International Journal of Computer Theory and Engineering, Vol. 3, No. 2, April 2011
ISSN: 1793-8201

Improved Ant Colony Algorithm to Solve the Aircraft Landing Problem

Ghizlane Bencheikh, Jaouad Boukachour and Ahmed EL Hilali Alaoui

Abstract— **Scheduling aircraft landing is a complex task encountered by most of the control towers. In this paper, we study the aircraft landing problem (ALP) in the multiple runway case. We present in the first part, a mathematical formulation of the problem with a linear and nonlinear objective function. In the second part, we consider the static case of the problem where all data are known in advance and we present a new heuristic for scheduling aircraft landing on a single runway, this heuristic is incorporated into an ant colony algorithm to solve the multiple runway case.**

Index Terms— **Aircraft landing, Ant Colony Optimization, Mathematical programming, Scheduling.**

I. Introduction

When an aircraft enter in an airport radar range (or radar horizon), it requires from air traffic control to assign it the authorization to land, a landing time and an appropriate runway if several runways are available. The landing time belongs to a predefined interval called landing window, bounded by an earliest and latest landing time. The earliest landing time corresponds to the time at which the aircraft could land if it use its fastest speed (which is not economical for aircraft) while the latest landing time depends on its autonomy of carburant.

Within the landing window, there is a target landing time that represents a preferred landing time and which corresponds to the time that aircraft could land if it flies at its cruise speed which is the most economical speed of the aircraft, it corresponds to the time announced to passengers. Any deviation from the target time causes disturbances in the airport. Consequently, a penalty cost is associated with each deviation before or after the target time of an aircraft. So, the objective is to minimize the total cost of penalties such as:

An interval of security between two successive landings on the same runway must be respected;

1) An interval of security that must separate two successive landings on different runways must be respected;
2) Each aircraft must land within a predetermined time window [Earliest landing time, Latest landing time].

Manuscript received September 20, 2010.

Dr. G. Bencheikh is with the Department of Economics, Faculty of Law, Economic and Social Sciences, Meknes, Morocco (e-mail: ghizlane_bencheikh@yahoo.fr).

Prof. Dr. J. Boukachour is with the Department of Computer Sciences, Le Havre University Institute of Technology, Le Havre, France (e-mail: boukachour@univ-lehavre.fr)

Prof. A. EL Hilali Alaoui is with the Department of Mathematics, Faculty of Science and Technology, Fez, Morocco (e-mail: ahmed.elhilali@fst-usmba.ac.ma).

II. Previous work

The aircraft landing problem (ALP) has been studied by several researchers in different countries. Abel et al. [1] presented a formulation of the problem as a linear mixed program which is exactly solved by a Branch and Bound algorithm (B&B), they present another approach by applying Genetic Algorithm (GA), a comparison of these two methods (B&B and GA) is presented. In [9], V. Ciesielski et al. applied a Genetic Algorithm to the ALP in a two runway case. In [20], A.T. Ernst et al. considered two versions of the ALP: the static and the dynamic one. They proposed both exact methods and heuristics for the resolution. J.E. Beasley et al. presented in [4] a mixed linear program formulation of the ALP in the static case; they resolved it by a method based on relaxation of the binary variables with some additional constraints. Computational results are presented involving up to 50 aircraft and 4 runways. A particular case in [6] has been presented by J.E. Beasley et al., they developed a heuristic population for improving the aircraft landing at Heathrow airport. They used this algorithm to solve the ALP in the dynamic case [5]. A.T. Ernst and M. Krishnamoorthy [19] presented two resolution methods, a Branch and Bound method and a Genetic Algorithm to solve the ALP. In the single runway case, J. Boukachour and A. El Hilali Alaoui [8] presented an application of a Genetic Algorithm. In [25] H. Pinol and J.E. Beasley presented a mathematical formulation of the ALP with a linear and non linear objective functions; they presented two approaches: Scatter Search and Bionomic Algorithm, computational results are presents involving up to 500 aircraft and 5 runways. N. Baüerle et al. are interested in reducing the waiting time of one or two runways in [3]; they presented a model for the landing procedure of aircraft. In [2] K. Artiouchine et al. are more interested by the complexity of the problem, they discussed several cases solved in polynomial time and presented a compact mixed integer programming formulation to solve large instances of the general problem where all time windows have the same size. They proposed a general hybrid branch and cut framework. In [27], M.J. Soomer and G.J. Franx studied the single runway arrival problem, they presented a local search heuristic specific to the problem where they assign a landing time to each flight taking into account the cost provided by the airline. This cost is related to the arrival delays of the flights. M.J. Soomer and G. Koole [28] used the aircraft landing problem to obtain an efficient and fair schedule with little cost for airlines. Their objective is to allow the airlines to provide different cost functions for each individual flight which has other characterizations different to its landing

time.

In this paper, we consider the aircraft landing problem on one and multiple runways. We present in the first part a mathematical formulation of the problem. In the second part, we present a new heuristic for scheduling aircraft landing on one runway. This heuristic is incorporated into an ant colony algorithm which we apply to the multiple runway case. Computational results are presented in the last section involving up to 50 aircraft and 5 runways.

The rest of this paper is organized as follows: the third section presents a mathematical formulation of the problem. In the 4th section we give an adaptation of the Ant Colony Algorithm to ALP. The 5th section presents a new heuristic to improve the scheduled landing time on a single runway. In section VI; we improve the ant colony algorithm by incorporating the heuristic presented in section V. Section VII is dedicated to computational results.

III. Mathematical Formulation

In the following section, we give a new mathematical formulation of the static case of the ALP based on the classical formulation presented in [4].

A. Notations

We use for the formulation, the following notations:

Problem's data

N : the number of aircraft waiting to land,
R : the number of available runways,
e_i : the earliest landing time for aircraft i,
l_i : the latest landing time of aircraft i,
ta_i : the target landing time for aircraft i,
Pb_i: penalty cost by one unit of time for aircraft i if it lands before its target time,
Pa_i : penalty cost by one unit of time for aircraft i if it lands after its target time,
S_{ij} : the separation time between aircraft i and aircraft j ($S_{ij} > 0$, $i \neq j$), if i lands before j on the same runway
s_{ij} : the separation time between aircraft i and aircraft j ($s_{ij} \geq 0$, $i \neq j$), if i lands before j on a different runways. In the following, we suppose that matrix of separation is symmetric ($S_{ij} = S_{ji}$ and $s_{ij} = s_{ji}$)

Decision variables

t_i : the scheduled landing time of the aircraft i
er_i : the advance made by aircraft i, $er_i = max\ (0,\ ta_i - t_i)$
tr_i : the tardiness made by aircraft i, $tr_i = max\ (0,\ t_i - ta_i)$

$$x_{ij} = \begin{cases} 1 & \text{if aircraft i lands before j} \\ -1 & \text{otherwise} \end{cases}$$

$$y_{ir} = \begin{cases} 1 & \text{if aircaft i lands on runway r} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{if aircarfts i and j land on the same runway} \\ 0 & \text{otherwise} \end{cases}$$

Note that similar variables have been used in some previous papers [4] [25]:

$$\mathbf{x_{ij}} = \begin{cases} \mathbf{1} & \text{if aircraft i lands before j} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$y_{ir} = \begin{cases} 1 & \text{if aircaft i lands on runway r} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{if aircarfts i and j land on the same runway} \\ 0 & \text{otherwise} \end{cases}$$

B. Constraints

For each aircraft, its scheduled landing time must belong to the landing window, $[e_i, l_i]$

$$e_i \leq t_i \leq l_i \qquad \forall i = 1, \ldots, N \tag{1}$$

The following constraints show that there are two cases: i lands before j or j lands before i.

$$x_{ij} + x_{ji} = 0 \qquad \forall i, j = 1, \ldots, N;\ j > i \tag{2}$$

$$x_{ij} \in \{-1, 1\} \qquad \forall i, j = 1, \ldots, N \tag{3}$$

Note that in [4], we found similar constraints:

$$x_{ij} + x_{ji} = 1 \qquad \forall i, j = 1, \ldots, N;\ j > i \tag{2a}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j = 1, \ldots, N \tag{3a}$$

In some cases, we can immediately decide if $x_{ij} = 1$ or $x_{ij} = -1$. For example, if $l_i < e_j$ then $x_{ij} = 1$ and $x_{ji} = -1$.

The separation constraints must be respected:

$$x_{ij}.t_j \geq x_{ij}.t_i + S_{ij}.z_{ij} + s_{ij}(1 - z_{ij}) \qquad \forall i, j = 1, \ldots, N;\ j > i \tag{4}$$

Let (i, j) be a pair of aircraft such as $i < j$ and suppose that aircraft i and j land on the same runway, i.e. $z_{ij} = 1$, ($1 - z_{ij} = 0$)

- If the aircraft i lands before aircraft j then $x_{ij} = 1$, the constraint (4) becomes :

$$t_j \geq t_i + S_{ij}$$

- If the aircraft j lands before aircraft i then $x_{ij} = -1$, the constraint (4) becomes :

$$-t_j \geq -t_i + S_{ij}$$

Since $S_{ij} = S_{ji}$, (The matrix (S_{ij}) is symmetric), we have:

$$t_i \geq t_j + S_{ji}$$

We can conclude that the two situations, aircraft i lands before aircraft j or j lands before i, can be expressed by the constraint (4).

In [25], H. Pinol and J. E. Beasley have considered the following constraint:

$$t_j \geq t_i + S_{ij}.z_{ij} + s_{ij}(1 - z_{ij}) - M.x_{ji} \qquad \forall i, j = 1, \ldots, N;\ j \neq i \tag{4a}$$

Where M is a great positive number and

$$\mathrm{x_{ij}} = \begin{cases} 1 & \text{if aircraft i lands before j} \\ 0 & \text{otherwise} \end{cases}$$

Let (i, j) be a pair of aircraft such as $i \neq j$ and suppose that aircraft i and j land on the same runway, i.e. $z_{ij} = 1$ ($1 - z_{ij} = 0$)

- If the aircraft i lands before aircraft j then $x_{ij} = 1$, the constraint (4a) becomes :

$$t_j \geq t_i + S_{ij}$$

- If the aircraft j lands before aircraft i then $x_{ij} = 0$, the constraint (4a) becomes :

$$t_j \geq t_i + S_{ij} - M$$

We can observe that our mathematical formulation of the last constraint is an improvement of the constraint (4a). In the constraint (4a), we must consider $N^2 - N$ relations (expressed

by $\forall i, j = 1,..., N; i \neq j$). In our formulation, constraint (4) considers only ($N^2 - N)/2$ relations (expressed by $\forall i, j = 1, ..., N; i < j$).

The deviation before and after target time are expressed by the constraints (5), (6), (7), (8) and (9) below:

$$er_i \geq ta_i - t_i \qquad \forall i = 1, \ldots, N \tag{5}$$

$$0 \leq er_i \leq ta_i - e_i \qquad \forall i = 1, \ldots, N \tag{6}$$

$$er_i \geq t_i - ta_i \qquad \forall i = 1, \ldots, N \tag{7}$$

$$0 \leq tr_i \leq l_i - ta_i \qquad \forall i = 1, \ldots, N \tag{8}$$

$$t_i = ta_i - er_i + tr_i \qquad \forall i = 1, \ldots, N \tag{9}$$

We introduce the following constraint to express the fact that an aircraft must be landed on one runway:

$$\sum_{r=1}^{R} y_{ir} = 1 \qquad \forall i = 1, \ldots, N \tag{10}$$

The matrix (z_{ij}) is symmetric

$$z_{ij} = z_{ji} \qquad \forall i, j = 1, \ldots, N;\ j > i \tag{11}$$

Constraints (12) link the variables y_{ir}, y_{jr}, and z_{ij}:

$$z_{ij} \geq y_{ir} + y_{jr} - 1$$
$$\forall i, j = 1, \ldots, N;\ j > i\ ;\ \forall r = 1, \ldots, R \tag{12}$$

Remark:

However, if one aircraft i or j lands on runway r and the other doesn't, we must have $z_{ij} = 0$, this isn't guarantied by (12), we can introduce the following constraint:

$$\forall i, j = 1,..., N,\ j > i,\ \forall r = 1,...,R$$
$$y_{ir}.y_{jr} \leq z_{ij} \leq y_{ir}.y_{jr} + (y_{ir} - 1)(y_{jr} - 1) \tag{12a}$$

If :

$(y_{ir}, y_{jr}) = (0,0)$	then	$0 \leq z_{ij} \leq 1$
$(y_{ir}, y_{jr}) = (0,1)$	then	$0 \leq z_{ij} \leq 0$
$(y_{ir}, y_{jr}) = (1,0)$	then	$0 \leq z_{ij} \leq 0$
$(y_{ir}, y_{jr}) = (1,1)$	then	$1 \leq z_{ij} \leq 1$

C. Objective function

The objective is to minimize the penalty cost of deviation between the actually landing time of all aircraft and their target landing times.

$$Min\left(\sum_{i=1}^{N} er_i.Pa_i + tr_i.Pr_i\right) \tag{13}$$

The mathematical program in this formulation is to minimize (13) such as constraints (1) - (12)

If we suppose that intervals of security are symmetric, our mathematical program is to minimize (13) under constraints (1)-(3), (4a), (5)-(12)

In [25], another objective function is defined. The aim is to land the aircraft as soon as possible to their earliest landing time. This is expressed by the following function:

$$Max \quad \sum_{i=1}^{N} D_i$$

where

$$D_i = \begin{cases} -d_i^2 & \text{if } d_i \geq 0 \\ +d_i^2 & \text{otherwise} \end{cases}$$

d_i deviationof theaircraftlandingtimefromits targetlandingtime

In this paper, we express this objective function by:

$$Max \quad \sum_{i=1}^{N} (er_i)^2 - (tr_i)^2 \tag{14}$$

The complete mathematical program is to maximize this objective function under constraints (1) - (12). In this case, the goal is to promote the advance of aircraft landing in order to use efficiently the runway.

IV. Ant Colony Optimization

Ant Colony Optimization (ACO) was first presented by Marco Dorigo in 1992 [14], who was inspired by the collect behavior of natural ants. Based on this process, ant colony algorithm consists on a number of artificial ants in charge to find the optimal solution of a combinatorial problem and communicate through pheromones [17]. The Ant Colony Optimization has been initially applied on traveling sales man problem (TSP) to find the shortest Hamiltonian path in a complete graph [15], [16]. Each ant constructs its own path and puts a quantity of pheromone according to its solution quality. The shortest path is the one with the greater quantity of pheromone. ACO was applied on both static and dynamic combinatorial problems including machine scheduling [24], job shop scheduling [11], [21], vehicle routing [13], [26], [7], graph coloring [10], knapsack problem [23], etc.

In this section, we propose an adaptation of ACO on the Aircraft landing problem in the static multiple runway case; this algorithm can be adapted to the dynamic case to take into account eventual uninspected changes in data such as the closing of a runway or the cancellation of a flight.

A. Graphical representation

In order to apply the Ant colony Algorithm, we define the following graphical representation of the problem. This is based on a bi-level graph. In the first level, we fix the available runways and in the second level the aircraft. We add two dummy nodes D and F corresponding respectively to the beginning and the end of graph.

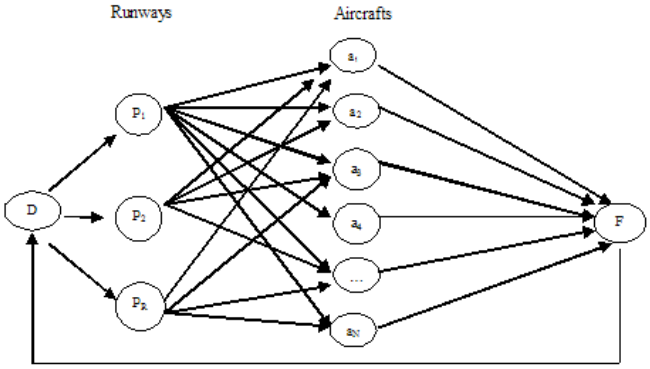


Fig.1. Graphical Representation of the problem

B. Construction of a solution

An ant begins its trajectory by the beginning of the graph corresponding to the node dummy D. First, it chooses the runway where it will insert the next aircraft; this choice may depend on the charge on the runway, or the runway that will be free sooner. After the choice of the runway, the ant has to choose the next aircraft to land on this runway; this choice depends essentially on the priority of the aircraft compared to the other aircraft and the memory of the ant colony. This

process is repeated until there is no aircraft available to land.

Runway selection

The probability rule to select a runway *r*, from the beginning of the graph *D* can be expressed by the following equation:

$$P_{Dr}^{k} = \begin{cases} \arg\min_{r=1,\ldots,R} \begin{pmatrix} number\,of\,aircrafts \\ affected\,to\,the\,runway\,r \end{pmatrix} & if\ q \prec q_0 \\ r_0 & otherwise \end{cases}$$

(*eq. 1*)

Where:

- *D* represents the node corresponding to the beginning of the graph
- $0 < q_0 < 1$ is a constant of the algorithm to ensure the diversification
- *q* is a value taken randomly into [0,1]
- r_0 is an index chosen randomly in *{1, ..., R}*

In this case, we select the runway with the smallest number of aircraft, without taking into consideration the intervals of security. This probability rule can be useful if the intervals of security are the same for all aircraft but actually, they depend of the nature of aircraft. That's why, we propose another probability rule which selects the runway according to its availability time to receive new aircraft. It is expressed in this paper by the following equation:

$$P_{Dr}^{k} = \begin{cases} \arg\min_{r=1,\ldots,R} \left(\min_{j \in Candidate_k} \left(x_{i0}^{r} + S_{i_0 j} \right) \right) & if\ q \prec q_0 \\ r_0 & otherwise \end{cases}$$

(*eq. 2*)

Where:

- $Candidate_k$ is the ant's *k* list of aircraft waiting to land
- i_0 the last aircraft affected to the runway *r*
- $x_{i_0}^{r}$ is the landing time of the last aircraft affected to the runway *r* for the ant *k*

Aircraft selection

After choosing a runway *r*, the ant has to choose an aircraft for this runway. This choice depends on two parameters. The first is the priority of the aircraft such as the earliest landing time, target time, penalty cost of the aircraft. The second is the cost calculated after we assign a landing time to the aircraft using the assignment algorithm presented in the next section. A weight of these two parameters is the "information heuristic" of the ant colony algorithm.

$$\eta_{rj} = \left(\frac{1}{(\text{Priority}(j)+1)} \right)^{\beta_1} \cdot \left(\frac{1}{Cost\_penalty(j)+1} \right)^{\beta_2}$$

Where:

- *Priority (j)* : The priority of the aircraft *j*,
- *Cost_penalty(j)* : Penalty cost of the aircraft *j*
- β_1 *and* β_2 : Coefficients of weighting

The other parameter that influences an ant's choice is the colony memory (i.e, pheromone trails noted τ_{ij} initially fixed on a value τ_0). To summarize, the probability rule to choose an aircraft is expressed by:

$$p_{rj}^{k}(t) = \begin{cases} \dfrac{(\tau_{rj})^{\alpha} \cdot (\eta_{rj})^{\beta}}{\sum_{l} (\tau_{rl})^{\alpha} \cdot (\eta_{rl})^{\beta}} & if\ j \in Candidate_k \\ 0 & otherwise \end{cases}$$

(*eq. 3*)

α and β define the relative importance of the pheromone trace and the visibility of the ants.

Assignment of aircraft landing times

This step is to assign landing times to aircraft while respecting the two constraints:

- *The landing time must be within the landing widow* $[e_i, l_i]$
- *The interval of security must be respected*

We have used two heuristics to assign aircraft landing times. The first heuristic assigns the target landing time if it respects the intervals of security between the previous aircraft, otherwise, it assigns the earliest time which respects the intervals of security. It can be expressed as follows:

$$t_j = \max\left(ta_j, \max_{i \in O}\left(t_i + S_{ij}\right)\right)$$

Where:

O : Set of aircraft which have previously been assigned a landing time.

The second heuristic used in this paper just cares about the intervals of security between the other aircraft and respecting the landing time window.

$$t_j = \max\left(e_j, \max_{i \in O}\left(t_i + S_{ij}\right)\right)$$

Both expressions are used to assign a landing time to aircraft in a single runway, what explain the use of the matrix of security (S_{ij}).

Evaporation of the pheromone trail

The pheromone trail is updated at each iteration end according to the following equation:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \Delta \tau_{ij}(t) \qquad (eq.\ 4)$$

Where

- ρ is a coefficient of evaporation ($\rho < 1$ to avoid an unlimited accumulation of trace)
- $\Delta \tau_{ij}$ is the quantity of trace left on the edge *(i, j)* by the colony at the end of an iteration:

$$\Delta \tau_{ij} = \begin{cases} \dfrac{Q}{C} & if\ (i,j) \in Best\ solution \\ 0 & otherwise \end{cases}$$

Q : Updating constant

C : Penalty cost of the best solution in iteration *t*

Best solution : The path with the smaller penalty cost

We can summarize the ant colony algorithm as follows:

Ant Colony Algorithm adapted to the aircraft landing problem

1. *Initialize the matrix of pheromone trails*
2. *For each ant k*
   i. *Initialize the first aircraft of each ant's runway*
   ii. *Initialize the list of candidate aircraft*
   iii. *Repeat*
      - *Select a runway r according to (eq. 2)*
      - *Select an aircraft according to (eq. 3)*
      - *Insert the aircraft j in the list of aircraft affected to the runway r and delete it from the candidate list*
      - *Assign a landing time to the aircraft j*

- *Return to the beginning of the graph*
  While the candidate list is not empty
3. *Update the pheromone trail according to (eq. 4)*
4. *Repeat steps 2, 3 and 4 until verify the stopping criterion*

V. New heuristic to Improve Scheduling Aircraft Landing Times

We introduce a new heuristic to improve the computed aircraft landing time in order to minimize the total penalty cost, in the single runway case; this heuristic will be incorporated into the ant colony algorithm which is applied for the multiple runway case.

Improvement algorithm

The improvement algorithm consists on reducing the total penalty cost made by all aircraft. The algorithm is applied on a sequence of aircraft which has already been assigned landing times, and it adjusts the landing time in order to reduce the total penalty cost. Before applying the algorithm, we have to set an order between aircraft and assign for them landing times.

Aircraft sequencing

The order between aircraft is set up according to priority rules which are based on the variables:

- e_i : The priority is given to the aircraft which has the sooner earliest landing time; this priority rule can be used if we are interested by the maximal exploitation of the runway.
- ta_i : The priority is given to the aircraft which has the earliest target landing time; We can choose this priority if we want to minimize the advance or late made by aircraft
- l_i : The priority is given to the aircraft which has the earliest latest landing time, in order to avoid that, we assign a landing time after the latest landing time of an aircraft.
- e_i/Pb_i : The priority is given to the aircraft which has the soonest earliest time and which causes the most important advance penalty.
- l_i/Pa_i : The priority is given to the aircraft which has the soonest latest time and which causes the most important lateness penalty.
- ta_i/Pb_i+Pa_i : The priority is given to the aircraft which has the soonest target time and which causes the most important advance and lateness penalty.
- $1/Pb_i+Pa_i$: The priority is given to the aircraft which causes the most important advance and lateness penalty.

To assign landing times to the aircraft we use the same heuristics presented above.

Adjusting aircraft landing times

The adjusting landing time is the most important step in the improvement algorithm; the aim is to reduce the total cost of penalty caused by all aircraft. Based on the landing times assigned to aircraft, we modify the landing time of a selected aircraft *i*, as follows: if the aircraft *i* lands in advance (resp. late), we increase (resp. reduce) its landing time by one unit of time, in this case we have to check the feasibility of the solution: the new landing time must respect the interval of security between the next (resp. previous) ones, if it is not the case, we have to increase (resp. reduce) the landing times of next (resp. previous) aircraft to keep the respect of intervals of security. We have to check that the new landing time should not be outside of the landing window; in this case we cancel the increase (resp. reduce) and keep the last feasible solution.

Remark:

If we increase (resp. reduce) the aircraft landing time, we check the interval of security with the next (resp. previous) aircraft only, because it always respect the intervals with the previous (resp. next) aircraft.

We can define two versions of the improvement heuristic:

– Improving the landing aircraft time during the initialization *(Parallel improving)*: in this algorithm, we adjust the landing time of each aircraft during its time assignment.
– Improving the aircraft landing time after the initialization *(Global improving)*: in this algorithm, we assign the landing time to all aircraft, after what, we adjust the landing time of the aircraft which increases the penalty cost. We apply the adjustment a number of iterations where in each iteration an aircraft is selected by the transition rule :

$$j = \begin{cases} \underset{k \in (1,\cdots,N)}{\operatorname{argmax}} [\text{cost penalty}(a_k)] \\ \qquad \text{if the last change was fruitful} \\ j_0 \qquad\qquad \text{otherwise} \end{cases} \qquad (eq.\ 5)$$

Where:

- $cost\_penalty(a_k)$ corresponds to the penalty cost made by aircraft *k*
- j_0 an aircraft selected randomly

We can summarize both algorithms as follows:

Parallel improving of aircraft landing times:
Let P be the list of aircraft set up according to a priority rule
1. $t_{P1} \leftarrow ta_{P1}$
2. *For i from 2 to N*
   $X_{Pi} \leftarrow max(e_{Pi}\ (or\ ta_{pi}),\ \max_{j \in O}\ (t_j+S_{j,Pi}))$
   end for
3. *Repeat*
   if($t_{Pi} > ta_{Pi}$)
   Reduce the landing time by 1 unit of time
   else
   Increase the landing time by 1 unit of time
   end if
   if(the solution is unfeasible)
   Reject the change and keep the last feasible solution
   break
   end if
while (there is decrease of penalty cost)

Global improving of aircraft landing times:
Let P be the list of aircraft set up according to a priority rule
1. $t_{P1} \leftarrow ta_{P1}$
2. *For i from 2 to N*
   $t_{Pi} \leftarrow max(e_{Pi}\ (or\ ta_{pi}),\ \max_{j \in O}\ (t_j+S_{j,Pi})$
   end for
3. *Select the aircraft i_0 causing the greater penalty cost*
4. *For k number of iterations*
   if($t_{i0}>ta_{i0}$)
   Reduce the landing time by 1 unit of time
   else
   increase the landing time by 1 unit of time

end if
if(the solution is unfeasible)
Reject the change and keep the last feasible solution
continue
end if
Select a new aircraft according to (eq. 5)
End For

We have tested both algorithms on instances involving 10 to 50 aircraft on one runway. We remember that these algorithms are tested on the single runway case in order to incorporate it in the Ant Colony Algorithm for the multiple runway case. The computational results are presented in table 3 and table 4 in section VII, we have tested the priority rules and both affectation heuristic.

A use case of the parallel improving:

Let be 10 aircraft waiting to land where the landing windows, the penalties and the security intervals are presented by the following tables:

TABLE I. LANDING WINDOWS, TARGET LANDING TIME AND PENALTY COST

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	*0*	*3*	*15*	*15*	*15*	*15*	*15*	*15*	*15*	*15*
2	*3*	*0*	*15*	*15*	*15*	*15*	*15*	*15*	*15*	*15*
3	*15*	*15*	*0*	*8*	*8*	*8*	*8*	*8*	*8*	*8*
4	*15*	*15*	*8*	*0*	*8*	*8*	*8*	*8*	*8*	*8*
5	*15*	*15*	*8*	*8*	*0*	*8*	*8*	*8*	*8*	*8*
6	*15*	*15*	*8*	*8*	*8*	*0*	*8*	*8*	*8*	*8*
7	*15*	*15*	*8*	*8*	*8*	*8*	*0*	*8*	*8*	*8*
8	*15*	*15*	*8*	*8*	*8*	*8*	*8*	*0*	*8*	*8*
9	*15*	*15*	*8*	*8*	*8*	*8*	*8*	*8*	*0*	*8*
10	*15*	*15*	*8*	*8*	*8*	*8*	*8*	*8*	*8*	*0*

TABLE II. TIME BETWEEN SUCCESSIVE ARRIVAL AIRCRAFT

Aircraft	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Earliest landing time	129	195	89	96	110	120	124	126	135	160
Target landing time	155	258	98	106	123	135	138	140	150	180
Latest landing time	559	744	510	521	555	576	577	573	591	657
Penalty before target time	10	10	30	30	30	30	30	30	30	30
Penalty after target time	10	10	30	30	30	30	30	30	30	30

1. The first step is to set up an order between aircraft according to a priority rule. Suppose that the priority rule is the earliest target time. The order is as follows:

Aircraft(a_i)	3	4	5	6	7	8	9	1	10	2

2. The second step is to assign a landing time to the first aircraft in the list (aircraft 3 in our case) by one of the assignment heuristic, let's apply the first one i, e.:

$$t_j = \max\left(ta_j, \max_{i \in O}\left(t_i + S_{ij}\right)\right)$$

The aircraft 3 is assigned its target time which is 98 :

Aircraft (a_i)	3	4	5	6	7	8	9	1	10	2
Landing Times (t_i)	98									

3. For the second aircraft in the list: (aircraft 4)

- The landing time calculated by the assignment heuristic is 106 which is the max (98+8, 106)

a_i	3	4	5	6	7	8	9	1	10	2
t_i	98	106								

- Adjusting time: The aircraft 4 has landed at its target time, so we don't have to adjust its landing time.

For the aircraft 5: the landing time calculated by the same assignment heuristic: 123 is also the target landing time

a_i	3	4	5	6	7	8	9	1	10	2
t_i	98	106	123							

For the aircraft 6, the calculated landing time is 135:

3	4	5	6	7	8	9	1	10	2
98	106	123	135						

For the aircraft 7, the calculated landing time is: 143 is greater than its target landing time, so we have to adjust time: 143 is greater than 138, then we reduce the landing time by 1 unit, so the new landing time is 142. We have to check the feasibility of the solution:

a_i	3	4	5	6	7	8	9	1	10	2
t_i	98	106	123	134	142					

The landing time of aircraft 6 is reduced by 1 unit to keep the feasibility of solution. If the penalty cost of aircraft 6 were 10 then the adjustment time will reduce the penalty cost from 5*30 = 150 to 4*30 + 1 * 10 = 130. On the contrary, if the penalty cost of aircraft 6 was 40, then the adjustment will increase the total penalty cost, in this case, we reject the new landing times and keep the last ones.

For aircraft 8: the assigned landing time (after applying the adjustment) is

a_i	3	4	5	6	7	8	9	1	10	2
t_i	98	106	122	130	138	146				

For aircraft 9:

a_i	3	4	5	6	7	8	9	1	10	2
t_i	98	106	121	129	137	145	153			

For aircraft 1:

3	4	5	6	7	8	9	1	10	2
98	106	120	128	136	144	152	167		

For aircraft 10:

3	4	5	6	7	8	9	1	10	2
98	106	118	126	134	142	150	165	180	

For aircraft 2:

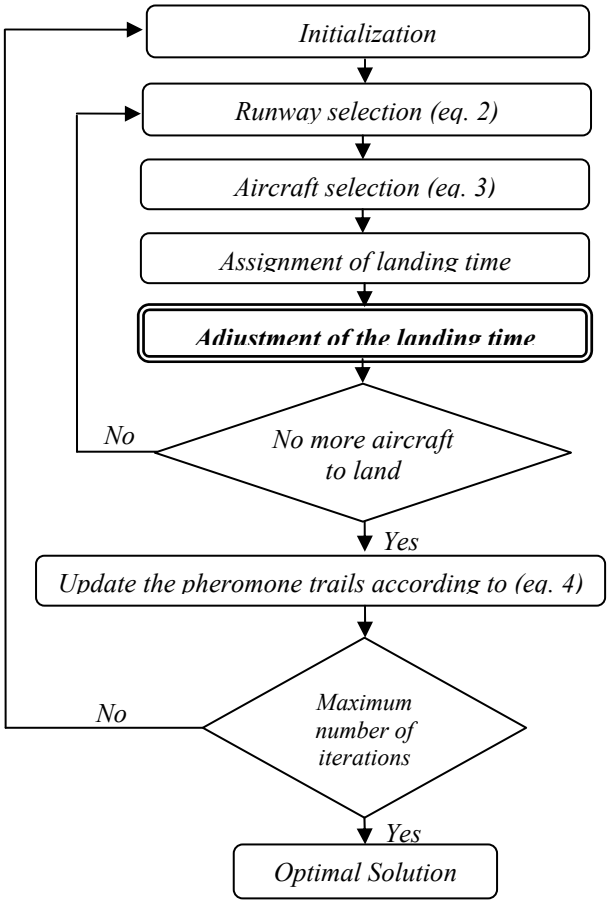
3	4	5	6	7	8	9	1	10	2
98	106	118	126	134	142	150	165	180	258

The total penalty cost is: 700 which coincide with the optimal solution calculated by ILOG Cplex applied on the linear program of the problem.

VI. IMPROVED ANT COLONY ALGORITHM (IACA)

To improve the ant colony algorithm, we combine it with the improvement algorithm presented above. This algorithm is useful for scheduling aircraft landing times in order to minimize the total penalty cost. We called this combination "Improved Ant Colony Algorithm" (IACA).

The Improved Ant Colony Algorithm is illustrated as follows:



Improved Ant Colony Algorithm

Dynamic case:

This algorithm can be adapted to the problem in the dynamic case where we consider two forms of effects: the cancellation of a flight and the closing of a runway. This adaptation is in the level of the graphical representation by the addition or delete of aircraft or runway. Ant Colony Optimization is a very robust heuristic which is known by its adaptation to the problems environment, whether there are changes or not.

VII. Computational Results

In this experiment we used a 1.6 GHz Intel Pentium M processor, 1 Go RAM. First, we present results of the improvement algorithms presented in section V. These algorithms have been tested on benchmarks involving up to 50 aircraft available on line in http://people.brunel.ac.uk/~mastjjb/jeb/info.html, where we fixed the number of runway at 1 runway. Secondly, we present results of ant colony algorithm and the improved ant colony algorithm using the same benchmarks and varying the number of runways from 1 to 5 runways.

A. Improvement algorithm

The table III presents a comparison between several priority rules to reduce total penalty cost (Parallel improving). The first column represents the benchmarks, in the second column we find the number of aircraft.

We note that the lower penalty cost is given by the second (T_i) and 5th priority rule $(2*T_i / (Pa_i+Pr_i))$ which are based on the target time T_i and calculation requires at least 0.05 second of CPU time.

The second algorithm (Global improving) was run 2000 iterations. Table IV presents the total penalty cost calculated by priority rules.

As observed in the table III. The lower penalty cost is given by the priority rules based on target time (T_i).

Looking over those tables, we argue that results given by the first algorithm (parallel improving) are better than the second one in terms of CPU time and penalty cost. This is the reason why we choose this algorithm to incorporate into the ant colony algorithm.

TABLE III. Computational Results of the Parallel Improving

Benchmark	*N*	E_i	*CPU (s)*	T_i	*CPU (s)*	L_i	*CPU (s)*	$2 / (Pa_i+Pr_i)$	*CPU (s)*	$2*T_i / (Pa_i+Pr_i)$	*CPU (s)*	E_i / Pa_i	*CPU (s)*	L_i / Pr_i	*CPU (s)*
1	10	1280	0.0	700	0.0	3470	0.0	930	0.0	930	0.0	930	0.0	1140	0.0
2	15	1810	0.0	1550	0.0	1830	0.0	2290	0.0	1530	0.0	1530	0.0	1550	0.0
3	20	2000	0.0	1820	0.05	8680	0.0	7170	0.0	840	0.0	900	0.0	2730	0.0
4	20	5410	0.0	3620	0.0	15140	0.0	4420	0.0	3620	0.0	3010	0.0	4960	0.0
5	20	9810	0.0	6730	0.0	10370	0.0	4390	0.0	3040	0.0	3220	0.0	3160	0.0
6	30	67525	0.0	67525	0.0	67525	0.0	116972	0.0	46270	0.0	46270	0.0	43670	0.0
7	44	39058	0.0	39058	0.0	39058	0.05	79946	0.05	49528	0.0	42738	0.0	50793	0.05
8	50	19055	0.0	2600	0.0	113580	0.0	271650	0.0	91185	0.0	103650	0.0	147940	0.0

TABLE IV. Computational Results of the Global Improving

Benchmark	*N*	E_i	*CPU (s)*	T_i	*CPU (s)*	L_i	*CPU (s)*	$2 / (Pa_i+Pr_i)$	*CPU (s)*	$2*T_i / (Pa_i+Pr_i)$	*CPU (s)*	E_i / Pa_i	*CPU (s)*	L_i / Pr_i	*CPU (s)*
1	10	1280	0.07	700	0.06	3470	0.06	880	0.06	880	0.04	880	0.06	1090	0.08
2	15	1800	0.08	1500	0.08	1820	0.02	2330	0.05	1480	0.08	1480	0.07	1500	0.08
3	20	4960	0.1	4840	0.07	8560	0.03	5120	0.05	1070	0.08	1160	0.07	2590	0.08
4	20	5000	0.13	2780	0.05	6630	0.03	4740	0.02	2610	0.08	2860	0.07	4640	0.08
5	20	7690	0.08	6970	0.02	8390	0.12	6220	0.03	5050	0.03	5060	0.07	5320	0.07
6	30	56816	0.21	56816	0.08	56816	0.04	85462	0.03	44293	0.04	44296	0.04	44279	0.15
7	44	51386	0.16	51386	0.10	51386	0.10	51386	0.05	37374	0.03	45807	0.07	51147	0.07
8	50	25355	0.12	3150	0.06	42795	0.04	182310	0.05	103490	0.04	106330	0.08	136805	0.16

B. Ant colony algorithm

Table V shows values for the linear objective obtained by applying the Ant Colony Algorithm and the Improved Ant colony algorithm (Parallel improving mixed with the Ant colony algorithm). The 4th column "Optimal" presents the optimal values that we obtained by applying the software ILOG Opl Studio. The column "ACA" presents the results obtained by applying the ant colony algorithm without improving solution, the column IACA presents the results obtained by the improved ant colony algorithm.

The parameters which have given the best solutions are as follows:

	N	*R*	*Optimal*	*ACA*	*CPU (s)*	*IACA*	*CPU (s)*
1	10	1	700	1150	0.49	**700**	0.00
		2	90	120	0.49	**90**	0.05
		3	0	0	0.43	**0**	0.0
2	15	1	1480	1840	0.93	**1480**	5.87
		2	210	210	0.82	**210**	0.6
		3	0	0	0.71	**0**	0.4
3	20	1	820	2540	1.42	**820**	8.35
		2	60	60	1.31	**60**	11.26
		3	0	0	1.20	**0**	9.28
4	20	1	2520	4820	1.42	**2520**	13.73
		2	640	680	1.26	**640**	7.91
		3	130	130	1.20	**130**	6.42
		4	0	0	1.09	**0**	5.82
5	20	1	3100	6260	1.42	**3100**	8.07
		2	650	1210	1.31	730	2.36
		3	170	330	1.2	**170**	7.03
		4	0	0	1.09	**0**	3.24
6	30	1	24442	64001	3.18	**24442**	8.08
		2	554	2394	2.74	837	3.75
		3	0	240	2.58	**0**	12.19
7	44	1	1550	53342	6.64	**1550**	90.22
		2	0	160	5.54	**0**	55.16
8	50	1	1950	13840	8.02	2185	98.48
		2	135	835	10.01	165	168.14
		3	0	195	9.72	15	108.24

Number of iterations	α	β_1	β_2	ρ	improvement algorithm
1000	1	3	5	0.7	Parallel improvement

TABLE V Shows a Remarkable Improvement in Results After the incorporation of the local search heuristic (improvement algorithm) with the ant colony algorithm. Bold values are those coincident with optimal solutions. They represent 80% of the total number of tests, whereas before the incorporation of the improvement algorithm the percentage of optimal cost does not exceed 32%.

The following figure illustrates the improvement of the hybrid algorithm over the ant colony algorithm.

However, the incorporation of the local search heuristic in the ant colony algorithm greatly increases the execution time compared to the ant colony algorithm, because additional calculations made at the time of assignment the landing time of each aircraft.

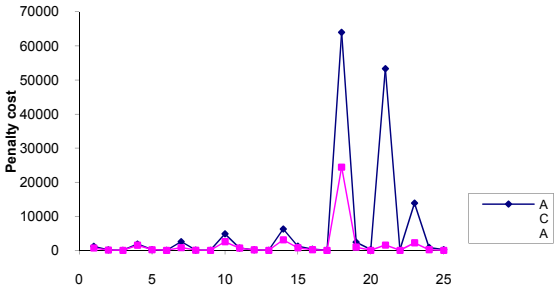


Fig. 2. Comparison between colony algorithm and improved ant colony algorithm

The following figure illustrates the optimal values and the values obtained by the improved ant colony algorithm (IACA).

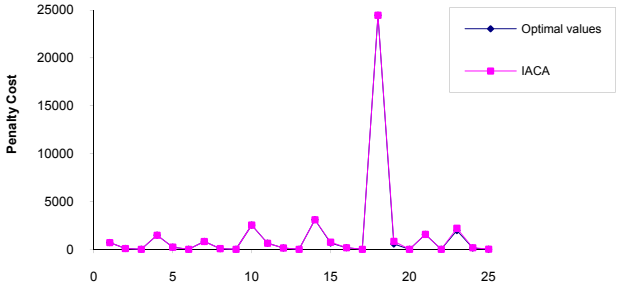


Fig. 3. Optimal values and obtained IACA values

We can observe that the graph representing the optimal values is almost identical to the graph representing the solution obtained by our algorithm. In 80% of cases, the optimal solution was achieved by our hybrid algorithm.

Table VI presents the computational results by applying the Improved Ant Colony Algorithm for the nonlinear objective where there is to maximize the advance made by aircraft.

The 3rd column "Max" presents the maximal values given in [25].

TABLE VI. COMPUTATIONAL RESULTS FOR THE NONLINEAR OBJECTIVE

	N	*R*	*Max*	*IACA*	*CPU (s)*
1	10	1	4849	4849	0.82
		2	5924	5924	0.49
		3	6185	6185	0.44
		4	6237	6237	0.33
2	15	1	18337	17688	1.76
		2	19948	19915	0.93
		3	20078	20078	0.82
3	20	1	35632	27740	4.78
		2	38524	38524	1.59
		3	38664	38664	1.32
4	20	1	20001	13732	5.22
		2	22888	22682	2.31
		3	23659	23659	1.92
		4	23955	23955	1.65
		5	24140	24140	1.48
5	20	1	19381	15322	3.75
		2	26021	25926	2.03
		3	26495	26495	1.59
		4	26699	26666	1.78
		5	26732	26732	1.58
6	30	1	-2847013	-15552753	7.96
		2	-8943	-130388	5.88
		3	0	-3785	4.84
7	44	1	-23266	-	-

		2	644749	429731	17.28
		3	646432	636507	10.58
8	50	1	728837	442716	44.21
		2	797116	727488	16.06
		3	799417	776492	16.52

For the instance of 44 aircraft and one runway, our algorithm could not give a feasible solution.

The following figure illustrates the values given by table VI.

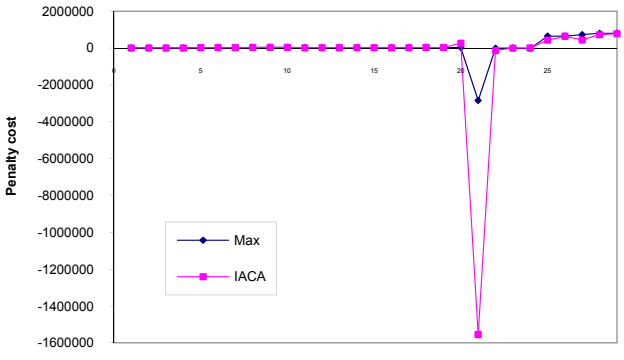


Fig. 4. Maximal solution value compared to obtained values given by IACA

We observe that solutions given by our algorithm are comparable to maximal values given by [25]. They coincide with the maximum values known in 48% of the total number of instances. We note that the deviation from the maximal solution does not exceed 10% for 32% solutions, the deviation of 20% of solutions do not exceed 40%, 8% and finally the solutions are far from the maximum solutions.

VIII. Conclusion

In this paper, we proposed, first, a new heuristic for scheduling aircraft landing times on a single runway from an order determined by a priority rule. We compared several priority rules for test our heuristic. Secondly, we adapted the ant colony algorithm to our problem in the multiple runway case, we have combined with the new heuristic in the assignment of aircraft landing times level for each runway.

Our algorithm has been tested on instances available online http://people.brunel.ac.uk/ mastjjb ~ / jeb / info.html, involving 10 to 50 aircraft and 1 to 5 runways. We compared our results with optimal solutions obtained by ILOG Cplex. In the linear objective case, our algorithm provides solutions that coincide with the optimal solutions in 80% of the total number of instances, with an average deviation of 5% from the optimal solutions for 20% of instances that remain. Regarding the nonlinear objective, optimal solutions are unknown, we compared our solutions with those given in [25]. In 48% of the tests we made, the solutions provided by IACA coincide with the maximum values known.

References

[1] J. Abela, D. Abramson, M. Krishnamoorthy, A.De Silva and G. Mills, "Computing optimal schedules for landing aircraft", Proceeding 12th National ASOR Conference, Adelaide, Australia, 71 – 90, 1993.

[2] K. Artiouchine, P. Baptiste, C. Dürr, "Runway sequencing with holding patterns", European Journal of Operational Research, 189 (3), pages 1254-1266, 2008.

[3] N. Bauerle, O. Engelhardt-Funk, and M. Kolonko, "On the waiting time of arriving aircratfs and the capacity of aiports with one or two runways", European Journal of Operational Research, 177 (2), 1180-1196, 2007.

[4] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha and D. Abramson, "Scheduling aircraft landings – The static case", Transportation Science, 34, 180 – 197, 2000.

[5] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha and D. Abramson, "Displacement problem and dynamically Scheduling aircraft landing", Journal of the Operational Research Society 55, 54-64, 2004

[6] J. E. Beasley, J. Sonander, and P. Havelok, "Scheduling aircraft landing at London Heathrow using a population heuristic", Journal of the operational Research Society, 52, 483 – 493, 2001.

[7] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem", Advanced Engineering Informatics 18, 41-48, 2008

[8] J. Boukachour and A. Elhilali Alaoui, "A Genetic Algorithm to solve a Problem of Scheduling Plane Landing", Advanced Computer Systems part II, 257 - 263, 2002.

[9] V. Ciesielski and P. Scerri, "Real Time Genetic Scheduling of Aircraft Landing Times". In D.Fogel, editor, Proceeding of the 1998 IEEE International Conference on Evolutionary Computation (ICEC98), 360 – 364. IEEE, New York, USA, 1998.

[10] D. Costa and A. Hertz, "Ants can colour graphs", Journal of the Operational Research Society 48, 295-305, 1997.

[11] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant system for job shop scheduling", Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL), 34, 39-53, 1994.

[12] J. Dréo, A. Pétrowski, P. Siarry and E. Taillard : "Métaheuristiques pour l'optimisation difficile". Eryolles 2003.

[13] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, L. M. Gambardella, "Time dependent vehicle routing problem with a multi ant colony system", European Journal of Operational Research, 185 (3), 1174-1191, 2008.

[14] M. Dorigo : " Optimization, Learning and Natural Algorithms". PhD thesis, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1992.

[15] M. Dorigo and L. M. Gambardella, "Ant colonies for the traveling salesman problem", BioSystems 43, 73-81, 1997.

[16] M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactions on Evolutionary Computation 1, 53-66, 1997.

[17] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man, and Cybernetics–Part B, 26 (1), 1-13, 1996.

[18] N. Durand : "Algorithmes génétiques et autres outils d'optimisation appliqués à la gestion du trafic aérien", 2004.

[19] A.T. Ernst and M. Krishnamoorthy, "Algorithms for Scheduling Aircraft Landing", CSIRO Mathematical and Information Sciences Private Bag 10, Clayton South MDC, Clayton VIC 3169, Australia. 2001.

[20] A.T. Ernst M. Krishnamoorthy and R.H. Store, "Heuristic and exact algorithms for scheduling aircraft landings", Networks 34; 229-241, 1999.

[21] J. Heinonen, F. Pettersson, "Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem", Applied Mathematics and Computation 187 (2), 989-998, 2007.

[22] G. Jung and M. Laguna, "Time segmenting heuristic for an aircraft landing problem", leeds school of Bisness, Univrsity of Colorado, Boulder, CO 80309, USA. working paper, 2003.

[23] M. Kong, P. Tian,Y. Kao, "A new ant colony optimization algorithm for the multidimensional knapsack problem", Computers & Operations Research, 35 (8), 2672-2683, 2008.

[24] C. J. Liao, H. C. Juan, "An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups", Computers & Operations Research 34, 1899-1909, 2007.

[25] H. Pinol and J. E. Beasley, "Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem", Europeen Journal of Operational Research, 127 (2), 439-462, 2006.

[26] M. Reimann, K. Doerner, R. F. Hartl, "D-Ants: Savings Based Ants divide and conquer the vehicle routing problem », Computers & Operations Research 31, 563–591, 2004.

[27] M.J. Soomer and G.J. Franx, "Scheduling aircraft landings using airlines' preferences", European Journal of Operational Research, vol. 190 (1), 277-291, 2008.

[28] M.J. Soomer and G. Koole, "Fairness in the Aircraft Landing Problem", submitted paper, VU University, De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands, 2008.

Ghizlane Bencheikh is an Assistant Professor at the Department of Economics, Faculty of Law, Economic and Social Sciences, Meknes,

Morocco. She is a member of the Laboratory Modeling and Scientific Computing of the Faculty of Sciences and Techniques of Fez, Morocco and the Laboratory CERENE of University of Le Havre, France. She works on scheduling problems and metaheuristics.

Jaouad Boukachour is an Associate Professor at the Department of Computer Sciences, Le Havre Institute of Technology, France. He received his PhD in computer science from the University of Rouen (France) in 1992 and Accreditation to Supervise Research from the University of Le Havre in 2006. His primary research interests are in scheduling problems, hard optimization, supply chain and Logistics Information System. He has more than 30 referred research papers and has supervised a number of PhD researchers in areas such as logistics and scheduling aircraft landings. Currently, he is supervising four PhD students working on traceability, risk management, supply chain performance and optimization and presently acts as scientific director of various research projects.

Ahmed El Hilali Alaoui is a PhD of Operational Research at the Faculty of Sciences and Techniques of Fez, Morocco. His research interests include: Scheduling Problems and Operational research. His is responsible for the operational research and computer science group, and he is supervising 10 PhD students working on job shop scheduling, scheduling aircraft landing, vehicle routing and optimization algorithms. He is a member of the La Société Marocaine de Recherche Opérationnelle (SOMARO).