3^3$ and $\mathcal{I}_4^4$.

3. In the same way, the new branch implies that points that could reach T_4^0, can now reach a new subset of $\mathcal{G}$ points. This time the useful information is stored by updating some of the $\mathcal{D}_i^k$ deduced from $\cup_{i \leq 4}\mathcal{I}_i^0$. The set of points to be added is $\mathcal{D}_4^0$ or simply a part of it due to redundancy.

In the example above, the constraint stems from the new individual plan. Analogous operations can be obviously done for the opposite direction by changing the role of $\mathcal{D}_i^k$ and $\mathcal{I}_i^k$. In any case, detection of directed cycles is done easily during the above operations. In our example, addition of the branch $T_4^2 T_1^0$ leads to the computation of $\mathcal{I}_1^0 = \{T_4^0, T_4^1, T_4^2, T_4^3, T_2^4\}$. That means on the branch H^0, all the points until T_4^0 (including then T_1^0) can lead to T_1^0. Addition of $T_4^2 T_1^0$ creates then a cycle.

With the data structure and the procedure described in this section, the useful information for checking DAGs can be generated, stored and updated in a distributed way. As one can see, the information is very concise (individual plan, set of $\mathcal{D}_i^k \neq \emptyset$ and set of $\mathcal{I}_i^k \neq \emptyset$) which is suitable for communication. Yet, its updating is easy and checking DAGs is straightforward. Nevertheless,

[4]Unless a redundancy is noticed by consulting $\mathcal{I}_i^2$ for $i \leq 1$ (see Definition 7).

updating the structure may lead to multiple communications between the planning aircraft $\mathcal{A}^0$ and some others. In order to reduce the number of communications, different operations of the procedure can be regrouped for different constraints of the same partial plan :

1. For any new vertical branch, $\mathcal{A}^0$ asks the affected aircraft for necessary informations to create the corresponding $\mathcal{D}_i^0$ or $\mathcal{I}_i^0$. This corresponds to sending simply a set of time points.
2. Once all these informations gathered, $\mathcal{A}^0$ creates all $\mathcal{D}_i^0 \neq \emptyset$, starting by the greater time points in order to avoid the creation of redundancy. The same operation is done for $\mathcal{I}_i^0 \neq \emptyset$ starting by smaller time points. One can prove that even if the effects of new vertical branches have not been propagated in the rest of the structure, these sets of $\mathcal{D}_i^0$ and $\mathcal{I}_i^0$ are the exact and minimal ones. Creation of cycles can be checked at this point (which lead to the rejection of this priority distribution), before updating the whole structure.
3. After analyzing redundancy, $\mathcal{A}^0$ communicates to some aircraft the summary of the informations that they need to update their part of the graph.

Thus at most, each affected aircraft sends one set of data and receives one for the merging of a partial plan. Therefore, our data structure is condensed and required small amount of communication.