Genetic algorithms applied to airport ground traffic optimization

Jean-Baptiste Gotteland, Nicolas Durand

Genetic Algorithms Applied to Airport Ground Traffic Optimization

Jean-Baptiste Gotteland
Ecole Nationale de l'Aviation Civile
7, av Edouard-Belin - BP 4005
F31055 Toulouse Cedex 4
gotteland@recherche.enac.fr

Nicolas Durand
Centre d'Etudes de la Navigation Aerienne
7, av Edouard-Belin - BP 4005
F31055 Toulouse Cedex 4
durand@recherche.enac.fr

Abstract- Due to air traffic growth and especially hubs development, major European airports can easily become bottlenecks in the global air transportation network. Therefore, accurate models of airport traffic prediction become more and more necessary for ground controllers.

In this paper, a ground traffic simulation tool is proposed and applied to Roissy Charles De Gaulle airport. Two global optimization methods, using genetic algorithms at the airport, are developed to minimize taxiing time while respecting aircraft separation and runway capacities.

In order to compare the efficiency of the different methods, simulations are carried out on a one day traffic sample, and ground delay is correlated to the traffic density at the airport.

1 Introduction

Traffic delay due to airport congestion and ground operations becomes more and more penalizing in the total gate-to-gate flight cycle. This phenomenon can be in a large part attributed to recent hubs development, as all departures and arrivals are tending to be scheduled at the same time. Moreover, many ATC[1] and ATM[2] inefficiencies can appear as a result of taxi queueing and take-off time uncertainty. As a consequence, airport ground traffic simulation tools become essential for airport designers and traffic managers. Highly detailed models of airport operations already exist, such as SIMMOD (SIMulation MODel, developed by the Federal Aviation Agency) or TAAM (Total Airspace and Airport Modeler, developed by the Preston Group [Gro99]). They can be useful to evaluate qualitatively the relative effects of various airport improvements.

In the context of the A-SMGCS concepts (Advanced Surface Movement Guidance & Control System), all the ground traffic information is supposed to be directly available for the ATC and many tools could be developed to assist ground controllers.

In this paper, a ground traffic simulation tool with a conflict resolution module is introduced in part 2. Parts 3 and 4 describe the different optimization methods used to find the best trajectory and the most adapted holding points for taxiing aircraft. In the last part, these optimization methods are applied on a one day traffic sample at Roissy Charles De Gaulle airport considering the operational airport configurations and speed uncertainties, and the results of the simulations are compared.

[1] ATC: Air Traffic Control

[2] ATM: Air Traffic Management

2 Ground traffic simulation

The ground traffic simulator used for this study is associated with a detailed airport model in order to assign a set of realistic paths to each aircraft.

Traffic prediction is regularly computed according to a given speed uncertainty. Algorithms estimate the best maneuvers that can be executed by each aircraft to ensure taxiing separations and runway sequencing while optimizing a global criteria such as the total delay.

This part briefly sums up the main topics concerning this ground traffic simulator.

2.1 Problem modelling

Inputs

The simulator inputs are the airport topological description, the aircraft types with their corresponding weight categories and a one day traffic sample at the airport.

A flight is described by its recorded flight-plan, containing the aircraft type, the runway and the gate used, and the departure or arrival time.

The aircraft type information enables the evaluation of the needed takeoff or landing distance, in order to select the possible runway entry or exit points. Moreover, it is also associated with a wake turbulence category, which determines the time separation between two aircraft (one, two or three minutes) on the runway.

Paths assignment

The airport is modelled as a graph linking its gates, taxiways and runways. Each link is assigned a cost, which is

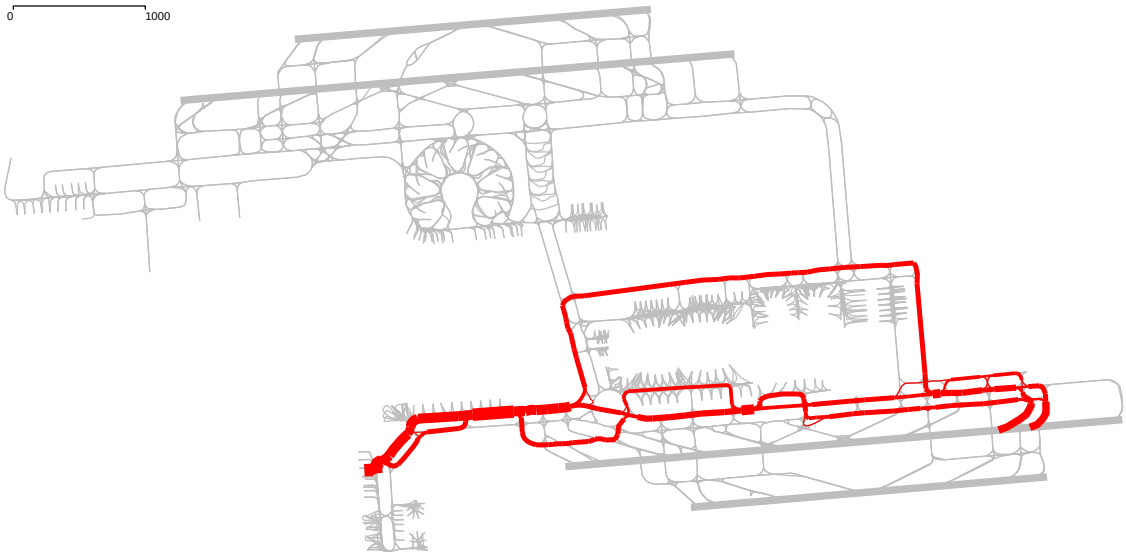

Figure 1: Airport graph

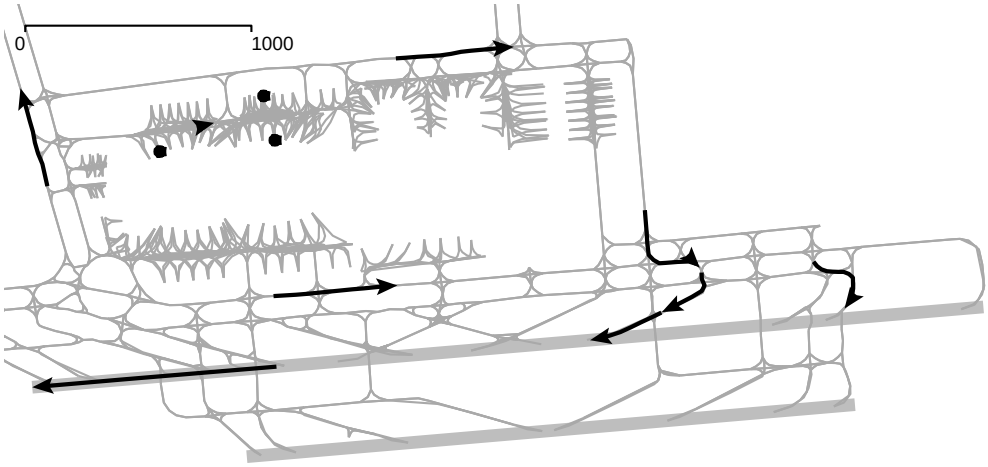

Figure 2: Aircraft multiple possible positions

an evaluation of the time spent by an aircraft when it proceeds via this link, added to some particular penalty related to runway areas, gate access or undesirable directions.

Classical graph algorithms can be used:

- The Dijkstra algorithm [AMO93] computes all the best paths and their corresponding minimal cost from a given node to every other ones.
- A k shortest loopless path enumeration algorithm [MPS97] can make use of the Dijkstra's result to find an acceptable set of different paths for each origin-destination (figure 1).

2.2 Conflict detection

Separation rules

A first model of aircraft separation rules consisted in the assignment of a maximal capacity to each portion of the airport graph. These capacities were defined as a linear function of the length of the taxiway portion. However, this model had to be refined as it could not ensure realistic aircraft separations near taxiways intersections nor manage dependent taxiways.

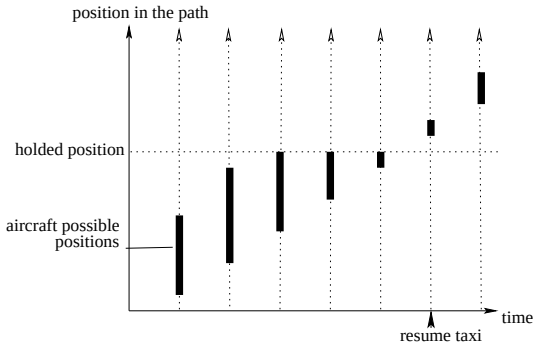

Figure 3: Uncertainty reduction on holding points

As a consequence, a discrete model of separation rules is defined as follows:

- The distance between two taxiing aircraft must never be less than 60 meters, except at the gate position.
- On the runway, a time separation of 1, 2 or 3 minutes (depending on the aircraft category) is necessary after a take off to clear next take off or landing from wake turbulence.
- When an aircraft is taking off or landing on a given runway, other aircraft can be taxiing in the same runway area only if they are behind it.
- When an aircraft is following an other one, its speed uncertainty is reduced (as the pilot won't go faster than the first one), so that the two aircraft are considered to be separated.

When the traffic prediction does not respect one of these rules, the two involved aircraft are called *a conflicting pair*.

Speed uncertainty

Each aircraft trajectory is predicted with a given speed uncertainty, fixed as a constant percentage of the nominal speed (which is function of procedures and turning rate).

Therefore, an aircraft is considered to occupy several possible positions at a given time (as shown figure 2).

Aircraft possible maneuvers

In order to ensure separations, aircraft trajectories can be modified by ground control orders. For each aircraft, a control order is defined by:

- The path that the aircraft must follow, chosen among the set of remaining possible paths for this aircraft;
- Optionally, a position where the aircraft must wait and a time ending this wait.

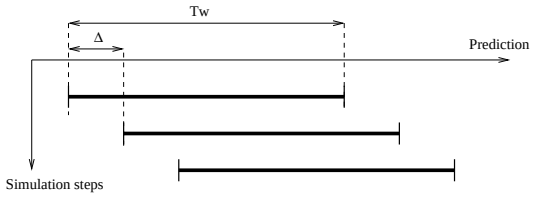

Figure 4: Shifted windows

Thus, the length of the segment representing the several possible positions of an aircraft is reduced when and where the aircraft is expected to wait, as the reference is a precise position and a precise end waiting time.

For an arriving aircraft, the waiting position can be in the air (this means that the aircraft is asked to slow down before landing). In this special case and to keep the simulation realistic, the delay cannot exceed λ seconds (usually $\lambda = 30$ seconds).

Clusters

In order to lower the complexity of the problem as often as possible, a transitive closure is applied on conflicting aircraft pairs and gives the different clusters of conflicting aircraft [DAN96]. The different clusters will be solved independently at first.

If the separated resolution of two clusters creates new conflicting positions between them, the two clusters are merged and the resultant cluster is solved.

2.3 Simulation steps

Shifted windows

The simulator works with a prediction time window T_w shifted every Δ minutes.

When the simulated current time is t, flight-plans expected to land or to leave the gate in the time range $[t; t+T_w]$ are activated: a set of appropriate paths is assigned to each of them and the resulting best trajectories are added to the ones of the already taxiing aircraft.

The conflict detection and resolution is performed in the time window and the resulting maneuvers are applied to build the new situation Δ minutes later. The problem is then reconsidered at the new simulation step $t + \Delta$ (see figure 4).

Global criteria to optimize

The global criteria to minimize is defined by the total taxiing time (including the time spent queueing for runway), increased by the time spent in lengthened trajectories.

With this definition, lengthening trajectory appears to be twice more penalizing than holding position.

3 The One-to-n method

This resolution method, called 1-to-n, treats a simplified problem. Aircraft are initially sorted and considered one after the other: first considered aircraft have priority on last considered ones.

The optimization problem is therefore reduced to one aircraft: the algorithm must find the best path and the best holding positions for the aircraft, taking into account the already considered aircraft trajectories.

3.1 The Branch & Bound algorithm

The 1-to-n method used in [BDA99] to find the best solution for an aircraft avoiding some other ones had to be refined in order to take into account the limited time window T_w and the speed uncertainties. However, this method is still modelled as a shortest path finding problem in a graph:

- A node of the graph represents a position of the aircraft, in a path p_i at time t.
- If the position conflicts with another aircraft, the node have no son. Otherwise, it has two sons, corresponding to the two possible choices for the aircraft at each time step: go forward or hold position.
- The root nodes are defined by the current position of the aircraft at current time t_0 on each remaining path p. The initial cost (to reach each root node) is the time length of the path.
- The terminal nodes are the ones describing a non conflicting position of the aircraft at the end of the time window ($t = t_0 + T_w$).

As the problem is to minimize delay, it is always better (when possible) to go forward than to hold position: thus, a Branch & Bound (instead of an A*) algorithm with a best first search strategy ([HT95]) can quickly find the best solution for the aircraft (or prove that there is no solution)

3.2 Initial classification

As last considered aircraft are extremely penalized (they must avoid all first considered aircraft) the way to sort aircraft, called the classification, is a determining factor. Moreover, the classification must respect some constraints:

- As landing aircraft cannot hold position before exiting runway, their priority level must be higher than all taking off aircraft on the same runway.
- Departures queueing for runway should be sorted according to their position in the queue.

Therefore, a time T_{a_i} is computed for each aircraft a_i as follows:

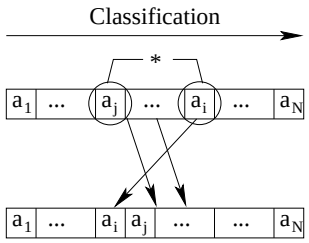

Figure 5: Classification reviews

- $T_{a_i} = t_0 + t_r$ for departures, where t_0 is the current simulation time and t_r is the minimal remaining time until the runway.
- $T_{a_i} = T_{land} - \delta_{land}$ for arrivals, where t_{land} is the expected landing time and δ_{land} the landing priority (1 hour for example).

Aircraft are sorted according to increasing values of T_a.

3.3 Classification reviews

Due to the limited time window for traffic predictions and speed uncertainties influence, the resolution of the current situation (at time t_0) can be very different than the previous one (at time $t_0 - \Delta$), so that some aircraft can finally be in a position with no solution. A common example is the one of an aircraft leaded in front of another one without having priority on it.

In order to manage these cases, the classification can be reviewed: when the algorithm cannot find any solution for an aircraft a_i, the penalizing aircraft $a_j (j < i)$ is identified, and the process is restarted after inserting a_i before a_j in the classification (see figure 5).

4 Genetic algorithms

4.1 Principles

Two methods using classical Genetic Algorithms [Gol89, Mic92] with partially separable functions and their adapted crossover and mutation operators as presented by N. Durand in [DA98] are developed and compared.

- The first method, called **GA_WAIT**, looks for an optimal association of paths and holding positions for aircraft. It was already developed in [BDA99], and was directly adapted to the new airport structure and traffic model.
- The second method, called **GA_SORT**, looks for an optimal association of paths and priority levels for aircraft. It is combined with the Branch and Bound algorithm described in part 3.

As the two methods implementation is very similar, they are simultaneously described in the next sections.

4.2 Data structure

For both methods, an element of the population (a chromosome) must be a set of parameters describing the N aircraft trajectories during the considered time window.

In order to keep the partial separability of the problem, each aircraft trajectory must be associated with its specific parameters, as detailed below.

Encoding for GA_WAIT

In this method, a chromosome is defined by $3N$ variables: $\{(n_i, t_{0i}, t_{1i})\}_{1 \leq i \leq N}$

n_i is the number of the path that the aircraft a_i must follow (chosen among the set of remaining possible paths for the aircraft) and $[t_{0i}; t_{1i}[$ the laps of time during witch it must hold position (if $t_{0i} \geq t_{1i}$, the aircraft a_i does not stop).

Encoding for GA_SORT

A chromosome is here defined by $2N$ variables: $\{(n_i, p_i)\}_{1 \leq i \leq N}$

n_i is the number of the path that the aircraft a_i must follow (as for GA_WAIT), and p_i is its priority level, in the range $[1, ..., N]$. The encoding must ensure that $(p_i)_{(1 \leq i \leq N)}$ is a deterministic classification:

$$\forall i \neq j,\ p_i \neq p_j \tag{1}$$

In this way, the N aircraft trajectories can be developed by running the Branch & Bound algorithm with the classification obtained with (p_i), restricted to one path per aircraft, given by (n_i).

4.3 Fitness function

Both methods are implemented with the same fitness function, in harmony with the global criteria to optimize (see part 2.3) constrained by the separation rules (see part 2.2).

The information needed for the evaluation of the N trajectories is stored in a fitness matrix M (see figure 6).

- $m_{ii} = d_i + 2.l_i$ represents the delay penalty of aircraft a_i. d_i is the time spent in holding position and l_i the time spent in lengthened trajectory due to the path n_i.
- m_{ij} is the number of conflicts between aircraft a_i and a_j.

To ensure that a chromosome describing a situation without any conflict is always better than a situation with a conflict, the fitness function F is defined as follows:

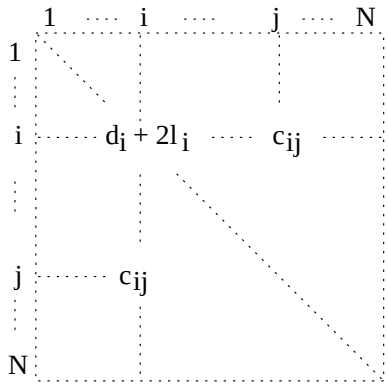

Figure 6: Fitness matrix

If the fitness matrix is diagonal (no conflict),

$$F = \frac{1}{2} + \frac{1}{2 + \sum_{i=1}^{N} m_{ii}} \quad (F > \tfrac{1}{2})$$

Otherwise:

$$F = \frac{1}{2 + \sum_{i<j} m_{ij}} \quad (F < \tfrac{1}{2})$$

Moreover, a local fitness f_i (to be minimised) can be assigned to aircraft a_i:

$$f_i = m_{ii} + K \sum_{j \neq i} m_{ij}$$

Where $K \gg m_{ii}$ (for example: $K = 10T_w$).

These local fitness are used to orient the crossover and mutation operators (see part 4.5).

4.4 Initial population

The initial population is generated with some heterogeneous random functions, as the best solutions are more expected to be in some particular subspaces:

- Values of n_i for which l_i (time lengthening) is low, for both encodings;
- Low waits for GA_WAIT: $t_{0i} \simeq t_{1i}$;
- Priority levels (p_i) for GA_SORT near the ones corresponding to the classification by expected runway access time (T_{a_i}), defined in part 3.2.

4.5 Crossover and mutation

The crossover operator used for both methods is the one described in [BDA99] (see figure 7), applying the principles described in [DA98] with the local fitness f_i.

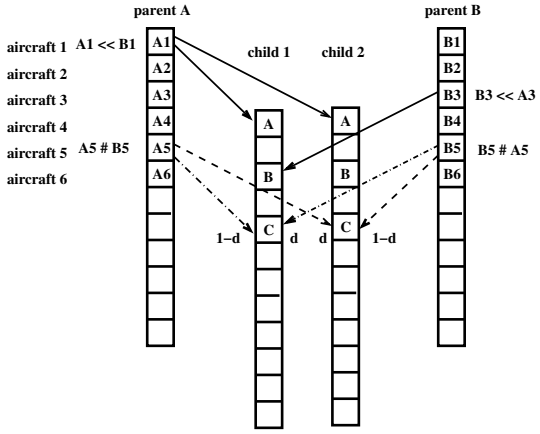

Figure 7: Crossover operator

The aim of this crossover operator is to increase the probability of generating children with a better fitness than their parents:

- when the local fitness f_i of an aircraft a_i is far greater in one of the two parents, the two children directly inherit of its associated variables (n_i, t_{0i}, t_{1i}) or (n_i, p_i) (see figure 4.5).
- when the local fitness f_i of the two parents are similar, the children inherit of a combination of the two versions of the associated variables.

The mutation operator randomly modifies the parameters of an aircraft having one of the worst local fitness.

In the GA_SORT encoding, the new elements of the population must be formated before insertion in the population, in order to restore the classification constraints (1).

4.6 Fitness matrix computation

GA_WAIT

To compute the fitness matrix (for a chromosome evaluation) in the GA_WAIT method, the aircraft trajectories are built according to the $3N$ variables and a conflict detection between each pair of aircraft is performed.

The conflict detection is the most penalizing operation: each of the N aircraft can have δT_w positions at each time step (δ is the speed uncertainty ratio). An elementary conflict detection (between two aircraft at a given time step) is implemented in $\mathcal{O}(\delta T_w)$ operations.

Therefore, the whole conflict detection is achieved in $\mathcal{O}(T_w * N^2 * (\delta T_w)) = \mathcal{O}(\delta N^2 {T_w}^2)$ operations.

As the time to solve the problem would be limited in an operational context, the evaluation process is accelerated by keeping the m_{ij} values of the fitness matrix when a pair (i, j) is unchanged after a crossover or mutation operation.

GA_SORT

In the GA_SORT method, the Branch & Bound algorithm described in part 3 (limited to a single path per aircraft) is used to develop sequentially all aircraft trajectories, with the classification given by (p_i).

If there is no solution for an aircraft a_i, the penalizing aircraft a_j ($p_j < p_i$) is identified by the Branch & Bound algorithm, and the m_{ij} (and m_{ji}) elements of the fitness matrix are increased: in this method, there is no need of any supplementary conflict detection.

For each aircraft, the graph exploration visits $\mathcal{O}({T_w}^2)$ nodes. Each node development needs $\mathcal{O}(N)$ elementary conflict detections ($\mathcal{O}(\delta T_w)$ operations for each of them). Thus, the fitness matrix computation is achieved in $\mathcal{O}(\delta N^2 {T_w}^3)$ operations.

However, this evaluation process cannot be accelerated, as the trajectory of an aircraft depends on all the trajectories of previously handled aircraft in the classification.

4.7 Sharing

In order to avoid a premature convergence of the algorithm around local optima, a clusterised sharing process is implemented as detailed in [YG93]. This sharing process applies the general principles introduced by Goldberg and Richardson [GR87] but can be computed in $\mathcal{O}(n \log(n))$ operations instead of $\mathcal{O}(n^2)$ for classical sharing (n is the population size).

The distance between two chromosomes A, B is defined as follows :

- For GA_WAIT encoding:

$$D(A,B) \quad = \quad \frac{\sum_{i=1}^{N} |d_{iA} - d_{iB}| + |l_{iA} - l_{iB}|}{2NT_w}$$

- For GA_SORT encoding:

$$D(A,B) \quad = \quad \frac{\sum_{i=1}^{N} |p_{iA} - p_{iB}|}{2N} + \frac{\sum_{i=1}^{N} |l_{iA} - l_{iB}|}{2T_w}$$

Where:

- $d_{iA} = t_{1iA} - t_{0iA}$ (resp. d_{iB}) is the time spent holding position by aircraft a_i in chromosome A (resp. B).
- l_{iA} (resp. l_{iB}) is the time spent in lengthened trajectory in chromosome A (resp. B).

In this way, the distance increases with the difference of penalization assigned to aircraft.

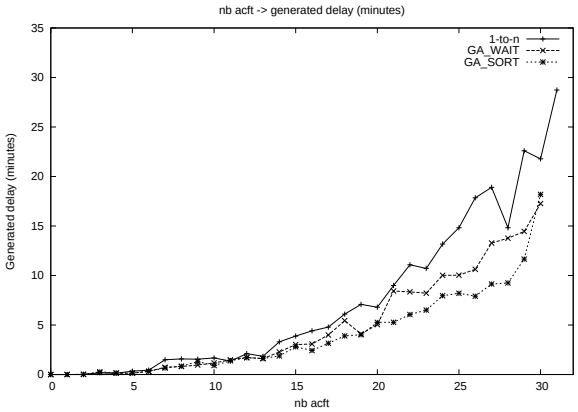

Figure 8: Generated delay

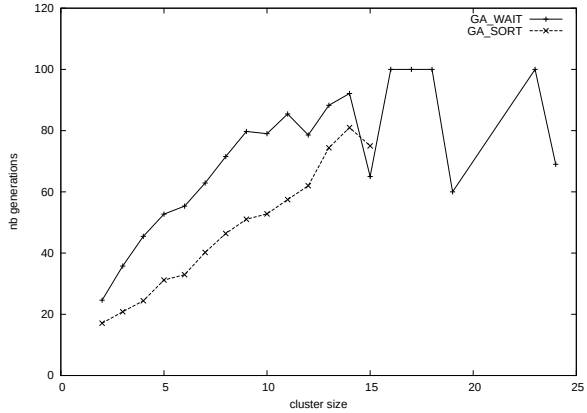

Figure 10: Mean number of generations

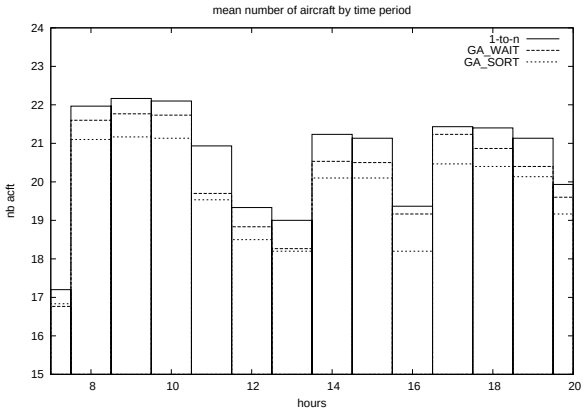

Figure 9: Mean number of aircraft by time period

4.8 Ending criteria

The ending criteria for GA_WAIT and GA_SORT is defined with two parameters N_{max}, N_{opt}:

- $N_{max} = 100$ is the maximal number of generations. If no available solution is found before, the GA will be aborted.
- $N_{opt} = 15$ is the maximal number of *solved* generations with the same best fitness. When the fitness of the best element of the population is greater then $\frac{1}{2}$ and is stationary during N_{opt} generations, the GA is stopped.

5 Simulation results

Simulations are carried out with real flight plans of Roissy Charles De Gaulle airport on a complete day: March 22^{nd} 2002, 1433 flights, 695 departures, 738 arrivals. The three simulations (one for each resolution method) are run with the following configuration:

- Time discretisation: $\sigma = 5$ seconds
- Time window: $T_w = 5$ minutes
- Simulation steps: $\Delta = 2$ minutes
- Max speed: $V_{max} = 10$ m/s
- Speed uncertainty: $\delta = \pm 10\%$

The two GA methods are run with a 200 chromosomes population size, a 60% crossover rate, a 15% mutation rate and the stochastic reminder without replacement selection principle, described in [Gol89].

Figure 8 gives the mean generated delay as a function of the number of aircraft involved in the situation. It appears that the 1-to-n method always generates more delay than the two others, and that GA_SORT is the most efficient method: solving the situation after sorting aircraft does not seem to be so penalizing, at the condition to get an adapted classification.

Figure 9 gives the mean number of aircraft simultaneously moving for each time period. It appears that the GA_SORT method keeps a lower number of moving aircraft during heavy time periods: good resolutions of ground traffic situations allow to decrease delay, but also lead to better situations with less moving aircraft.

In order to observe the GA efficiency, figure 10 gives the mean number of generations required by the two GA methods as a function of the clusters' size and figure 11 gives the mean value of the best fitnesses of each resolution. It can be noticed that the GA_SORT method never deals with clusters bigger than 15 aircraft. It generally requires less generations than the GA_WAIT method and find a solution with a bigger fitness to the problem. This can be explained by the one wait per aircraft limitation of the GA_WAIT method, which appears to be more penalizing than the classification restrictions needed by the GA_SORT method.

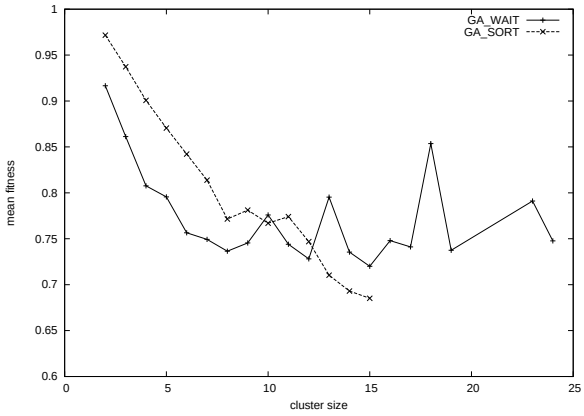

Figure 11: Mean value of best fitnesses

6 Conclusions

The simulations carried out in this study show that ground traffic delay at big airports is not only due to runway capacity but is very sensitive to the way that aircraft are managed: at Roissy Charles De Gaulle, one of the busiest European airport, the generated delay was largely reduced in the simulations using genetic algorithms.

Even if it is not yet possible to compare the solutions coming from the simulations to the reality (taxiing speed and followed paths can be very different and ground controllers actions are not even recorded), these results are promising : it seams profitable to develop some traffic management tools that could assist ground controllers and provide them some optimized solutions.

Moreover, it can be noticed that the whole airport model was easily upgraded with some new and more detailed data (push-back procedures, undesirable directions, new runways, aircraft wake turbulence categories, speed uncertainties, etc) without changing the algorithm itself. Thus, this kind of simulations can also be useful to evaluate benefits of new airport structures or new traffic procedures.

Further work will concentrate in refining the modelling and the global criteria to optimize, taking into account for example takeoff sequencing needs of approach sectors or priority levels for slotted departures.

Bibliography

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows, Theory, Algorithms and Applications.* Prentice Hall, 1993.

[BDA99] B. Pesic, N. Durand and J. M. Alliot. Aircraft Ground Traffic Optimization using a Genetic algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference.* San Francisco, 2001.

[DA98] N. Durand and J. M. Alliot. Genetic crossover operator for partially separable functions. *Genetic Programming Conference.* 1998.

[DAN96] N. Durand, J. M. Alliot, and J. Noailles. Automatic aircraft conflict resolution using genetic algorithms. *Proceedings of the Symposium on Applied Computing.* Philadelphia, 1996.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[GR87] D. E. Goldberg and J. Richardson. Genetic Algorithms with Sharing for Multimodal Function Optimization. *Proceedings of the International Conference on Genetic Algorithms*, pp41-49. 1987.

[Gro99] The Preston Group. *TAAM Reference Manual.* 1999.

[HT95] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches.* Springer-Verlag, 1996.

[Mic92] Z. Michalewicz. *Genetic algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1992.

[MPS97] E. Martins, M. Pascoal and J. L. Santos *A new algorithm for ranking loopless paths.* Departamento de Matematica, Universidade de Coimbra, Portugal, May 1997.

[Pea84] J. Pearl. *Heuristics.* Addison-Wesley, 1984. ISBN: 0-201-05594-5.

[YG93] X. Yin and N. Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. *Proceedings of the Artificial Neural Nets and Genetic Algorithm International Conference*, pp450-457. Springer-Verlag, 1993.