Automatic Planning of Ground Traffic

Charles Lesire

ONERA, Toulouse, France

Ground traffic optimization is a major issue of air traffic management: optimal ground circulation could decrease flight delays and consequently decrease costs and increase passenger wellness. This paper proposes a planning algorithm for ground traffic based on contract reservation. It deals with time and speed uncertainty to ensure the feasability of the planned trajectories while avoiding conflicts between aircrafts. The algorithm efficiency is improved using pre-computed heuristics. The different algorithm versions are evaluated through simulations of a large european airport, namely the Frankfurt airport.

I. Introduction

One of the major issues of Air Traffic Management concerns the optimization of airport traffic. Indeed, the air traffic growth is having a hard impact on airport congestion. Flight delays are obviously impacted leading to an economic interest on ground traffic optimization methods. This optimization may also take into account ecologic issues such as noise and pollution reduction.

Ground traffic optimization can hardly be performed by human controllers: managing several aircrafts moving on the airport during rush hours on quite complex taxiway networks may be difficult. It is especially the case when hard weather conditions occur (e.g. fog).

A lot of researches have tried to help ground controllers either by defining new visualization displays (DST,[1] AMAN,[2] DMAN, etc.) or by improving traffic predictivity by sharing flight data between airports and controllers (CDM[3]). The development of the ASMGC[a] aims at combining all these methods to improve ground traffic efficiency.[4]

Following the concept of ASMGC, some works have developped methods for ground traffic optimization. Gotteland[5] uses genetic algorithm to generate aircraft trajectories. The algorithm is used for real-time control of aircrafts – mainly for simulation issues – and does not provide a complete trajectory of the aircraft (usefull for pilot or controller understanding of aircraft behaviors).

Deau *et al.*[6] work on runway sequencing to help fluidifying ground traffic but does not consider the relation between runways and complete taxiway trajectories.

Constraint propagation and relaxation methods are used by van Leeuwen *et al.*[7] but no significant results are given on useability of such algorithms.

The work presented in this paper proposes an algorithm for ground traffic planning based on a classical path-finding algorithm, namely A*, improved to deal with time constraints, and time and speed uncertainty. In section II, ground traffic and uncertainty models are presented. Then the algorithm is introduced and evaluated in section III. Some improvements are then made to make the algorithm more efficient, using a more accurate heuristic (section IV.A), and a pruning algorithm to reduce the state-space (section IV.B).

II. Ground Traffic Modelling

The airport structure is represented by a graph $G = (N, E)$, where the set of nodes N represents the airport waypoints (taxiway intersections, gates, runway accesses, etc.) and the set of edges E represents the

[a] Advanced Surface Movement Guidance and Control System

taxiways connecting these nodes.

Just before arriving (for an arrival) or leaving its gate (for a departure), each flight is announced to the planning algorithm. This annoucement is defined by:

- a start point n_S (either a runway exit point or a gate);
- a start time window I_S; the window width $|I_S|$ depends on the confidence the "controller" has on the flight announcement;
- a goal point n_G (either a gate or a runway access).

Moreover, airport-related constraints are taken into account:

- a separation distance Δ between aircrafts;
- a maximal ground speed V_{max}.

The proposed algorithm does not model aircraft accelerations but uses a constant-speed movement prediction to update the aircraft path. Then a speed uncertainty δV has to be defined to manage movement noises and accelerations.

From a flight annoucement and the airport graph, the algorithm computes a **contract** $\gamma = (n_i, I_i)_{0 \leq i \leq m}$ where each node n_i is associated to a time interval I_i, such that:

- $n_0 = n_S$,
- $I_0 = I_S$,
- $n_m = n_G$,
- $\forall\, i,\ (n_i, n_{i+1}) \in E$.

Moreover, on each edge the separation with other aircrafts must be respected, and aircraft speed must be lower than V_{max}. Finally γ should be optimal, i.e. I_m should be minimal.

On this contract, each edge (n_i, n_{i+1}) is *reserved* for the aircraft between time I_i and I_{i+1}. The following algorithm is so named *Contract Reservation Algorithm* (CRA).

III. The Contract Reservation Algorithm

III.A. Algorithm and complexity

Algorithm 1 is an A*-like algorithm. The functions of the algorithm are g the cost function (i.e. the time elapsed from start point to the current node), h the heuristic (i.e. the estimated time from current point to the goal) and $f = g + h$ the resulting function, optimized during search.

The $father$ set indicates the path to current node. When the algorithm ends (line 8), the $father$ set allows to find the complete itinerary for the flight.

Other functions are used in the algorithm:

- the Next function (line 10) returns all the successors of the current node, as defined in the airport graph structure;
- the Previous function (line 11) returns the set of previous nodes in the current explored path; the associated condition is used to check that node i has not been explored before, so that loops are avoided;
- the Contract function (line 12) computes the available contract for the aircraft on node i (algorithm 2); this computation ensures separation and conflict avoidance with other existing contracts and satisfies speed constraints; when a contract is available (the interval is not empty – line 13) node i is added to the open list (lines 22 to 25) so that the search can go on; when the algorithm ends, the complete aircraft contract is saved;

Algorithm 1 The Contract Reservation Algorithm.

```
 1: $O \leftarrow \{(n_S, I_S)\}$
 2: $g((n_S, I_S)) \leftarrow \max(I_S)$
 3: $f((n_S, I_S)) \leftarrow g((n_S, I_S)) + h(n_S)$
 4: $father((n_S, I_S)) \leftarrow \emptyset$
 5: while $O \neq \emptyset$ do
 6:     $(n, I) \leftarrow \text{argmax}_{\text{argmin}_{x \in O} f(x)} g(x)$
 7:     if $n = n_G$ then
 8:         return
 9:     end if
10:     for all $i \in \text{Next}(n)$ do
11:         if $i \notin \text{Previous}(n)$ then
12:             $J \leftarrow \text{Contract}(i)$
13:             if $J = \emptyset$ then
14:                 $J \leftarrow \text{Wait}(n, I)$
15:                 if $J \neq \emptyset$ then
16:                     $g((n, J)) \leftarrow \max(J)$
17:                     $f((n, J)) \leftarrow g((n, J)) + h(n)$
18:                     $father((n, J)) \leftarrow (n, I)$
19:                     $O \leftarrow O \cup \{(n, J)\}$
20:                 end if
21:             else
22:                 $g((i, J)) \leftarrow \max(J)$
23:                 $f((i, J)) \leftarrow g((i, J)) + h(i)$
24:                 $father((i, J)) \leftarrow (n, I)$
25:                 $O \leftarrow O \cup \{(i, J)\}$
26:             end if
27:         end if
28:     end for
29: end while
```

- the Wait function (line 14) is used to compute the time the aircraft should wait at node n until a contract is available without conflict (algorithm 3); then node n is added again in the open list with time interval J (lines 16 to 19).

Equation (1) gives the algorithm complexity with $|N|$ the number of airport waypoints, L the maximal temporal length of an aircraft itinerary, B the airport branching factor (the worst number of successors by node) and C the number of contracts currently reserved.

$$\mathcal{O}(|N|^2.L.(\log(|N|) + B.C^2)) \tag{1}$$

As the length of an aircraft itinerary is generally limited, the hard points are the number of nodes $|N|$, the number of contracts C, and the branching factor B. Hence the algorithm may be slown down by large congested airports.

Next paragraph presents the algorithm behavior on a simple airport. Then more extended results on realistic simulations are presented.

III.B. A simple example

The example presented in this section illustrates the flight contracts resulting from CRA. Figure 1 shows the example graph and Table 1 the flights to plan on this graph.

Algorithm 2 The Contract function for an edge (i,j) and a starting interval I.

```
1:  $J \leftarrow [\min(I) + \frac{g(i,j)}{V_{max}}; \infty]$
2:  for all existing contract $C$ on $(i,j)$ do
3:      $\delta \leftarrow \frac{\Delta \,.\, |J-I|}{g(i,j)}$
4:      if $I$ overlaps $start(C)$ then
5:          return $\emptyset$
6:      else
7:          $\delta \leftarrow J \cup (end(C) + \frac{g(i,j)}{\delta v})$
8:          $J \leftarrow J \cap (J + \delta)$
9:      end if
10: end for
11: for all existing contract $C$ on $(j,i)$ do
12:     if $C < I$ and $C\{overlaps, <\}J$ then
13:         return $\emptyset$
14:     else
15:         $J \leftarrow J \cap (J - (C \cap J))$
16:     end if
17: end for
18: return $J$
```

Algorithm 3 The Wait function for an edge (i,j) and a starting interval I.

```
1:  $start \leftarrow -\infty$
2:  $J \leftarrow I + \frac{g(i,j)}{V_{max}}$
3:  for all existing contract $C$ on $(i,j)$ do
4:      $start \leftarrow \max(start, end(C) + \Delta \frac{\min(end(C) - start(C))}{g(i,j)})$
5:  end for
6:  for all existing contract $C$ on $(j,i)$ do
7:      if $C$ overlaps $I \cup J$ then
8:          $start \leftarrow \max(start, C)$
9:      end if
10: end for
11: if $start < I$ then
12:     return $\emptyset$
13: else
14:     if $start > I$ then
15:         return $start + \frac{g(i,j)}{\delta v}$
16:     else
17:         return $[start, \max(I)]$
18:     end if
19: end if
```

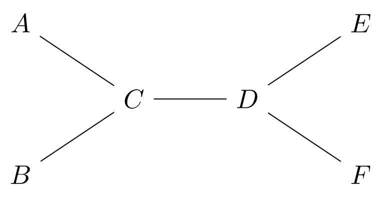

Figure 1. The example graph.

Table 1. Example flights.

Flight	Start point	End point	Start time interval
1	A	E	[-10, 10]
2	F	B	[-10, 10]
3	A	E	[0, 20]

Conflicts will mainly occur on the CD edge. The flight itineraries produced by CRA are presented in Fig. 2.

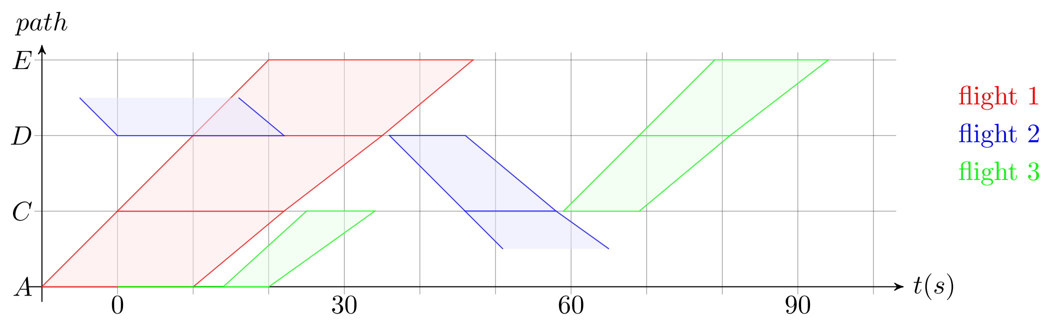

Figure 2. Flight contracts.

The red contract is reserved to flight 1. The contract width represents the available time for the aircraft, including uncertainty on speed and acceleration. The aircraft leaves point A during interval $[-10; 10]$ and arrives at E during interval $[20; 47]$.

The blue contract is reserved to flight 2: coming from point F, the aircraft has to wait at D until the DC edge is free, i.e. until filght 1 has passed. Then, from time 36, flight 2 can go on to point B.

The green contract is reserved to flight 3. At the beginning, the aircraft has to be delayed to be separated from flight 1: it can leave point A during interval $[14; 20]$. At C, the aircraft must wait until flight 2 has passed away. It finally arrives at E during interval $[79; 94]$.

III.C. Simulation and first results

The CRA is evaluated using a simulation tool (Fig. 3). Aircraft trajectories are generated using more realistic dynamics (including accelerations) so that aircrafts stay in their reserved contracts. Figure 3 shows a complex situation managed by the algorithm:

- The two green aircrafts on the left are correctly separated;
- The red aircraft has stopped to let the green aircrafts pass;
- The red aircraft is now waiting until it is correctly seperated to the upper green aircraft.

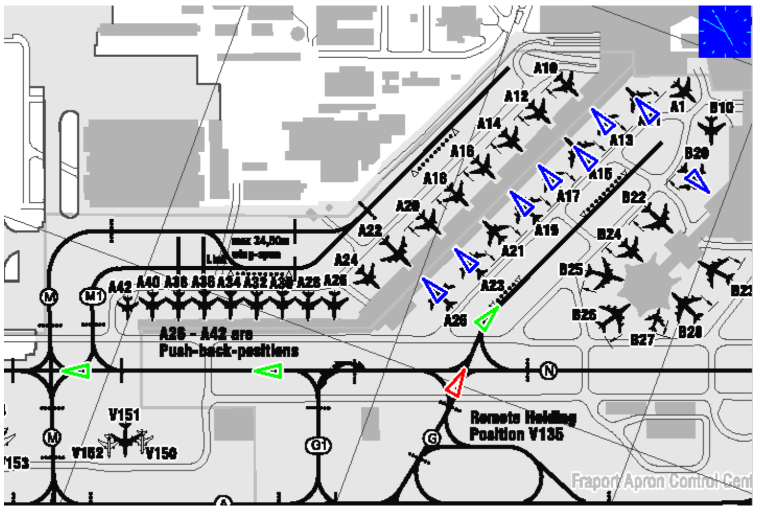
A10
A12
A14
A16
A18
A20
A22
A24
A1
A13
A15
A17
A19
A21
A23
A25
B10
B20
B22
B24
B25
B26
B27
B28
A42
A40 A38 A36 A34 A32 A30 A28 A26
A26 - A42 are Push-back-positions
V151
V152
V150
Remote Holding Position V135
Fraport Apron Control Cent

Figure 3. Simulation of ground traffic on Frankfurt airport: blue aircrafts are parked, red aircrafts are waiting and green aircrafts are moving.

The CRA algorithm has been evaluated on a one-day traffic on Toulouse-Blagnac airport. Figure 4 shows the evolution of aircraft delays according to the $|I|$ parameter (initial time uncertainty – Δ has been fixed to 50m and δV to 0). The delay evolution is linear in $|I|$, meaning that the uncertainty is taken into account but does not induce an exponentially increasing delay.

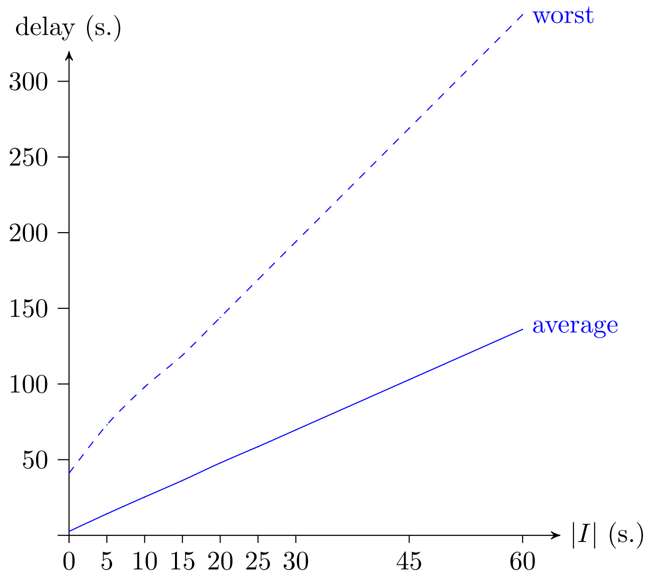

Figure 4. Average and worst delays on Blagnac airport, according to initial uncertainty.

Moreover, algorithm complexity (Fig. 5) and time computation (Fig. 6) are approximately constant according to $|I|$. Nevertheless, the worst time computation is around 30 ms on Blagnac airport, up to a quarter second.

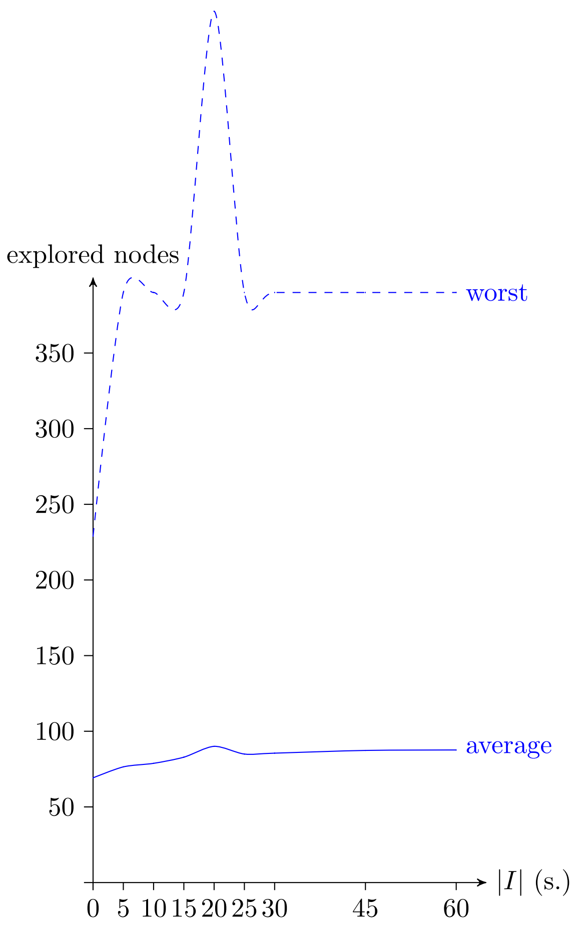

Figure 5. Average and worst number of explored nodes on Blagnac airport, according to initial uncertainty.

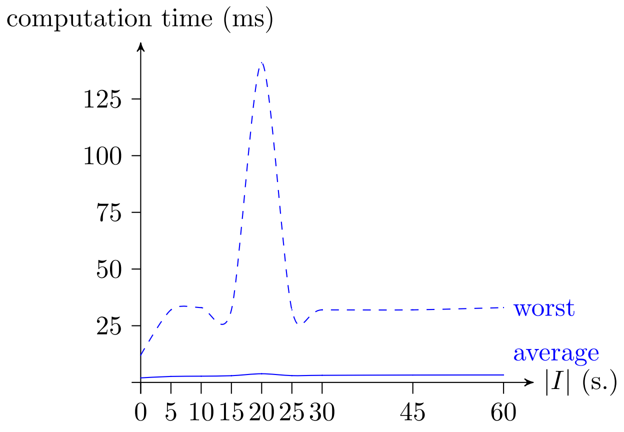

Figure 6. Average and worst computation times on Blagnac airport, according to initial uncertainty.

Results on Frankfurt airport (Fig. 7) are not so successfull: worst computation time seems not to be linear (Fig. 8). The computation time is around 1 minute for an initial uncertainty lower than 30s. These computation times are too long to have the CRA be used online, where many aircrafts must be planned in short time during rush hours.

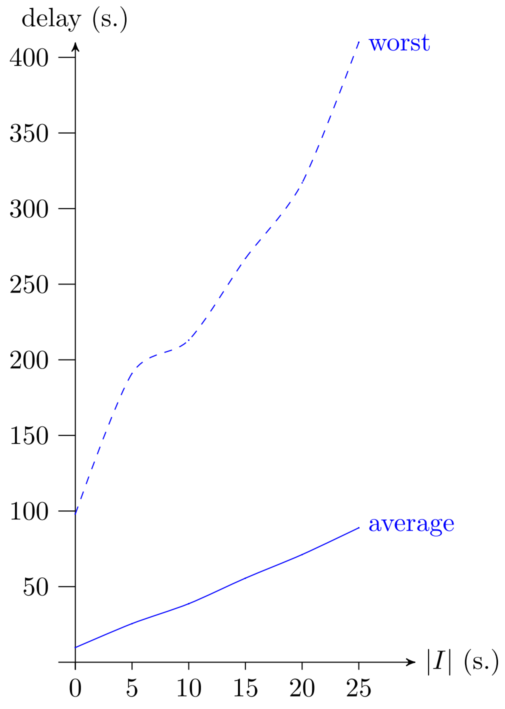

Figure 7. Average and worst delays on Frankfurt airport, according to initial uncertainty.

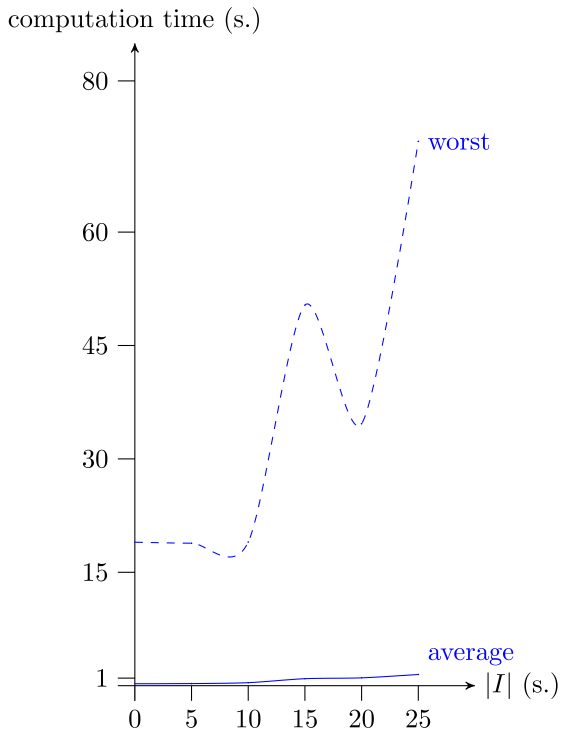

Figure 8. Average and worst computation times on Frankfurt airport, according to initial uncertainty.

IV. Improving CRA Efficiency

Two ways of improvement have been studied:

- computing a more accurate heuristic (the h function) so that the algorithm should faster find the shortest path;
- pruning the state-space by allowing only a short number of possible path per flight.

IV.A. Heuristic Computation

The heuristic used in the first CRA version (algorithm 1) was only based on direct 2D distance between points. As aircrafts have to follow roads, especially to avoid runways, the direct distance may sometime be meaningless.

IV.A.1. The Floyd-Warshall algorithm

The Floyd-Warshall algorithm (algorithm 4)[8] is used as a pre-computation of the CRA heuristic: it computes the shortest path between every pair of graph nodes. This distance is stored in matrix D. When asking for the heuristic value in CRA (algorithm 1, line 17 and 23), this matrix is used instead of the 2D distance function by getting the $D[i][goal]$ value.

Matrix D is initialized using direct distance when two nodes are directly connected (lines 2 and 3) and undefined (equal to infinity) when nodes are not connected (line 5).

Then D is updated to compute the shortest distance between every pair of nodes (line 15).

The algorithm complexity is $\mathcal{O}(|N|^3)$, but as the algorithm is launched only once per airport, its complexity is not relevant for online planning.

Algorithm 4 The Floyd-Warshall Algorithm.

```
 1: for all $i, j \in N$ do
 2:     if $(i, j) \in E$ then
 3:         $D[i][j] \leftarrow d(i, j)$
 4:     else
 5:         $D[i][j] \leftarrow \infty$
 6:     end if
 7: end for
 8: for all $i \in N$ do
 9:     for all $j \in N, j \neq i$ do
10:         if $D[i][j] + D[j][i] < 0$ then
11:             return
12:         end if
13:         if $D[j][i] < \infty$ then
14:             for all $k \in N, k \neq j$ do
15:                 $D[j][k] \leftarrow \min(D[j][k], D[j][i] + D[i][k])$
16:             end for
17:         end if
18:     end for
19: end for
```

IV.A.2. Results

Figure 9 compares results of CRA with and without the Floyd-Warshall computed heuristic on Frankfurt airport. The computation time is greatly decreased, having the algorithm usable on concrete situation when the worst computation time is lower than 20s (on a Core2 1.6GHz 2Go RAM Computer).

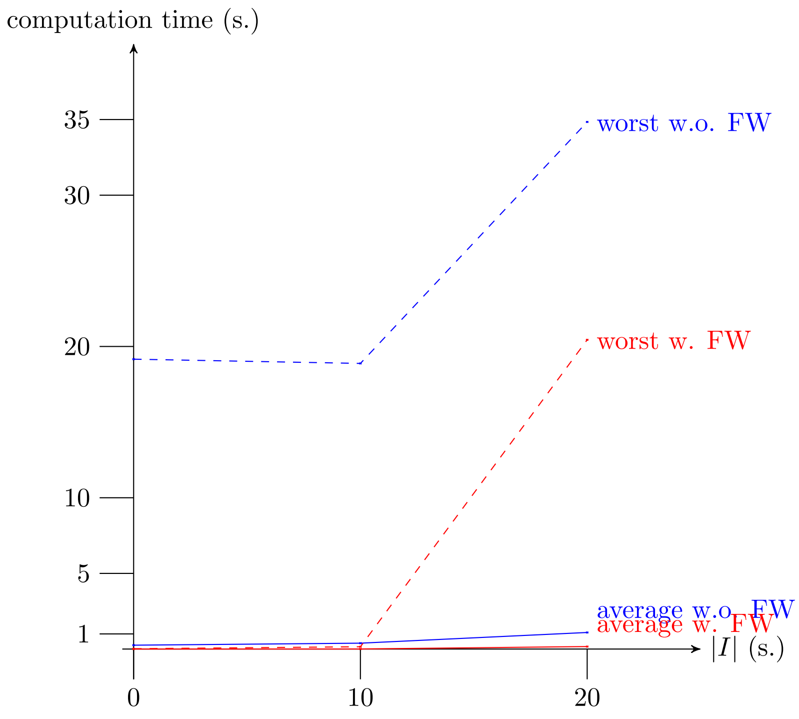

Figure 9. Comparison of computation time on Frankfurt airport using Floyd-Warshall algorithm.

We can notice that the computation time is highly reduced: about 90% on average delays, 90% to 50% on worst cases.

If the Floyd-Warshall heuristic has drastically reduced the computation time, the introduction of speed uncertainty in flight paths hardly complexifies the problem. Frankfurt flights are computed in a huge amount of time (around 2 min. up to 2 hours). The heuristic improvement is not sufficient to have the algorithm usable in an online context and other ways of improvement must be considered.

IV.B. State-Space pruning

Previous results have shown that the algorithm may be really slow on particular situations (large airports, numerous aircrafts...) One way to decrease the algorithm computation time is to prune the search graph. In this section, the MPS algorithm[9] is used to compute K possible paths per flight.

IV.B.1. The MPS algorithm

MPS (algorithm 5) computes the K shortest deviations from the start point to the goal according to the shortest path from start to goal. This shortest path must have been previously computed using for example a Floyd-Warshall algorithm[b]. When the algorithm finishes, the C set contains the desired paths.

Algorithm 5 The MPS Algorithm.

```
1: $Q \leftarrow \{(D[start][goal], goal, \emptyset)\}$
2: $C \leftarrow \emptyset$
3: **while** $|C| < K$ and $Q \neq \emptyset$ **do**
4:     $(c, n, \gamma) \leftarrow \text{argmin}_{(x,y,z)\in Q}\ a$
5:     **while** $n \neq start$ and $n \notin \gamma$ **do**
6:         $\gamma \leftarrow (n.\gamma)$
7:         $p \in N$, s.t. $D[start][p] + d(p,n) = D[start][n]$
8:         **for all** $m \neq p$, s.t. $D[start][m] < \infty$ and $d(m,n) < \infty$ **do**
9:             $Q \leftarrow Q \cup \{(c + D[start][m] + d(m,n) - D[start][n], m, \gamma)\}$
10:        **end for**
11:        $n \leftarrow p$
12:    **end while**
13:    **if** $n = start$ and $s \notin \gamma$ **then**
14:        $C \leftarrow C \cup \{(start.\gamma)\}$
15:    **end if**
16: **end while**
```

The MPS algorithm complexity is $\mathcal{O}(K.|N|^2)$. As the MPS algorithm is used offline before the CRA computation, this complexity is acceptable. By reducing the number of possible paths, the MPS algorithm improves the CRA complexity: the branching factor B can be replaced by the maximal number of deviations, which is around K over the length (in points) of a path. However, as the complete state-space is not explored, the solution will be sub-optimal. The CRA complexity using MPS is given in (2).

$$\mathcal{O}(|N|^2.(L.\log(|N|) + K.C^2)) \tag{2}$$

IV.B.2. Results

Figures 10 and 11 draw the evolution of respectively flight delays and computation time according to K on Blagnac airport.

[b] D is the shortest distance matrix computed by the Floyd-Warshall algorithm while d is the direct distance between edges (equal to ∞ when the nodes are not connected)

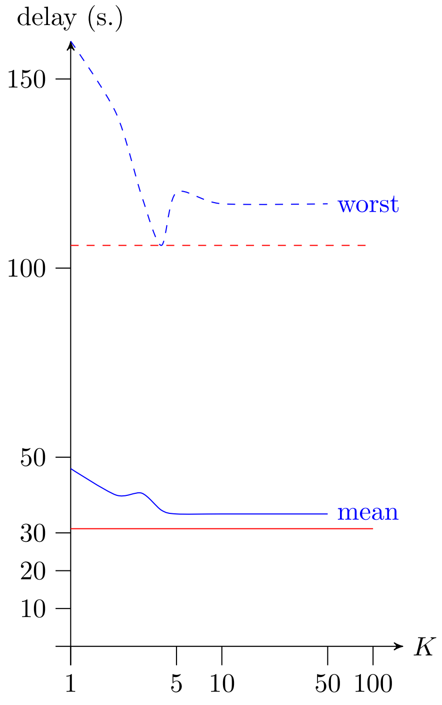

Figure 10. Average and worst delays on Blagnac airport, according to K. In red, results obtained without MPS.

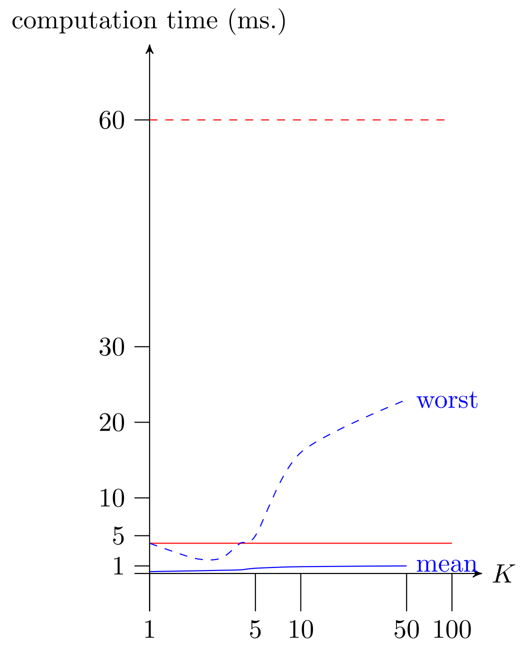

Figure 11. Average and worst computation time on Blagnac airport, according to K. In red, results obtained without MPS.

Figures 12 and 13 show the improvements made on the algorithm using the MPS algorithm. The mean delay is about 3% of the optimal delay while computation time is reduced by 30% and 10% on bost cases. These improvements correspond to a mean computation time lower than 10ms, and a worst computation time around half a second.

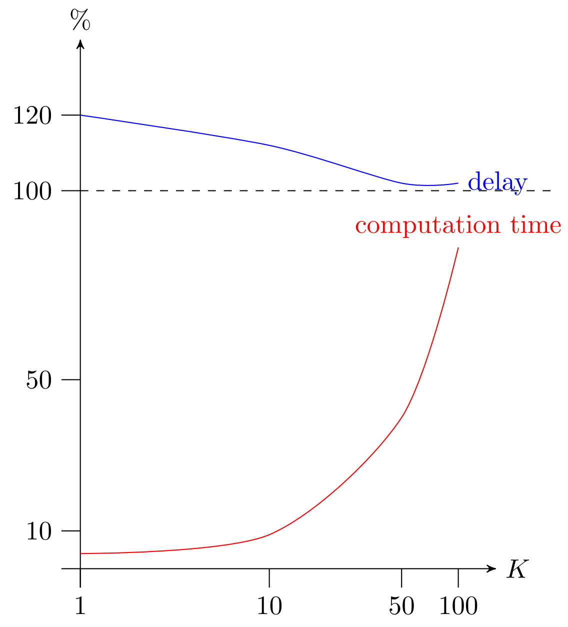

Figure 12. Improvement of CRA algorithm using MPS ($|I| = 10$).

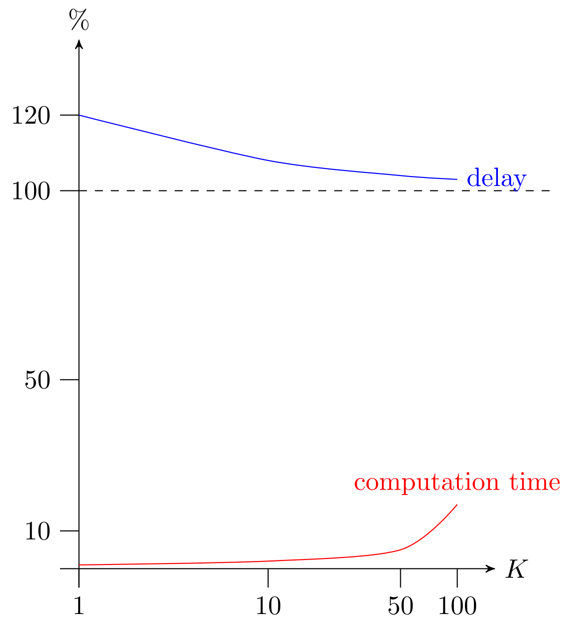

Figure 13. Improvement of CRA algorithm using MPS ($|I| = 20$).

The resulting planning algorithm, using Floyd-Warshall and MPS precomputations, can be concretely used on large airports to plan aircraft trajectories.

V. Conclusion

In this paper, an algorithm for ground traffic planning has been proposed. This algorithm is based on flight contract reservation and allows to deal with time and speed uncertainty while ensuring aircraft separation.

The algorithm (CRA) has been improved using both a more accurate heuristic – using Floyd-Warshall algorithm – and a pruning algorithm – MPS. The different versions of CRA have been evaluated and compared on a one-day traffic simulation on Toulouse-Blagnac and Frankfurt airports. The influence of time uncertainty, heuristic and pruning on flight delays and computation time has been studied.
The evaluations have lead to conclude that the use of both Floyd-Warshall and MPS algorithms off-line significantly improves the CRA efficiency. The resulting algorithm is now usable in real situations on large airports.

More complete simulations have to be performed to validate the first conclusion presented in this paper by analysing the influence of the K parameter according to time and speed uncertainty and other large airports such as JFK.

Acknowledgments

This work is part of the IESTA programme, funded by a set of European (ERDF - European Regional Development Fund) and national French public credits.
The activities described in this paper are included in a collective work carried out by the whole team of the IESTA programme. Moreover, the IESTA programme owes its existence to a federative and close collaboration between several Onera scientific departments that gather a multi-disciplinary team of scientific experts of the wide range following domains: Long-Term Design & Systems Integration, Systems Control and Flight Dynamics, Computational Dynamics & Aeroacoustics, Physics, Instrumentation & Sensing, Aerodynamics & Energetics Modelling.

References

[1]S. Swierstra and S. Green. Common trajectory prediction capability for Decision Support Tools. In *ATM R&D Seminar*, Budapest, Hungary, 2003.
[2]H. Oberheid and D. Soffker. Designing for cooperation mechanisms and procedures for air-ground integrated arrival management. In *IEEE Conf. on Systems, Man and Cybernetics*, 2007.
[3]P. Martin, O. Delain, and F. Fakhoury. Collaborative decision making: results of experiments to identify limitations of information exchanges in stand and gate operations. In *ATM R&D Seminar*, Santa Fe, NM, USA, 2001.
[4]J.-C. Vallée. Evolution of ground movements tracking on airports. Technical Report 61, DGAC/STNA, Toulouse, France, 2001.
[5]J.-B. Gotteland. *Ground Traffic Optimization on Large Airports.* PhD thesis, INPT, Toulouse, France, 2004.
[6]R. Deau, J.-B. Gotteland, and N. Durand. Runways sequences and ground traffic optimisation. In *ICRAT'08*, Fairfax, VA, USA, 2008.
[7]P. van Leeuwen and N. van Hanxleden. Scheduling aircraft using constraint relaxation. In *UK Planning and Scheduling Meeting*, Glasgow, UK, 2003.
[8]T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*, chapter pp. 558-565. MIT Press and McGraw-Hill, 2001.
[9]M. Pascoal, J. dos Santos, and E. de Queiros Vieira Martins. An algorithm for ranking loopless paths. In *INFORMS/CORS*, Montreal, Canada, 1998.