Solving the Aircraft Landing Problem with time discretization approach

March 26, 2013

Alain Faye[1]
Laboratoire CEDRIC, ENSIIE, Evry

Abstract

This paper studies the multiple runway aircraft landing problem. The aim is to schedule arriving aircraft to available runways at the airport. Landing times lie within predefined time windows and safety separation constraints between two successive landings must be satisfied. We propose a new approach for solving the problem. The method is based on an approximation of the separation time matrix and on time discretization. It provides lower bound or upper bound depending on the choice of the approximating matrix. These bounds are used in a cut algorithm to, exactly or heuristically, solve the problem. Computational tests, performed on publicly available problems involving up to 500 aircraft, show the efficiency of the approach.

Keywords: Aircraft landing problem, Mixed Integer Programming.

1 Introduction

This paper adresses the problem of sheduling aircraft landings at an airport. Given a set of planes, the problem is one of assigning a runway and a landing time for each plane. Each plane has to land within its predefined time window and safety separation distances have to hold between any pair of planes. This problem, referenced in the literature as the Aircraft Landing Problem (ALP), has been extensively studied. Beasley et al. [2] present a mixed integer zero-one formulation of the problem. Each plane has a target landing time within its time window. A cost is accounted when a plane lands after or before its target time. The objective is to minimize the total cost of deviation from the target times. The problem is solved optimally with a linear programming based tree search algorithm. An heuristic is also proposed. Using a similar model, Ernst et al. [11] propose a different heuristic. When the binary variables are fixed the remaining continuous linear problem is solved by a specialised simplex method which evaluates the landing times very quickly. The method is used in a space search heuristic as well as a branch and bound algorithm. Fahle et al. [12] present a comparison of several modelizations and heuristics. They study a MIP program and an integer linear program using time discretization introduced in [2]. They also compare two heuristic algorithms, Hill Climbing and Simulated Annealing methods. A SAT (Satisfiability Problem) formulation is proposed to decide if there exists a valid solution for the problem. In Hansen paper [13], four genetic algorithms are tested for the problem. The authors introduce time windows runway dependent. Computational results are given for small instances. Beasley et

[1] alain.faye@ensiie.fr

al. [4] use a population heuristic to solve a problem instance based on observations during a busy period at London Heathrow airport. Diallo et al. [8] solve the problem for the single runway case. Experiments based on real datasets of Léopold Sédar Senghor Airport of Dakar are presented. Pinol and Beasley [15] consider extensions of the previous model given in [2]. Time windows and separation distance between two successive aircraft are assumed to be runway dependent. These new criterias involve respectively linear and quadratic constraints. Next, two population heuristic approaches are applied to the problem. In order to consider airline preferences for individual flights, a new objective function is introduced by Soomer and Franx in [16]. Equity considerations lead to a convex piecewise linear function that has to be minimized. The problem is solved by a local search heuristic. Artiouchine et al. [1] consider the case where more than one time window is assigned to each plane. These time windows modelize the mechanisms as vector spaces and holding patterns, used by air controllers to delay planes. The separation criterias are not a constraint but the objective is to maximise the minimum elapsed time between any two consecutive landings. The authors study the complexity of the problem and identify polynomially solvable cases. For solving general problems, a branch and cut framework is used. Bianco et al. [7] propose a job-shop scheduling model with sequence dependent processing times. Jobs modelize planes. The paths of the planes in the terminal area are decomposed into machines like runways. The processing times take into account the separation distances between planes. The model is quite general and is suitable for arrivals and departures. Several objectives are considered as average delays minimization, maximum delays minimization and throughput (capacity) maximization. The problem is solved by a fast local search heuristic. Bencheikh et al. [6] modelize the landing problem as a job-shop scheduling problem. The problem is solved by a hybrid method combining genetic algorithms with Ant Colony optimization algorithm. D'Ariano et al. [10] consider take-off and landing operations at a busy terminal area. Scheduling and timing aircraft problem is modelized as a job shop scheduling problem and is solved by a truncated branch and bound algorithm for fixed routes. Aircraft rerouting is performed by a tabu search in order to improve the solution. The overall algorithm is tested on practical instances from Rome Fiumicino airport. Diaz and Mena [9], van Leeuwen et al. [14] have used Constraint Programming. These methods are well suited for small instances. But quality of the provided solutions is less good for large instances. The dynamic case of ALP is considered by Beasley et al. in [3]. Decisions must be taken in a dynamic fashion as time passes. Each new decision must take into account the previous decision that was made. The problem is solved using two heuristics.

In this research, we propose a new approach for solving the Aircraft Landing Problem. The method is based on an approximation of the separation time matrix and on time discretization . It provides lower bound or upper bound depending on the choice of the approximating matrix. These bounds can be used in a cut algorithm to optimally solve the problem. They can also be used in an heuristic method. In the next sections, we briefly recall the classical formulation of the problem. Next, we present the approximation method, time discretization, the cut algorithm and the heuristic method based on cuts. Computation results are reported and concluding remarks follow.

2 Classical modelisation

Each aircraft entering within the radar range at its destination airport, receives instructions from air traffic control. A landing time and a runway on which to land are assigned it. The landing time must be between an earliest landing time and a latest landing time. The earliest landing time corresponds to the time at which the aircraft can land if it flies at its fastest speed. To delay the landing time, the speed of the aircraft can be decreased or the flight plan can be

lenghtened by circling. The latest time corresponds to the maximum landing time achievable by these delaying mechanisms. Within this time window, there is a target time which corresponds to the time at which aircraft can land if it flies at its cruise speed. The target time is the preferred landing time. Landing after or before this time causes disturbance at the airport. Consequently, a cost can be incurred with each deviation (before or after) of the target time. Safety distances between pair of successive planes must be respected. Separation distances are converted into separation times using a fixed landing speed depending on the aircraft type. Then, a minimal time must be elapsed between the landing of a plane and the landing of any successive plane. Separation time holds between pair of planes landing on the same runway or on different runways.
We give a modelisation of the Aircraft Landing Problem which is based on the one presented in [2, 15].

The data are the following:

- P set of planes, R set of runways available for landing
- E_i the earliest landing time for plane i $\forall i \in P$
- T_i the target (preferred) landing time for plane i $\forall i \in P$
- L_i the latest landing time for plane i $\forall i \in P$
- $S_{ij} \geq 0$ the minimum separation time between planes i and j where i lands before j on the same runway
- $s_{ij} \geq 0$ the minimum separation time between planes i and j where i lands before j on a different runway

S_{ij} is called longitudinal separation time between leading aircratf i and trailing aircraft j. s_{ij} is the diagonal separation time (cf.[7]). We assume $S_{ij} > s_{ij}$.

The decision variables are the following:

- δ^1_{ij} $\forall i, j \in P (i \neq j)$ binary variable where $\delta^1_{ij} = 1$ if and only if aircraft i lands before aircraft j and on the same runway.
- δ^2_{ij} $\forall i, j \in P (i \neq j)$ binary variable where $\delta^2_{ij} = 1$ if and only if aircraft i lands before aircraft j and on a different runway.
- z_{ij} $\forall i, j \in P$ $(i < j)$ binary variable where $z_{ij} = 1$ if and only if aircraft i and j land on the same runway.
- y_{ir} $\forall i \in P, r \in R$ binary variable where $y_{ir} = 1$ if and only if aircraft i lands on runway r.
- $x_i \geq 0$ the scheduled landing time for plane i $\forall i \in P$

The constraints are the following:

$$(P_B)\begin{cases} x_j \geq x_i + S_{ij} + (1-\delta^1_{ij})(-S_{ij} - L_i + E_j), \forall i \neq j \in P \ (1) \\ x_j \geq x_i + s_{ij} + (1-\delta^2_{ij})(-s_{ij} - L_i + E_j), \forall i \neq j \in P \ (2) \\ \delta^1_{ij} + \delta^1_{ji} = z_{ij}, \forall i < j \in P \ (3) \\ \delta^2_{ij} + \delta^2_{ji} = 1 - z_{ij}, \forall i < j \in P \ (4) \\ z_{ij} \geq y_{ir} + y_{jr} - 1, \forall i < j \in P \ \forall r \in R \ (5) \\ \sum_{r \in R} y_{ir} = 1, \forall i \in P \ (6) \\ E_i \leq x_i \leq L_i, \forall i \in P \ (7) \\ \delta^1_{ij}, \delta^2_{ij} \in \{0,1\}, \forall i \neq j \in P \\ z_{ij} \in \{0,1\}, \forall i < j \in P, y_{ir} \in \{0,1\}, \forall i \in P \ \forall r \in R \end{cases}$$

We give a slightly different formulation of the one given in [2, 15]. Constraints (1) are related to planes landing on the same runway, while Constraints (2) are related to planes landing on different runways. They ensure minimum separation time in each case if $\delta^1_{ij} = 1$ or $\delta^2_{ij} = 1$. If $\delta^1_{ij} = 0$ then (1) becomes $x_j \geq x_i - L_i + E_j$ which is true by (7). The same holds for (2). Hence for $\delta^1_{ij} = 0$ and $\delta^2_{ij} = 0$, these constraints are inactive. In [2, 15], conditions (1), (2) are aggregated in a single set of constraints.
If planes i and j $(i < j)$ land on the same runway i.e. $z_{ij} = 1$, Constraint (3) becomes $\delta^1_{ij} + \delta^1_{ji} = 1$ meaning that either aircraft i or j must land first. If planes i and j land on different runways i.e. $z_{ij} = 0$ then $\delta^1_{ij} + \delta^1_{ji} = 0$ and Constraint (1) is inactive.
Constraints (4) and (2), relative to planes landing on different runways, are linked in the same manner.
If $y_{ir} = y_{jr} = 1$ i.e. planes i and j land on the same runway r then Constraint (5) ensures that $z_{ij} = 1$. Otherwise z_{ij} will be either equal to 0 or 1. The value $z_{ij} = 0$ will active Constraint (2) which is less constraining than (1) since $S_{ij} > s_{ij}$, and zero value will be preferred in an optimization problem.
Constraint (6) ensures that each plane lands on exactly one runway.
Constraint (7) ensures that each plane lands within its time window . The remaining constraints are integrality conditions.
Note that for a given runway assignment, the binary variables y and z are fixed. Furthermore, whenever y and z are fixed, for a given landing sequence i.e an ordering of the planes $(i_1, i_2, ..., i_{|P|})$ such that i_1 lands before i_2 which lands before i_3 ... , the binary variables δ^1, δ^2 are fixed. Hence, for a given landing sequence and runway assignment, the continuous landing time variables x are the only remaining free variables.
Various objective functions can be defined for the problem. Two examples will be described in Section 3.2 .

3 Modelisation in the ideal case

The modelisation is based on the decomposition of the separation time matrix S, as well as s, into a rank two matrix, and on time discretization. First, we assume that the separation matrix is a rank 2 matrix and describe the modelization in this case. In the next section, we

will consider the general case. A time discretization approach was proposed in [2, 12] but here, a different way is followed.

3.1 The single runway case

The entries of the separation matrix S give the minimal required times between any pair of planes. S_{ij} is the required separarion time between leading aircraft i and trailing aircraft j. In fact, planes are partitioned into classes (for example, *heavy*, *medium*, *light*) and separation times are given between pair of classes (see for instance [7]). For simplicity of the presentation, we assume that the entries of S are indexed by planes.

Assume that S can be decomposed into the sum of a matrix with identical column-vectors a (with nonnegative entries) and a matrix with identical row-vectors b (with nonnegative entries) i.e. $S_{ij} = a_i + b_j$ with $a_i \geq 0$ and $b_j \geq 0$. Planes can be viewed as activities (non-preemptive tasks) sharing a single resource that is the runway. An activity i is divided into 2 parts: the first one of duration b_i and the second one of duration a_i. The landing time is just at the beginning of the second part. The total duration is equal to $a_i + b_i$. Now, assume that aircraft i lands before j (or activity i is scheduled before j). Then the separation time between i and j is at least $a_i + b_j$. Figure 1 illustrates this point where x_i and x_j, landing times of planes i and j respectively, are represented on the time axis t.

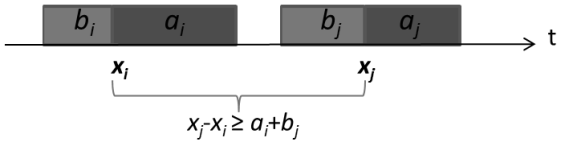

Figure 1: Safety time beetween leading aircraft i and trailing aircraft j

The planning horizon is discretized into time slots. For example, one hour is discretized into 3600 slots of one second and it is assumed that no event occurs between two consecutive time slots. Figure 2 gives an example of plane i with $a_i = 2$ and $b_i = 3$ landing at $t = 4$. The plane (activity) picks up the runway (resource) on time slots $t = 1$ to $t = 5$. On the next time slot $t = 6$, the runway (resource) is free and an other aircraft (activity) can follow.

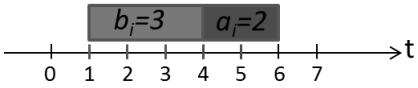

Figure 2: Aircraft i with landing time equals to 4

Note that a_i must be at least equal to 1 in order that plane i can pick up the runway (resource) at its landing time slot. With this discretization process, we can modelize the problem via a finite number of scenarios. Scenarios for a plane correspond to the different landing times in its time window. Scenarios are data of the problem and the decision variables are concerned by the choice of a scenario. The number of scenarios for a plane i is equal to the number of time slots in its time window.
Let us define $slot_{ikj} = 1$ if the runway is occupied in the time slot j by plane i in the scenario k and $slot_{ikj} = 0$ otherwise. In a scenario k of a plane i, the number of consecutive slots j

such that $slot_{ikj} = 1$ is equal to $a_i + b_i$, and $slot_{ikj} = 1\ \forall j \in [t_{ik} - b_i, t_{ik} + a_i\ [$ where t_{ik} is the landing time of plane i in scenario k. Let H be the set of time slots of the planning horizon and $Scen(i)$ be the set of scenarios for plane i.
With these datas, the single runway aircraft landing problem is modelized by the following set of constraints:

$$\begin{cases} \sum_{k \in Scen(i)} \lambda_{ik} = 1, \forall i \in P \ (8) \\ \\ \sum_{i \in P} \sum_{k \in Scen(i)} slot_{ikj} \lambda_{ik} \leq 1, \forall j \in H \ (9) \\ \\ \lambda_{ik} \in \{0, 1\}\ \forall i \in P, \forall k \in Scen(i) \end{cases}$$

Constraints (8) ensure that one and only one scenario is chosen for each aircraft. Constraints (9) ensure that the runway is occupied by at most one plane on each time slot of the planning horizon. The remaining constraints are integrality conditions of the decision variables λ_{ik}.

3.2 Objective

Time discretization and scenario formulation allow modelisation of quite general objective function. Let us define c_{ik} as the cost of the k^{th} scenario of plane i and let us show, on two examples, how c_{ik} can be computed.
Consider the linear deviation cost function used in [2] . Let T_i be the target (preferred) landing time for plane i. g_i (resp. h_i) is the penalty cost per unit of time for landing before (resp. after) the target time T_i for plane i. Let t_{ik} be the landing time of plane i in scenario k.
Then :

$$c_{ik} = \begin{cases} (T_i - t_{ik}) g_i \text{ if } T_i \geq t_{ik} \\ (t_{ik} - T_i) h_i \text{ otherwise} \end{cases} \quad (10)$$

The objective is to landing planes as close to their preferred landing time as possible. Hence, the problem is to minimize the overall cost: $\min_{\lambda} \sum_{i \in P} \sum_{k \in Scen(i)} c_{ik} \lambda_{ik}$

Consider the non-linear deviation cost function used in [15]. As above, let T_i be the target (preferred) landing time for plane i, and let t_{ik} be the landing time of plane i in scenario k.
Then :

$$c_{ik} = \begin{cases} (T_i - t_{ik})^2 \text{ if } T_i \geq t_{ik} \\ -(t_{ik} - T_i)^2 \text{ otherwise} \end{cases} \quad (11)$$

A plane landing after its target time is penalised. On the contrary, a plane landing before its target time is preferred. The objective is to landing planes as earlier as possible. Hence, the problem is to maximize the overall plane contribution: $\max_{\lambda} \sum_{i \in P} \sum_{k \in Scen(i)} c_{ik} \lambda_{ik}$

Other objective functions of the litterature can be modelized. For instance, coefficients c_{ik} can easily handle the scaled objective function used in [16].

3.3 Multiple runway case

We now consider the multiple runway case. Our approach is unchanged. The multiple runway case only increases the number of scenarios for each plane. The number of scenarios for plane i is now equal to the number of time slots in its time window multiplied by the number of runways. We define $slot_{ikrj} = 1$ if runway r on time slot j is occupied by plane i in its k^{th} scenario, and $slot_{ikrj} = 0$ otherwise.

In the multiple runway case, separation time between planes landing on different runways must be respected. s_{ij} is the required separation time between leading plane i and trailing plane j when these planes land on different runways. As for matrix S in Section 3.1, rank two decomposition of matrix s is assumed i.e. $s_{ij} = \alpha_i + \beta_j$ with $\alpha_i \geq 0$ and $\beta_j \geq 0$.
A fictive runway f is added to take into account this diagonal separation time constraint. We define $slot_{ikfj} = 1$ if plane i in its k^{th} scenario picks up the fictive runway f on time slot j, and $slot_{ikfj} = 0$ otherwise.
If plane i in scenario k lands on runway r, the number of consecutive slots j such that $slot_{ikrj} = 1$ is equal to $a_i + b_i$ and the number of consecutive slots j such that $slot_{ikfj} = 1$ is equal to $\alpha_i + \beta_i$. More precisely, $slot_{ikrj} = 1\ \forall j \in [t_{ik} - b_i, t_{ik} + a_i[$ and $slot_{ikfj} = 1\ \forall j \in [t_{ik} - \beta_i, t_{ik} + \alpha_i[$ where t_{ik} is the landing time of plane i in scenario k.
Figure 3 illustrates how longitudinal and diagonal separation time constraints are taken into account.

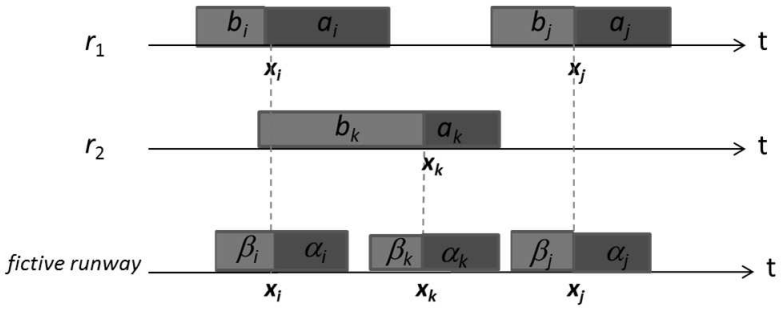

Figure 3: Fictive runway for taking into account diagonal separation times

Planes i, j land on runway r_1, plane k on runway r_2. Let x_i, x_j, x_k be the landing times of planes i, j, k respectively. The longitudinal separation contraint on r_1 is satisfied since $x_j - x_i \geq a_i + b_j = S_{ij}$. The diagonal separation constraints are modelized on the fictive runway. They are: $x_k - x_i \geq \alpha_i + \beta_k = s_{ik}$, $x_j - x_i \geq \alpha_i + \beta_j = s_{ij}$, $x_j - x_k \geq \alpha_k + \beta_j = s_{kj}$. The separation time constraint between i and j is accounted twice, on runway r_1 and on the fictive runway, but this last one is dominated by the first one since $S_{ij} > s_{ij}$ and, in fact, is inactive.
Dependent runway datas can be easily taken into account. In [15], runway dependent separation times are considered. S^r_{ij} is the required separation time between i and j landing on runway r. Assuming rank two decomposition of matrices S^r, this situation can be easily modelized in the scenario approach. Note that modelizing this case in the classical model, involves quadratic constraints (cf. [15]). Runway dependent cost function, runway dependent earliest, target, latest times and runway restrictions (cf.[13, 15]) can also be easily modelized.
Finally, under the rank two assumption of separation time matrices S and s, the general model for the aircraft landing problem can be stated as the following 0-1 linear problem:

$$(P_{sc}(S,s)) \quad \left| \begin{array}{l} \min / \max_{\lambda} \sum_{i\in P} \sum_{k\in Scen(i)} c_{ik}\lambda_{ik} \\ \text{s.t.} \begin{cases} \sum_{k\in Scen(i)} \lambda_{ik} = 1, \forall i \in P \ (12) \\ \sum_{i\in P} \sum_{k\in Scen(i)} slot_{ikrj}\lambda_{ik} \leq 1, \forall r \in R \cup \{f\}, \forall j \in H \ (13) \\ \lambda_{ik} \in \{0,1\} \ \forall i \in P, \forall k \in Scen(i) \end{cases} \end{array} \right.$$

Constraints (12) ensure that one and only one scenario is chosen for each aircraft. Constraints (13) ensure that each runway, fictive or not, is occupied by at most one plane on each time slot of the planning horizon. For $r \in R$, these constraints modelize longitudinal separation time. For the fictive runway $r = f$, they modelize diagonal separation time requirements. Note that if no diagonal constraint is considered (i.e. $s_{ij} = 0, \forall i, j \in P$), these constraints are vacant. The remaining constraints are integrality conditions of the decision variables λ_{ik}.

4 Solving the problem in general case

In previous Section 3, we assumed rank 2 decomposition of separation time matrices. We now consider the general case. We follow the previous scheme with approximated matrices. We approximate the longitudinal separation time matrix S and the diagonal separation time matrix s by rank 2 matrices. The way we approximate leads to exact or heuristic algorithms. If we use lower bounding approximations, the separation times are too short and we have to correct them with the exact durations. For doing this, we use constraints of classical model P_B presented in Section 2. This leads to a cut algorithm. Other approximations provide heuristic solutions. In this case, it is natural to use the best approximation. In the next sections, we explore these schemes.

4.1 Separation time matrix approximation

In this section, we explain how separation time matrices are approximated. We consider the longitudinal separation time matrix S. But the results extend to diagonal separation matrix. The planes are partitioned into categories and the separation times are not given for each pair of planes but for each pair of plane categories (see [7]). The matrix S is indexed by plane categories and S_{ij} is the minimum separation time between a plane of category i and a plane of category j when the first one lands before the second one. For example 3 categories, *heavy*, *medium*, *light*, lead to 3×3 entries matrix. This is an important point for the approximation process. Let C be the set of aircraft classes.
For any separation time matrix, an approximation by a rank two matrix is always possible. There are a lot of ways to approximate the matrix S. We explore the linear fashion. The problem can be stated as follows:

$$(\mathcal{P}) \quad \min_{a,b,e^+,e^-} \sum_{i,j\in C} e^+_{ij} + e^-_{ij} \ \text{s.t.} \begin{cases} a_i + b_j + e^+_{ij} = S_{ij} + e^-_{ij}, \forall i, j \in C \ (14) \\ a_i \geq 1, b_i \geq 0, \forall i \in C \ (15) \\ e^+_{ij} \geq 0, e^-_{ij} \geq 0, \forall i, j \in C \end{cases}$$

S_{ij} is approximated by $a_i + b_j$ in Constraint (14). $(\mathcal{P})$ minimizes $\sum_{i,j\in C} |a_i + b_j - S_{ij}|$ since in any optimal solution $e^+_{ij} > 0 \Rightarrow e^-_{ij} = 0$ and conversely. Hence, this linear program gives the best rank 2 approximation of matrix S.

Constraints (15) ensure that block lenghts modelizing planes of category i ($\forall i \in C$) are non negative. Moreover, the block lenght a_i cannot be equal to zero (see Section 3.1) .
For our time discretization process, we need to have a_i and b_i integer. The constraints of linear problem ($\mathcal{P}$) are totally unimodular and integrality conditions are not necessary. S is supposed to have integer entries.
If we set $e_{ij}^{+} = 0$ then (14) gives $a_i + b_j \geq S_{ij}$. In this case, the program provides an upper bounding rank 2 approximation. If we set $e_{ij}^{-} = 0$ then (14) gives $a_i + b_j \leq S_{ij}$. Then we obtain a lower bounding rank 2 approximation. Note that in this case, ($\mathcal{P}$) may be infaisible by reason of the inequalities $a_i \geq 1 \ \forall i \in C$.

We denote by :

- S_{LB} rank 2 matrix lower bounding S
- S_{best} best rank 2 approximation of matrix S
- S_{UB} rank 2 matrix upper bounding S

Note that the set of feasible solutions of ($\mathcal{P}$) does not cover the entire set of rank 2 matrices, even with $a_i \geq 1 \ \forall i \in C$ relaxed to $a_i \geq 0 \ \forall i \in C$ and the set of rank 2 matrices restricted to rank 2 matrices with non negative entries.

4.2 Exact solution

Lower bounding separation matrices provide relaxation of the separation time constraints, and then lower bound of the aircraft landing problem in minimization case (upper bound in maximization case). The idea is to use it in a cut algorithm. We use separation time constraints (1), (2) of model P_B and add them as cuts in P_{sc}. We have to link variables of model P_B with variables of problem P_{sc}. This is done by defining new datas.
Landing times depend on scenarios and appear in P_{sc} as data. Let t_{ik} be the landing time of plane i in scenario k. The variable x_i is now easily achieved by $x_i = sum_{k \in Scen(i)} t_{ik} \lambda_{ik}$.
The runway where a plane is landing depends on scenario. Let us define the data $runw_{ikr} = 1$ if plane i, in scenario k, lands on runway r, $runw_{ikr} = 0$ otherwise. The variables y_{ir} can be easily modelized with these new datas. We have $y_{ir} = sum_{k \in Scen(i)} runw_{ikr} \lambda_{ik}$.
The remaining variables δ_{ij}^1, δ_{ij}^2, z_{ij} are linked to y_{ir} by Constraints (3), (4), (5). Then, Constraints (1), (2) and related Constraints (3), (4), (5) can be expressed with variables λ_{ik}, and δ_{ij}^1, δ_{ij}^2, z_{ij}. They are introduced in P_{sc}.
The number of constraints may be important in case of a big number of planes. Moreover, these constraints involve binary variables. So, we use a cut algorithm and introduce only constraints that are necessary for finding the optimal solution. The algorithm can be stated as follows:

1. let (PB)=$P_{sc}(S_{LB}, s_{LB})$ be the initial problem built with lower bounding matrices.
2. solve (PB). Let λ be the solution.
3. compute $x_i = sum_{k \in Scen(i)} t_{ik} \lambda_{ik} \ \ \forall i \in P$
4. find a violated longitudinal separation time constraint (1), i.e. find a pair of planes i, j landing on the same runway with $x_i \leq x_j$ and $x_i + S_{ij} > x_j$.
5. find a violated diagonal separation time constraint (2), i.e. find a pair of planes i, j landing on different runways with $x_i \leq x_j$ and $x_i + s_{ij} > x_j$.

6. if violated inequalities have been found, express these constraints in variables λ_{ik} , δ^1_{ij}, δ^2_{ij}, z_{ij} , add them to (PB) and reiterate (go to 2); otherwise stop, the problem is solved.

The algorithm terminates with a solution satisfying all separation time constraints. Hence, it is an optimal solution.

4.3 Heuristic solution in large planning horizon

In previous Section 4.2, we have used rank 2 matrices lower bounding separation time matrices S and s, to exactly solve the problem. Now, we use the best rank 2 approximation of matrices S and s in order to obtain heuristic solutions.
First, we solve $P_{sc}(S_{best}, s_{best})$. We obtain a landing sequence and runway assignment. Next, we compute optimal landing times relative to the landing sequence and runway assignment previously found. In this scheme, the first phase can be improved by adding cuts (1), (2) of model P_B as in the algorithm presented in Section 4.2. Here, we can limit the number of iterations since we look at an heuristic solution. Note that even without limiting iteration number, the algorithm provides an heuristic solution because approximating matrices S_{best} and s_{best} are not lower bounds of separation time matrices S and s. Some entries of the approximating matrices may be greater than the original ones. Best approximating matrices being as close to the original matrices as possible, this effect is limited.
Time discretization of a large planning horizon may induce a huge number of time slots which can cause serious troubles in using the problem P_{sc}. One way to bypass this drawback is the following. We choose a larger unit width time period i.e. we discretize planning horizon in less time slots. For example, if the initial width time period is one , we choose $d > 1$ width time period and the total number of time slots is divided by d. We compute all the data according to the new width time period. L, E, T are divided by d, unitary costs are multiplied by d. Fractionnal values encountered in the division of L, E, T are converted into the first lower integer. Entries of separation time matrices S and s are divided by d. Fractionnal values encountered in the division of S and s are converted into the first upper integer. S^c and s^c denote respectively the new matrices S and s obtained by this contraction process. Best rank 2 approximation of matrices S^c and s^c are computed.

Then, we solve P_{sc}. We obtain a landing sequence and an assignment of planes to runways. Next, we go back to the initial datas, L, E, T, unitary costs, and matrices S, s, and we compute the landing times corresponding to this landing sequence and runway assignment.
Hence, the computation is decomposed into two phases. First, we compute the landing sequence of aircraft and runway assignment in $\frac{1}{d}$-contracted planning horizon. Next, we compute the landing times in the real planning horizon. The algorithm can be stated as in Algorithm1 for the first phase and Algorithm2 for the second one.

Algorithm 1 phase 1: solve the problem in $\frac{1}{d}$-contracted planning horizon

Require: S^c_{best} (resp. s^c_{best}) best rank 2 approximation of $\frac{1}{d}$-contracted matrix S^c (resp. s^c),
$(PB) = P_{sc}(S^c_{best}, s^c_{best})$ initial problem built with S^c_{best} and s^c_{best},
$L = \emptyset$;

1: **repeat**
2: Solve (PB)
3: Add the landing sequence and runway assignment found in list L
4: Find violated separation time inequalities and add them to (PB)
5: **until** no violated inequality is found or maximum iteration number is reached

Algorithm 2 phase 2: compute landing times in real planning horizon

Require: L=list of solutions (landing sequences and runway assignments) found in the previous phase

1: **for** landing sequence and associate runway assignment $\in L$ **do**
2: compute optimal landing times for the chosen objective
3: **end for**
4: output the best solution found

In phase 2, landing times computation depends on the chosen objective. We discuss two cases : linear one and non-linear one.
For the linear objective function (10), it is possible to use P_B. As mentioned in Section 2, for a given landing sequence and runway assignment, P_B contains only continuous landing time variables x. Then, computing optimal landing times is a linear problem which can be solved very quickly (see [2, 15]).
For the non-linear objective (11), with the $close - up$ property i.e. each plane is as close to its earliest time as can be achieved by separation constraints, it is possible to compute optimal landing times using a polynomially bounded algorithm. See [15].
One may notice that our heuristic can lead to infaisible solution. Indeed, the runway assignment and landing sequence found in the first phase are computed with approximating separation time matrices. In the second phase, the exact separation time matrices are used whose entries may be greater than the approximating separation times involved in the initial phase . To avoid this difficulty, one can use upper bounding separation time matrices S_{UB}, s_{UB} in the first phase. Then, if the first phase problem has a faisible solution, we obtain a faisible solution with the exact separation time matrices in the second phase. Nevertheless, there is no garantee obtaining a faisible solution in the first phase. Note that the problem of finding a faisible solution is NP-complete as mentioned in [16].

5 Computational results

Computational tests run on a PC-Windows computer, processor Intel(R) Core(TM) i5-2520M (2.5 GHz) with 4Go of RAM. Algorithms and mathematical models were implemented using FICO-Xpress Mosel language and problems were solved with FICO-Xpress Optimizer (version 7.0) software.
Computational tests have been performed on 13 instances publicly available from OR-library [5] involving from 10 to 500 aircraft. For each instance, the number of runways varies in the same range used in [15]. We consider two objectives, linear and non linear ones, described in Section 3.2. The instances are partitioned into two sets: small instances involving from 10 to 50 aircraft and large instances involving from 100 to 500 aircraft. For small instances the cut algorithm described in 4.2 is used to exactly solve the problem. For large instances, the problem is heuristicaly solved. As the planning horizon is large, the heuristic method described in 4.3 is used.
In all instances, time datas are integer and diagonal separation times are equal to zero. The given longitudinal separation time matrix S is indexed by planes. After analyzing matrix S, we have deduced the number of aircraft categories. The number of categories depends on the instances and is equal to 2 or 4 except for the instance with 50 aircraft which involves 34 categories. The separation time matrix between aircraft categories is then approximated by a rank two matrix (see Section 4.1). The required computation time is negligible. For 50 aircraft instance where a huge number of categories is encountered, computing S_{LB}, rank two matrix

lower bounding S, leads to an infaisible problem ($\mathcal{P}$). So, we cannot use our cut algorithm to exactly solve this instance. Despite of this, we use the best rank two approximation of matrix S in the cut algorithm described in Section 4.3 with $d = 1$ (i.e. without time contraction). This provides an heuristic solution. For large instances, the best rank two approximation of matrix S is used in the same heuristic method but with $d > 1$. The value of d depends on the number of planes of the instance.

The results for small instances with linear objective are given in Table 1. The description of the instances is reported with P=number of planes, R=number of runways, and optimal value of each instance. This table gives the number of iterations and CPU time required by our cut algorithm, and LP value (continuous relaxation) of the initial problem P_{sc}. For comparison, the results given in [2] are reported. In [2], the problem is solved using MIP formulation. This MIP is tighten each time an improved faisible solution is found. LP value reported is the best LP value found in this improvement process. CPU time is the total time taken to solve the problem. Beasley et al. use an heuristic algorithm and when this algorithm gives a solution value of zero, the optimal solution is found (i.e. all planes land on target). In this case, results concerning MIP are not reported.
We observe that LP value of our method is greater than LP value of improved Beasley et al.'s MIP, except for problem with 44 aircraft and one runway. This lower bound is not always very tight but it is sufficient for solving the problem within a short CPU time. When the lower bound is too weak as for 44 aircraft and one runway instance, the problem cannot be solved in an acceptable time.

Instance			Our exact method			Beasley et al.'s MIP	
P	R	Opt value	iter	CPU time	LP value	CPU time	LP final value
10	1	700	3	1.1	550	0.4	321.16
	2	90	1	0.5	90	0.6	0
	3	0	1	0.8	0	-	-
15	1	1480	6	5.1	1110	5.2	430
	2	210	1	0.7	210	1.8	0
	3	0	1	1.3	0	-	-
20	1	820	3	1.7	610	2.7	449.40
	2	60	1	0.9	60	3.8	0
	3	0	2	3.5	0	-	-
20	1	2520	5	3.1	2450	220.4	924.37
	2	640	3	3.7	600	1919.9	0
	3	130	4	7.1	120	2299.2	0
	4	0	1	2.6	0	-	-
20	1	3100	8	6.0	3030	922.0	964.83
	2	650	3	4.3	530	11510.4	0
	3	170	6	13.1	90	1655.3	0
	4	0	2	5.4	0	-	-
30	1	24442	4	61.0	5807	33.1	5393.25
	2	554	4	82.0	390	1568.1	0
	3	0	3	58.8	0	-	-
44	1	1550	-	-	89	10.6	184
	2	0	1	15.7	0	-	-

Table 1: Small instances - Linear objective

The results for small instances with non-linear objective are given in Table 2. The description of the instances is reported with P=number of planes, R=number of runways, and optimal value of each instance. This table gives the number of iterations and CPU time required by our cut algorithm, and LP value of the initial problem P_{sc}. For comparison, the results given in [15] are reported. Pinol and Beasley consider two population heuristic algorithms. We have reported

the best result with respect to quality of solution. The column Gap gives their best gap (in percent) against the optimal value, that is $Gap = \frac{heuristic\ value - optimal\ value}{optimal\ value} \times 100$.
We observe that LP value of our method is a good upper bound (recall it is a maximization problem), except for two problems 44 aircraft, one runway and 30 aircraft, one runway. In the first case, gap between LP and optimal value, is huge and the problem cannot be solved. In the second instance, the gap is quickly closed and the problem is solved. The problems are solved in a reasonable CPU time. These times may be greater than those of Pinol and Beasley algorithm but recall that our algorithm is an exact method.

Instance			Our exact method			Pinol, Beasley best heuristic	
P	R	Opt value	iter	CPU time	LP value	CPU time	Gap
10	1	4849	4	2.5	5719	6.0	0
	2	5924	5	4.5	6105	6.0	0
	3	6185	2	2.1	6237	8.0	0
	4	6237	1	1.3	6237	9.0	0
15	1	18337	5	6.0	19178	8.0	0
	2	19948	4	4.0	19966	8.0	0
	3	20078	2	2.9	20078	10.0	0
20	1	35632	6	11.0	37277	9.0	0.28
	2	38524	3	3.9	38559	10.0	0
	3	38664	2	3.8	38664	12.0	0
20	1	20001	5	7.5	20837	70.0	0.59
	2	22888	3	4.8	23453	9.0	0
	3	23659	5	9.5	23865	11.0	0
	4	23955	5	13.3	24140	12.0	0
	5	24140	3	9.7	24140	13.0	0
20	1	19381	8	54.5	22654.7	71.0	1.03
	2	26021	4	5.3	26213	11.0	0
	3	26495	4	7.4	26563	11.0	0
	4	26699	5	13.7	26732	12.0	0
	5	26732	4	14.0	26732	13.0	0
30	1	-2 847013	3	27.5	-325773	15.0	0
	2	-8943	5	121.0	-3826	15.0	0
	3	0	6	140.2	0	14.0	0
44	1	-23266	-	-	584904	24.0	0
	2	644749	1	15.6	644749	21.0	0
	3	646432	1	25.8	646432	21.0	0

Table 2: Small instances - Non-linear objective

The results for the instance with 50 planes are given in Table 3 . The description of the instance is reported with P=number of planes, Lin/NLin=linear/non-linear objective function, R=number of runways and best known value. The value found by our heuristic method is reported with the associated gap (in percent) against the best known value. CPU time and required iteration number are also given. For comparison, Pinol and Beasley results given in [15] are reported. As above, their best results, with respect to quality of solution, are reported. The column Gap gives their gap (in percent) against the best known value.
Our heuristic method provides good solutions (i.e. with low gap) except for the linear case with 2 runways. The computation does not require a lot of iterations. Hence, CPU times are low except for the non-linear instance with one runway.

The results for large instances with linear objective are given in Table 4. The description of the instances is reported with P=number of planes, R=number of runways and best known value of each instance (column Best value). This table gives the value found by our heuristic method and required CPU time. Gap (in percent) is calculated with reference to the best known value as $\frac{heuristic\ value - best\ known\ value}{best\ known\ value} \times 100$. When our heuristic method ameliorates the best

Instance $P = 50$			Our heuristic method				Pinol, Beasley best heuristic	
Lin/NLin	R	Best value	Value	CPU time	Gap	iter	CPU time	Gap
Lin	1	1950	1950	9.7	0	3	287	36.15
	2	135	165	15.9	22.22	3	121	0
	3	0	0	34.0	0	4	139	0
NLin	1	728837	714067	362.0	2.02	7	122	4.65
	2	797116	792609	40.2	0.56	4	22	0
	3	799417	799417	13.1	0	2	24	0

Table 3: Small instance with 50 planes

known solution, the corresponding gap is negative. Our heuristic is the one described in Section 4.3. The maximum number of iterations is set to 4. First iteration result and best result among the four iterations are reported. Each iteration of our algorithm is assigned a maximum CPU run time which is fixed to 120s. Total required CPU times are reported. For comparison, the best results, with respect to quality of solution, given in [15] are reported. The column Gap gives the best gap (in percent) of Pinol and Beasley heuristic, against the best known solution value.
The initial solution (first iteration) of our method is generally good. The quality of the solution is improved with a few number of additional iterations. Our heuristic method gives better solution than Pinol and Beasley heuristic on 14 instances among 24. Moreover, it ameliorates the best known solution of 9 instances. The average of our best gaps is equal to 10.41 percent. Note that in this average, we have excluded 200 planes, 5 runways instance whose gap is undefined. Indeed, the value returned by our heuristic is 1.73 and the best known value is equal to zero. For comparison, the best average of Pinol and Beasley heuristics is equal to 22 percent. Our method requires a few amount of time for these sizes of problem.

The results for large instances with non-linear objective are given in Table 5. The content of this table is the same as previous Table 4. These problems are maximization problems and gaps (in percent) are calculated with reference to the best known value as $\frac{best\ known\ value - heuristic\ value}{best\ known\ value} \times 100$. When our heuristic method ameliorates the best known solution, the corresponding gap is negative. Our heuristic is the one described in Section 4.3. The maximum number of iterations is set to 4. For each iteration, CPU run time limit is fixed to 120s, except for instance with 500 aircraft, 1 runway. This instance is solved for one iteration with a time limit fixed to 600s.. For this instance, 120s. were not sufficient for finding interesting solution.
The results are similar to the linear case. The initial solution (first iteration) of our method is very good except in the case 500 aircraft, 1 runway. A few number of iterations provides substantial benefit. Our heuristic method gives better solution than Pinol and Beasley heuristic on 12 instances among 23. Moreover, it ameliorates the best known solution of 6 instances. The average of our best gaps is equal to 0.92 percent. For comparison, the best average of Pinol and Beasley heuristics is equal to 2.9 percent. Once again, amount of time required for solving these large instances is not excessive.

Instance			Our heurictic method				Pinol, Beasley best heuristic	
P	R	Best value	Value	CPU time	Gap	iter	CPU time	Gap
100	1	5611.7	6119.27	20	9.04	1	554	14.51
			5792.31	263.4	3.22	4		
	2	452.92	520.91	9	15.01	1	342	5.67
			481.78	30.5	6.37	2		
	3	75.75	120.2	15	58.68	1	390	0
			75.75	48.6	0	2		
	4	0	44.5	23.7	-	1	336	0
			0	57.7	0	2		
150	1	12329.31	14523	8.8	17.79	1	925	33.9
			13266.4	132.4	7.60	2		
	2	1288.73	1322.87	13.4	2.65	1	608	7.87
			1172.79	163.2	-9.00	3		
	3	220.79	329.6	22.3	49.28	1	668	8.88
			241,88	70.1	9.55	2		
	4	34.22	66.54	35.6	94.45	1	647	16.74
			45.14	168.8	31.91	3		
	5	0	9.49	50.8	-	1	607	0
			0	382.9	0	2		
200	1	12418.32	13343.3	8	7.45	1	1417	16.67
			12857.2	234.2	3.53	3		
	2	1540.84	1453.77	13.3	-5.65	1	959	9.19
			1335.95	145.5	-13.3	4		
	3	280.82	328.39	18.3	16.94	1	1021	21.59
			285.27	125.2	1.58	4		
	4	54.53	137.59	26.7	152.32	1	993	2.77
			56.26	93.2	3.17	2		
	5	0	28.68	40.8	-	1	956	0
			1.73	328.5	-	3		
250	1	16209.78	18310.7	56	12.96	1	381	22.15
			18024.6	169.1	11.2	2		
	2	1961.39	1816.66	16.4	-7.38	1	1266	18.80
			1790.32	198.8	-8.72	3		
	3	290.4	310.11	28.9	6.79	1	1454	17.48
			248.45	158.6	-14.45	4		
	4	3.49	35.92	33.5	929.23	1	1445	271.63
			2.44	155.9	-30.09	4		
	5	0	0	70.5	0	1	1386	0
500	1	44832.38	41897.3	16.4	-6.55	1	5852	1.03
	2	5501.96	4386.27	12.5	-20.28	1	3836	3.72
			4216.96	375.8	-23.36	4		
	3	1108.51	914.93	15.7	-17.46	1	4560	1.98
			781.9	203.5	-29.46	3		
	4	188.46	167.68	23	-11.03	1	4413	22.98
			132.71	71.2	-29.58	2		
	5	7.35	35.62	29.7	384.63	1	4421	0
			15.9	83.9	116.33	2		

Table 4: Large instances - Linear objective

Instance			Our heuristic method				Pinol, Beasley best heuristic	
P	R	Best value	Value	CPU time	Gap	iter	CPU time	Gap
100	1	10 926459	10 903841	18.5	0.21	1	177	11.34
	2	13 690290	13 671934	12.1	0.13	1	188	0.43
			13 690376	36.4	-0.0006	2		
	3	14 037508	14 023612	17.9	0.10	1	50	0.01
			14 037508	43.3	0	2		
	4	14 090232	14 073032	25.9	0.12	1	48	0
			14 086928	78.5	0.02	2		
150	1	14 713752	13 585847	50.2	7.67	1	223	12.38
			13 857413	174.8	5.82	2		
	2	19 829588	19 788658	16.0	0.21	1	261	0.08
			19 797830	365.5	0.16	4		
	3	20 126522	20 107648	26.4	0.09	1	282	0
			20 126522	64.7	0	2		
	4	20 136384	20 112323	39.7	0.12	1	84	0
			20 136384	111.8	0	2		
200	1	21 827542	21 184939	42.3	2.94	1	323	13.29
			21 324444	352.8	2.30	4		
	2	26 786444	26 808643	18.2	-0.08	1	154	0
			26 862823	105.9	-0.29	3		
	3	27 409004	27 330969	28.4	0.28	1	163	0.01
			27 395708	95.2	0.05	3		
	4	27 484328	27 444917	41.2	0.14	1	137	0.01
			27 481119	222.9	0.01	4		
	5	27 506007	27 460372	81.6	0.17	1	120	0
			27 504531	164.1	0.01	2		
250	1	27 619476	27 001408	51.3	2.24	1	416	13.32
			27 028285	105.3	2.14	2		
	2	33 676680	33 800828	11.9	-0.37	1	467	0.15
			33 809839	107.7	-0.40	4		
	3	34 330656	34 312694	19.3	0.05	1	229	0.03
			34 343826	95.0	-0.04	3		
	4	34 401946	34 370653	27.8	0.09	1	201	0
			34 401946	70.2	0	2		
	5	34 405915	34 374622	51.2	0.09	1	179	0
			34 405915	126.0	0	2		
500	1	45 677264	40 108125	600.0	12.19	1	1268	15.94
	2	61 557052	62 117925	13.0	-0.91	1	688	0.01
			62 149293	315.4	-0.96	4		
	3	63 641468	63 573672	16.5	0.11	1	1212	0.02
			63 666346	155.0	-0.04	3		
	4	63 886852	63 787254	23.7	0.16	1	674	0
			63 880612	191.8	0.01	4		
	5	63 891976	63 807412	30.9	0.13	1	687	0
			63 885152	169.8	0.01	3		

Table 5: Large instances - Non-linear objective

6 Conclusion

We have proposed a new approach for solving the Aircraft Landing Problem. It is based on the approximation of the separation time matrix by a rank 2 matrix and on discretization of planning horizon. The approximation leads to interesting lower bound that has been used to exactly solve the problem by a cut algorithm. The method has been used to heuristically solve large instances of the problem using the best approximation matrix. Time discretization provides easiness in modeling. Various objective functions, runway dependent constraints can

be easily modelized by this approach.
The overall method is quite simple and does not require a lot of algorithmic development. It requires the use of a solver. It could be applied to take-off problem.
Future work and improvement are worth to be explored. The method is based on scenarios, each scenario representing landing time of a plane. If the planning horizon is large, the number of scenarios is huge. It could be interesting to develop a column generation algorithm in order to consider only interesting scenarios. There are a lot of rank 2 approximation matrices for a given separation time matrix. It could be interesting to find the one that maximizes the lower bound of the objective (in a minimization problem) in order to improve efficiency of cut algorithm.

References

[1] K. Artiouchine, P. Baptiste, C. Durr (2008), *Runway Sequencing with Holding Patterns*, European Journal of Operational Research **189(3)**, 1254–1266.

[2] J.E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha, D. Abramson (2000), *Sheduling Aircraft Landings-The static case*, Transportation Science **34(2)**, 180–197.

[3] J.E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha, D. Abramson (2004), *Displacement problem and dynamically scheduling aircraft landings*, Journal of the Operational Research Society **55**, 54–84.

[4] J.E. Beasley, J. Sonander, P. Havelock (2001), *Sheduling aircraft landings at London Heathrow using a population heuristic*, Journal of the Operational Research Society **52**, 483–493.

[5] J.E. Beasley (1996), *Obtaining test problems via Internet* , Journal of Global Optimization **8**, 429–433. Available from: http://people.brunel.ac.uk/ mastjjb/jeb/orlib/airlandinfo.html

[6] G. Bencheikh, J. Boukachour, A. El Hilali Alaoui, F. El Khoukhi (2009), *Hybrid method for aircraft landing scheduling based on a Job Shop formulation*, International Journal of Computer Science and Network Security **9(8)**, 78–88.

[7] L. Bianco, P. Dell'Olmo, S. Giordani (2006), *Scheduling models for air traffic control in terminal areas*, Journal of Scheduling **9(3)**, 223–253.

[8] C. Diallo, B. M. Ndiaye, D. Seck (2012), *Sheduling aircraft landings at LSS Airport*, American Journal of Operations Research **2**, 235–241.

[9] J. F. Diaz, J. A. Mena (2005), *Solving the aircraft sequencing problem using Concurrent Constraint Programming*, Lecture Notes in Computer Science **3389**, 292–304.

[10] A. D'Ariano, M. Pistelli, D. Pacciarelli (2012), *Aircraft retiming and rerouting in vicinity of airports*, IET Intelligent Transport Systems **6(4)**, 433–443.

[11] A. T. Ernst, M. Krishnamoorthy, R. H. Storer (1999), *Heuristic and Exact Algorithms for Scheduling Aircraft Landings*, Networks **34(3)**, 229–241.

[12] T. Fahle, R. Feldmann, S. Gotz, S. Grothklags, B. Monien (2003), *The aircraft sequencing problem*, Lecture Notes in Computer Science **2598**, 152–166.

[13] J. V. Hansen (2004), *Genetic search methods in air traffic control*, Computers and Operations Research **31**, 445–459.

[14] P. van Leeuwen, H. Hesselink, J. Rohling (2002), *Scheduling aircraft using constraint satisfaction*, Electronic Notes in Theoritical Computer Science **76**, 252–268.

[15] H. Pinol, J.E. Beasley (2006), *Scatter Search and Bionomic Algorithms for the aircraft landing problem*, European Journal of Operational Research **171**, 439–462.

[16] M. J. Soomer, G. J. Franx (2008), *Scheduling aircraft landings using airlines' preferences*, European Journal of Operational Research **190**, 277–291.