Journal of King Saud University – Computer and Information Sciences (2016) **28**, 98–109

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

A memetic algorithm to solve the dynamic multiple runway aircraft landing problem

Ghizlane Bencheikh [a,*], **Jaouad Boukachour** [b], **Ahmed El Hilali Alaoui** [c]

[a] *Faculty of Law, Economics and Social Sciences, B.P. 3102, Toulal, Meknes, Morocco*
[b] *University of Le Havre, 5 rue Boris Vian, 76610 Le Havre Cedex, France*
[c] *Faculty of Science and Technology, B.P. 2202, Route d'Imouzzer, Fes, Morocco*

Received 13 September 2014; revised 16 June 2015; accepted 8 September 2015
Available online 2 November 2015

KEYWORDS

Dynamic aircraft landing problem;
Ant colony optimization;
Local search;
Metaheuristics

Abstract The aircraft landing problem (ALP) consists of scheduling the landing of aircrafts onto the available runways in an airport by assigning to each aircraft a landing time and a specific runway while respecting different operational constraints. This is a complex task for the air traffic controller, especially when the flow of aircrafts entering the radar range is continuous and the number of aircrafts is unknown a priori. In this paper, we study the dynamic version of the ALP when new aircrafts appear over time, which means that the landing of the previous aircrafts should be rescheduled. To solve this problem, we propose a memetic algorithm combining an ant colony algorithm and a local heuristic.

1. Introduction

When an aircraft (i) enters into the radar horizon of an airport (at an appearing time a_i), it becomes a responsibility of the air traffic controller to determine for that aircraft an appropriate path, a runway and a landing time (t_i). This landing time has to be as close as possible from a preferred landing time of the aircraft, called target landing time (ta_i) which corresponds to the time that the aircraft could land at if it flies at its cruise speed which is the most economical speed of the aircraft and it corresponds to the time announced to passengers. The landing time (t_i) must belong to a time window, bounded by an earliest landing time (e_i) and a latest landing time (l_i). The earliest landing time corresponds to the time at which the aircraft could land if it flies at its fastest speed (which is not economical for aircraft) while the latest landing time depends on its autonomy of carburant. In addition, there are aerodynamic considerations that arise because of the turbulence created by landing aircraft. These considerations impose a separation time between the landings of two aircrafts. This minimum separation time depends on the aircraft types and may vary for different pairs of aircrafts. Any deviation of the scheduled landing time from the target landing time causes disturbances in the airport. To better control it, a penalty cost is associated with any earliness (er_i) or tardiness (tr_i) from the target landing time of aircraft i.

* Corresponding author.
E-mail addresses: ghizlane_bencheikh@yahoo.fr (G. Bencheikh), jaouad.boukachour@univ-lehavre.fr (J. Boukachour), ahmed.elhilali@fst-usmba.ac.ma (A. El Hilali Alaoui).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2015.09.002

In the dynamic version of the aircraft landing problem, new aircrafts appear over time. This means that, over time, landed aircraft are removed and newly appearing aircraft are added and rescheduling is periodically performed. Aircrafts being scheduled to land sufficiently close to the current time are prevented from being rescheduled; this is expressed by an interval called "freezing time" (fz): when the scheduled aircraft landing time lies within this time window, it is frozen, and cannot be rescheduled.

2. Previous work

The aircraft landing problem has interested a large number of authors, but few papers deal with the dynamic case of the problem contrary to those treating the static case. Ernst et al. (1999) propose an exact algorithm to calculate the aircrafts' landing times using a simplex method. This method is based on a partial ordering between aircrafts, which allows fixing the sequencing variables. This algorithm was also used in two other approaches proposed in this paper. The first one is the branch and bound algorithm, they used the simplex to calculate the lower bounds. And the second method is based on a problem space search heuristic. These methods were tested on benchmarks involving up to 50 aircrafts and 4 runways. Authors also indicate that these methods can be used in the dynamic version of the problem.

Beasley et al. (2004), present a mathematical formulation to minimize the displacement of the problem, they developed three solution approaches; first, they divide the problem into sub-problems by grouping airplanes according to their appearing times, each sub problem is solved optimally using Cplex. Two metaheuristics based on populations of solutions are used to solve the problem. Computational results are presented involving up to 500 aircrafts and 5 runways. To our knowledge, no comparison has been made with these results.

Moser and Hendtlass (2007) treat the dynamic case of the problem with a single runway, they applied the Extremal optimization hybridized with a local search heuristic to schedule the aircraft landing and computational results are presented for 10–50 aircrafts.

Among papers which study the ALP in the static case, we cite the work of Abela et al. (1993) who presented a formulation of the problem as a mixed linear program and solve it optimally by a branch and bound algorithm, they present another approach by applying a genetic algorithm; a comparison of the two methods is presented. Ciesielski and Scerri (1998) applied a genetic algorithm for the ALP in a two runways case. Beasley et al. (2000) presented a mixed linear program formulation of the ALP in the static case; they solved it by a method based on the relaxation of the binary variables in addition to some constraints. Computational results are presented involving up to 50 aircrafts and 4 runways. A particular case has been presented by Beasley et al. (2001), they developed a heuristic based on a heuristic population for improving the aircraft landing at Heathrow airport. Ernst and Krishnamoorthy (2001) presented two solution methods, a branch and bound method and a genetic algorithm. In the single runway case, Boukachour and El Hilali Alaoui (2002) presented a genetic algorithm. Pinol and Beasley (2006) presented a mathematical formulation of the ALP with linear and non linear objective functions; they presented two genetic approaches: Scatter Search and bionomic algorithm, computational results are presented involving up to 500 aircrafts and 5 runways. Another application of genetic algorithm is proposed by Bencheikh et al. (2013), in their work, they also applied a Tabu search algorithm which they hybridize with GA. Computational results are presented involving up to 500 aircrafts and 5 runways. Baüerle et al. (2007) were interested in reducing the waiting time of one or two runways; they presented a model for the landing procedure of aircrafts. Artiouchine et al. (2008) 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 in order 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. Soomer and Franx (2008) 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 incurred by the airline companies. This cost is related to the arrival delays of the flight. Soomer and Koole (2008) considered an additional objective function which represents the airlines' preferences. The airlines company provides different cost functions for each individual flight which has different characterizations in addition to its landing time. Randall (2002) solved the ALP by using the ant colony optimization. We also applied an ant colony optimization (ACO) combined with a local search (Bencheikh et al., 2011), this algorithm was tested on benchmarks from OR Library (Beasley, OR-Library) involving 50 aircrafts and 5 runways.

In this paper, we propose to solve the aircraft landing problem in the dynamic and a multiple runway case. We adapt the ant colony algorithm (ACA) used in Bencheikh et al. (2011) to the dynamic case where we have to deal with two situations: the first one is the apparition of new aircrafts in the range radar, so rescheduling must be performed and the second one is freezing time, any time of the program, aircrafts entering in the freezing time must be deleted from the list.

In the section 3, we present a mathematical formulation of the ALP in the dynamic case, this formulation is the mathematical formulation proposed by Beasley et al. (2000). We present in Section 4 the ACA applied to the dynamic aircraft landing problem (DALP). In Section 5, we propose a memetic algorithm combining the ACA and a local search algorithm used to improve solutions given by the colony. Computational results are presented and discussed in Section 6. Conclusion and some further works are given in Section 7.

3. Mathematical formulation

In this section, we present the mathematical formulation of the aircraft landing problem in the dynamic case as proposed by Beasley et al. (2000).

We use the following notations:

- N: Number of aircrafts
- R: Number of runways
- a_i: appearing time of aircraft i
- t_i: landing time of aircraft i
- ta_i: target landing time of aircraft i
- e_i: earliest landing time of aircraft i
- l_i: latest landing time of aircraft i

- S_{ij}: separation time between the landings of aircrafts i and j if they land on the same runway
- s_{ij}: separation time between the landings of aircrafts i and j if they land on different runways
- Pb_i: penalty cost by one unit of time if aircraft i lands before its target landing time
- Pa_i: penalty cost by one unit of time if aircraft i lands after its target landing time
- $X_{ij} = \begin{cases} 1 & \text{if aircraft } i \text{ lands before } j \\ 0 & \text{otherwise} \end{cases}$
- $y_{ir} = \begin{cases} 1 & \text{if aircraft } i \text{ lands on runway } r \\ 0 & \text{otherwise} \end{cases}$
- $z_{ij} = \begin{cases} 1 & \text{if aircraft } i \text{ and } j \text{ land on the same runway} \\ 0 & \text{otherwise} \end{cases}$

3.1. Objective function

In this formulation, the objective is to minimize the total penalty cost and the displacement function which corresponds to the total move of the current solution from the previous one. This program is defined as:

$$Min \sum_{i=0}^{N} D_i(T,t) + \left(\sum_{i=1}^{N} \max(0, ta_i - t_i).Pb_i + \max(0, t_i - ta_i).Pa_i \right) \quad (1)$$

The minimization of the penalty cost of deviation between the actually landing time of all aircrafts and their target landing times is expressed by the term:

$$\sum_{i=1}^{N} \max(0, ta_i - t_i).Pb_i + \max(0, t_i - ta_i).Pa_i$$

In Beasley et al. (2000), a new function called displacement function is presented. This function is defined by

$$D(T,t) = \sum_{i=1}^{N} D_i(T,t)$$

Since the problem studied is dynamic, it can happen that the value of the landing time assigned to one or more aircraft is modified during the scheduling. So, they defined $D_i(T, t)$ to quantify the effect of the modification of each decision variable i from its previous (known) solution value T_i to its new (currently unknown) value t_i.

It can be expressed by

$$D_i(T,t) = (T_i - t_i)^2$$

3.2. Constraints

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

$$e_i \leqslant t_i \leqslant l_i \quad \forall i = 1, \ldots, N \quad (2)$$

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

$$x_{ij} + x_{ji} = 1 \quad \forall i,j = 1, \ldots, N; \quad j > i \quad (3)$$

In some cases, we can immediately decide if $x_{ij} = 1$ or $x_{ij} = 0$. For example, if $l_i < e_j$ then $x_{ij} = 1$ and $x_{ji} = 0$.

The separation constraints must be respected:

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

where: M is a great positive number.

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 (4) becomes:

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

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

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

which is always true.

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

$$\sum_{r=1}^{R} y_{ir} = 1 \quad \forall i = 1, \ldots, N \quad (5)$$

It is natural to see that the matrix (z_{ij}) is symmetric

$$Z_{ij} = z_{ji} \quad \forall i,j = 1, \ldots, N; \quad j > i \quad (6)$$

Constraints (7) and (8) link the variables y_{ir}, y_{jr}, and z_{ij}:

$$z_{ij} \geqslant y_{ir} + y_{jr} - 1 \quad \forall i,\ j = 1, \ldots, N; \quad j > i; \quad \forall r = 1, \ldots, R \quad (7)$$

$$z_{ij} \leqslant 1 - y_{ir} + y_{jr} \quad \forall i,j = 1, \ldots, N, r = 1, \ldots, R \quad (8)$$

Indeed,

- If aircraft i lands on runway r (i.e., $y_{ir} = 1$) and aircraft j lands on the same runway r (i.e., $y_{jr} = 1$), then the variable $z_{ij} = 1$. This is guarantee by the constraints

  $1 \leqslant z_{ij} \leqslant 1$

- If aircraft i lands on the runway r ($y_{ir} = 1$), but j does not ($y_{jr} = 0$), then $z_{ij} = 0$

  $0 \leqslant z_{ij} \leqslant 0$

- If aircrafts i and j did not land on the runway r, so we can't fix the value of z_{ij}, they can be landed on the same runway or not:

$$-1 \leqslant z_{ij} \leqslant 1$$

A concrete formulation of the problem is presented in ANNEXE.

4. ACO applied to the DALP

The ACO (ACO) is a metaheuristic approach introduced by Marco Dorigo in 1992 (Dorigo, 1992) to solve combinatorial optimization problems. Based on the behavior of real ants while searching food, the ACO consists of a population of artificial ants that iteratively construct solutions to a given instance of a combinatorial optimization problem and use pheromone

trails to communicate. Each ant of the colony starts its solution from null and, from iteration to another, adds new component according to the problem information and the pheromone trail. When all solutions are constructed, each ant updates the trail of pheromone according to its solution quality.

The ACO has been initially applied to the traveling salesman problem (TSP) to find the shortest hamiltonian path in a complete graph (Dorigo and Gambardella, 1997a,b). After that, ACO has been applied to both static and dynamic combinatory problems including machine scheduling (Liao and Juan, 2007), job shop scheduling (Colorni et al., 1994), vehicle routing (Donati et al., 2008; Bell and McMullen, 2008; Reimann et al., 2004), graph coloring (Costa and Hertz, 1997), knapsack problem (Kong et al., 2008), etc. and it represents a powerful tool to solve dynamic problems because of its adaptation to environment changes.

Real ants are dynamic by nature, they can find the shortest path to the food sources even when obstacles appear and disappear constantly.

In this paper, we propose an adaptation of ACA to the dynamic aircraft landing problem in the multiple runway case. It's a modified version of the algorithm applied to the static case of the problem (Bencheikh et al., 2011), here we have to deal with the appearing of new aircrafts over time while the solution process has already started. We also have to take into consideration the freezing time, when the landing time of an aircraft is "so close" from the current time, it is frozen and it could not be rescheduled.

4.1. Graphical representation

To solve the aircraft landing problem using an ACA, we need to present it as a graph. The graphical representation used in this paper is based on a bi-level graph. In the first level, we place the available runways and in the second level, we place the aircrafts. We add two dummies nodes D and F corresponding respectively to the input and the output of graph (Fig. 1). To construct its solution, an ant starts its trajectory from node D, in the first step, it selects a runway from those that are available. Then comes the selection of an aircraft from those which have appeared in the radar control and the assignment of a landing time for it. Finally, the ant arrives at the end of the graph. Before inserting the following aircraft, the ant has to check the apparition of new aircrafts in the range horizon and the freezing of the aircraft's landing time. This process is repeated until there is no aircraft available to land.

4.2. Solution construction

We define a global candidate list containing the indices of all aircrafts. Before actually beginning the construction of the solution, an ant has to create its own local candidate list which corresponds to the available aircrafts already appeared in the range radar. It starts its path from the dummy node D; first, it selects the runway where the next aircraft will land, this choice depends on the charge of the runway, or on the runway that will be free the sooner. After the choice of the runway, the ant has to choose the next aircraft to land on this runway from its local candidate list; this choice depends essentially on the priority of an aircraft compared to the other aircrafts and the memory of the ant colony. This process is repeated until there is no aircraft available to land. To respect the freezing time constraint, at the end of the iteration, we have to fix the landing times of aircrafts which have entered the freezing window. When the landing time of an aircraft is fixed (frozen), it is definitively removed from the global candidate list.

Before presenting the detailed ACA applied to the dynamic aircraft landing problem, we present the original algorithm which was applied to the static case of the problem (Bencheikh et al., 2011). Each step is described in the following subsections.

4.2.1. ACA applied to the ALP in the static case

1. Initialize the candidate list by the indexes of all aircrafts
2. Initialize the matrix of pheromone trails
3. For each ant k
   i. Repeat
      – Select a runway r according to Eq. (10)
      – Select an aircraft according to Eq. (11)
      – Insert the aircraft j in the list of aircrafts affected to the runway r and delete it from the candidate list
      – Affect a landing time to the aircraft j
      – Return to node D

      While the candidate list is not empty
4. Update the pheromone trail according to Eq. (12)
5. Steps 3 and 4 a number of iterations.

4.2.2. Representation of an ant

To solve a combinatorial problem using ant colony optimization, we have to define the representation of an ant. An ant represents one solution of the problem, in our case, it is represented by:

– A candidate list according to the ant constructs its solution
– R lists representing each a runway: it contains both the indexes of aircrafts affected to and their landing times
– Penalty cost of the solution represented.

Example of an ant:

Runway 1	1:125	5:201	4:56	–
Runway 2	2:108	3:184	6:300	8:655
Runway 3	7:54	10:407	9:520	–

The solution represented by this ant means that aircrafts 1, 5 and 4 land on the runway 1 at respectively 125, 201 and 356. Aircrafts 2, 3, 6 and 8 land on the runway 2 at respectively 108, 184, 300 and 655. Finally, aircrafts 7, 10 and 8 land on the runway 3 at respectively 54, 407 and 520.

4.2.3. Initialization

In the static case of the problem, all aircrafts are known in the beginning of the scheduling process, so for each ant, we initialize the candidate list by all aircraft indexes and the runways by NULL.

In ant colony optimization, ants use pheromone trail to communicate, this is represented by a matrix initialized in the beginning by a τ_0 which is a parameter of the algorithm.

4.2.4. Runway selection

For ant k, the probability rule to select a runway r, from node D is expressed by the following equation:

$$P_{Dr}^{k} = \begin{cases} \underset{r=1,\ldots R}{\operatorname{argmin}}(\text{number of aircrafts affected to the runway } r) & \text{if } q \prec q_0 \\ r_0 & \text{otherwise} \end{cases} \tag{9}$$

where:

- $0 < q_0 < 1$ is a constant of the algorithm
- q is a value taken randomly in the interval $[0, 1]$
- r_0 is an index chosen randomly in $\{1, \ldots, R\}$.

Another probability rule is used to select the runway according to the soonest time t at which the runway will be available to receive new aircrafts. It is expressed in this paper by the following equation:

$$P_{Dr}^{k} = \begin{cases} \underset{r=1,\ldots,R}{\operatorname{argmin}}\left(\underset{j \in Candidate_k}{\min}(t_{i0}^{r} + S_{i_0 J})\right) & \text{if } q \prec q_0 \\ r_0 & \text{otherwise} \end{cases} \tag{10}$$

where:

- $Candidate_k$ is the local candidate list created by ant k
- i_0 the last aircraft affected to the runway r
- $t_{i_0}^{r}$ is the landing time of the last aircraft affected to the runway r for the ant k.

4.2.5. Aircraft selection

After choosing a runway r, the ant has to select an aircraft to land on this runway. This choice depends on three parameters:

- The most important is the priority of the aircraft (Priority (i)): we have tested several priority rules such as the appearing time (a_i), earliest landing time (e_i), target time (ta_i), latest landing time (l_i), inverse of the sum of penalties cost before and after target landing time ($1/(Pb_i + Pa_i)$), the quotient of the earliest landing time and Penalty before (e_i/Pb_i), (l_i/Pa_i) and ($ta_i/Pb_i + Pa_i$).
- The second parameter is the penalty cost (penalty cost_ (i)) which will be generated by the aircraft if it is affected to the runway r, this cost is calculated after we assign a landing time to the aircraft using one of the following expressions:

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

O is a set of aircrafts which have previously been assigned a landing time.

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

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

The choice of the assignment function for the aircraft's landing time depends on the objective function, if the objective is to minimize the total penalty cost, we use the first one in order to have landing times as close as possible from the target times. If the objective is to land aircrafts as close as possible from their earliest landing time, we use the second expression.

A weighting of these two parameters (Priority(i) and penalty_cost(i)) corresponds to the heuristic information.

$$\eta_{rj} = (1/(\text{Priority}(j) + 1))^{\beta_1}.(1/(\text{penalty\_cost}(j) + 1))^{\beta_2}$$

β_1 and β_2 are coefficients determining respectively the importance of Priority(i) and penalty_cost(i)

Note that we add the term + 1 to avoid the zero case.

- The third parameter that influences an ant's choice is the colony memory (i.e., pheromone trails noted τ_{ij}).

To summarize, the probability rule to choose an aircraft is expressed by:

$$p_{rj}^{k}(t) = \begin{cases} \frac{(\tau_{rj})^{\alpha}.(\eta_{rj})^{\beta}}{\sum_{l}(\tau_{rl})^{\alpha}.(\eta_{rl})^{\beta}} & \text{if } j \in Candidate_k \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

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

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

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{12}$$

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:

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

Q is an updating constant, C is the penalty cost of the best solution in iteration t and Best solution is related to the path with the smallest penalty cost.

4.3. ACA applied to the ALP in the dynamic case

The ant colony algorithm is a powerful tool for solving combinatorial problems including dynamic problems. Its robustness is due to its adaptation to the change which can affect the problem environment. Indeed, the ant colony algorithm is a constructive metaheuristic when an ant builds a solution, it begins with the null and a step to another, it adds new components to the solution until it is complete. To add new components, the ant requires only the available information, so if new data are added, it can be inserted without changing the previous components of the solution. Likewise, if data are removed, then the ants of next iterations, will not take it into consideration. In this paper, the proposed algorithm supports two dynamic events which are the appearance of new aircrafts and the closure of a runway. Other dynamic events can be supported indirectly as the distress of an aircraft. It can be handled by a simple modification in its data.

4.3.1. Appearing of new aircrafts

In the dynamic case of the aircraft landing problem, the number of aircrafts are unknown in advance. So, new aircrafts can appear during the scheduling process. This is controlled by the appearing time of each aircraft. So, when an ant constructs its solution and a new aircraft arrives, it does not affect the assignment of the previous ones. Each ant creates its own local candidate list containing only aircrafts which appear in the radar, if new aircraft arrives, it is automatically inserted in the local candidate list.

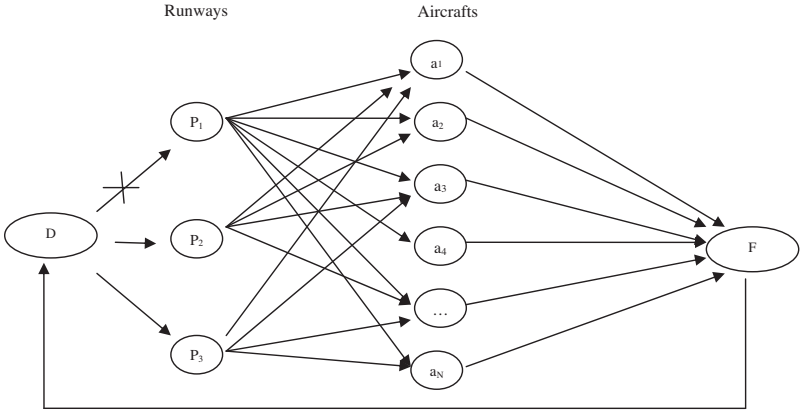
The way in which the ant updates its local candidate list is detailed in Section 4.3.4.

Another new parameter characterizing the dynamic aspect of the problem is the freezing time. If the scheduled landing time of an aircraft is too close from the current time, it has to be frozen, and can't be rescheduled in the next iteration of the algorithm. In this case, the frozen aircraft is definitely removed from the candidate list, which means that ants of successive iterations won't reschedule its landing time.

The way in which the ant freezes aircrafts is detailed in Section 4.3.5.

4.3.2. The closure of a runway

In the case of the closure of a runway, we remove the edge linking the beginning of the graph and the closed runway. So, when the ant begins its path, this runway won't be accessible for it.



4.3.3. ACA applied to the DALP

The ACA applied to the dynamic case of the problem is presented as follows:

1. Initialize the current time Tc
2. Initialize the global candidate list by the indices of all aircrafts
3. Initialize the matrix of pheromone trails
4. For each ant k
   i. Create a local candidate list containing the aircrafts whose appearing times exceed the current time Tc
   ii. Repeat
      – Select a runway r according to Eq. (10)
      – Select an aircraft according to Eq. (11)
      – Insert the aircraft j in the list of aircrafts affected to the runway r and delete it from the local candidate list
      – Affect a landing time to the aircraft j
      – Return to node D

   While the candidate list is not empty
5. Fixation of the landing time of aircrafts entering the freezing time
6. Update the pheromone trail according to Eq. (12)
7. Steps 4 and 5 a number of iterations.

4.3.4. Creation of the local candidate list

The local candidate list is formed by the available aircrafts waiting to land. Each ant has its own local candidate that is created according to the appearing times of the aircrafts. If the appearing time of an aircraft i is lower than the current time of scheduling (Tc), this aircraft is automatically inserted in the local candidate list.

Consider appearing times of a set of 10 aircrafts presented in the following table:

Aircrafts	1	2	3	4	5	6	7	8	9	10
Appearing time (a_i)	54	120	14	21	35	45	49	51	60	85

– In the beginning of the scheduling process, i.e., current time $Tc = 0$, no aircraft appeared in the radar,
– When $Tc = 14$, aircraft 3 appears, so it is inserted in the local candidate list,
– When $Tc = 50$, for example, the local candidate list will contain aircrafts 3–7.

So, at each iteration, the ants update their local candidate lists to insert new aircrafts, i.e. aircrafts whose appearing time exceeds the current time.

4.3.5. Aircraft landing fixation

When the scheduled landing time of an aircraft enters into the freezing window, it has to be fixed and removed from the global candidate list, to make sure that the next ants won't select it.

Let's consider that the freezing time is $fz = 10$. If we suppose that, in the current time $Tc = 80$ the scheduled landing times for aircrafts 1, 3–8 are as follows

Aircrafts	1	3	4	5	6	7	8	9
Appearing time (a_i)	54	14	21	35	45	49	51	60
Scheduled landing time (t_i)	130	89	100	130	160	210	145	250

We note that the first aircraft which is going to land is aircraft 3 at time 89. Since current time is $Tc = 80$ and the freezing time is 10, so $Tc - t_i < fz$. In this case, aircraft 3 is frozen and it is definitively removed from the candidate list.

5. Memetic algorithm

To improve the results obtained using the ACA, we propose a local search heuristic which adjusts the scheduled landing time for aircrafts in order to minimize the total cost of penalties. We called this combination "Improved Ant Colony Algorithm" (IACA).

We can summarize the Improved ACA as follows:

5.1. ACA adapted to the dynamic aircraft landing problem

1. Initialize the current time Tc
2. Initialize the global candidate list by the indices of all aircrafts
3. Initialize the matrix of pheromone trails
4. For each ant k
   i. Create a local candidate list consisting of the aircrafts the appearing times of which exceed the current time Tc
   ii. Repeat
   - Select a runway r according to Eq. (10)
   - Select an aircraft according to Eq. (11)
   - Insert the aircraft j in the list of aircrafts affected to the runway r and delete it from the local candidate list
   - Affect a landing time to the aircraft j
   - Adjust the landing time
   - Return to node D

   While the candidate list is not empty
5. Fixation of the landing time of aircrafts entering the freezing time
6. Update the pheromone trail according to Eq. (12)
7. Steps 4 and 5 a number of iterations.

5.1.1. Adjusting the aircrafts landing time

The adjusting landing time aims to reduce the total cost of penalty caused by all aircrafts. Based on the landing times assigned to aircrafts, 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) aircrafts to keep the respect of intervals of security. We have to check that the new landing time is not outside of the landing window; in this case we cancel the increase (resp. reduce) and keep the last feasible solution.

5.1.2. A use case of the adjustment of the landing times

Let 10 aircrafts be expected to land at an airport. The landing windows, appearing times, target landing times, the penalties and the security intervals are presented in Tables 1 and 2.

To simplify the procedure, we consider that there is one runway.

1. According to the appearing times, aircrafts will appear in the following order:

Aircrafts (i)	3	4	5	6	7	8	1	9	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.:

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

Aircraft 3 is assigned its target time which is 98:

Aircrafts (i)	3	4	5	6	7	8	1	9	10	2
Landing times (t_i)	98									

Table 1 Landing windows, appearing, target landing time and the penalty cost.

Aircrafts	1	2	3	4	5	6	7	8	9	10
Appearing time (a_i)	54	120	14	21	35	45	49	51	60	85
Earliest landing time (e_i)	129	195	89	96	110	120	124	126	135	160
Target landing time (ta_i)	155	258	98	106	123	135	138	140	150	180
Latest landing time (l_i)	559	744	510	521	555	576	577	573	591	657
Penalty before target time (Pb_i)	10	10	30	30	30	30	30	30	30	30
Penalty after target time (Pa_i)	10	10	30	30	30	30	30	30	30	30

Table 2 Separation time between aircrafts (S_{ij}).

	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

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)

i	3	4	5	6	7	8	1	9	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

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

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

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

For the aircraft 7: the calculated landing time is: 143 which 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:

i	3	4	5	6	7	8	1	9	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 was 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.

Thus, following the same approach, we complete the assignment of the landing adjusted times to all the aircrafts.

6. Computational results

The memetic algorithm was implemented in C++, and executed on a computer Intel(R) Core (TM) Duo 2,40 GHz 2,40 GHz, 4,00 Go RAM and tested on benchmarks involving up to 50 aircrafts and 1–4 runways available on line in http://people.brunel.ac.uk/~mastjjb/jeb/orlib/airlandinfo.html (Beasley et al., 2000).

The best solutions are obtained with the following settings for parameters (see Table 3):

Table 4 summarizes the results obtained for 8 benchmark instances, where the number of aircrafts varies from 10 to 50 aircrafts and the number of runways varies from 1 to 4 runways. Columns Z_{opt} and Z_{stat} represent the penalty cost of respectively optimal solutions found by the software OPL Studio and solutions found by the static version of the memetic algorithm presented by Bencheikh et al., (2011) applied both to the aircraft landing problem in the static case where all aircrafts are known in advance and there is no freezing time. $Z_{DALP\text{-}OPT}$, $Z_{DALP\text{-}H1}$ and Z_{FCFS} represent the solution values found by respectively DALP – OPT, DALP – H1 and FCFS presented by Beasley et al. (2004); the DALP – OPT approach consists of regrouping aircrafts according to their appearing times and solving each sub problem optimally using Cplex, the DALP – H1 approach described by Beasley et al. (2004) and the FCFS consists of landing aircrafts according to the First Come First Served procedure. The Z_{dyn} column presents the solution found by our memetic algorithm applied to the dynamic case of the problem. Finally, the last column shows the CPU time expressed in seconds.

We first compare the optimal solutions calculated using Opl-studio and those found by the memetic algorithm for the static case of the problem. These results are presented by Bencheikh et al. (2011). In this paper, authors also present a comparison between results of the ACA before and after the incorporation of the local algorithm.

We observe that in 80% of the total number of tests optimal values coincide with values given by the memetic algorithm (Bold values in the table).

In the dynamic case of the ALP, note that for the instance 7 of 44 aircrafts and 1 runway, our algorithm could not give a feasible solution. For the instance of 10 aircrafts and 1 runway, the solution value given by our algorithm is better than those given by DALP – OPT, DALP – H1 and FCFS.

Figs. 2–4 present the results given by the memetic algorithm and those given by FCFS First Come First Served procedure presented by Beasley et al. (2004) with respectively 1–3 runways.

We can see that in the worse case, the memetic algorithm values coincide with those calculated by the First come first served procedure.

Table 3 Parameters of the memetic algorithm.

α	β_1	β_2	ρ
1	2	4	0.3

Table 4 Computational results.

Benchmark	N	R	Z_{opt}	Z_{stat}	$Z_{DALP\text{-}OPT}$	$Z_{DALP\text{-}H1}$	Z_{FCFS}	Z_{dyn}	CPU (s)
1	10	1	700	700	740	740	1790	**710**	233.00
		2	90	90	90	120	120	**120**	237.87
		3	0	0	0	0	0	**0**	233.00
2	15	1	1480	1480	1730	1870	2610	2030	319.01
		2	210	210	210	210	210	**210**	317.14
		3	0	0	0	0	0	**0**	317.20
3	20	1	820	820	940	1440	2930	2450	384.00
		2	60	60	60	60	60	**60**	384.02
		3	0	0	0	0	0	**0**	384.28
4	20	1	2520	2520	2700	2670	6290	6540	307.00
		2	640	640	680	680	1560	1040	307.02
		3	130	130	130	130	330	220	307.04
		4	0	0	0	0	60	**0**	307.14
5	20	1	3100	3100	3810	6130	8370	7210	333.01
		2	650	730	680	1070	1440	720	333.09
		3	170	170	240	240	240	390	333.06
		4	0	0	0	0	0	**0**	333.21
6	30	1	24,442	24,442	24,442	24,442	24,442	**24,442**	465.67
		2	554	837	809	882	882	882	466.08
		3	0	0	0	0	0	**0**	466.09
7	44	1	1550	1550	3974	3974	1550	–	–
		2	0	0	0	0	0	**0**	776.88
8	50	1	1950	2185	2000	2915	26,835	4600	688.02
		2	135	165	135	255	10,140	3755	688.01
		3	0	15	0	0	4825	1635	688.01

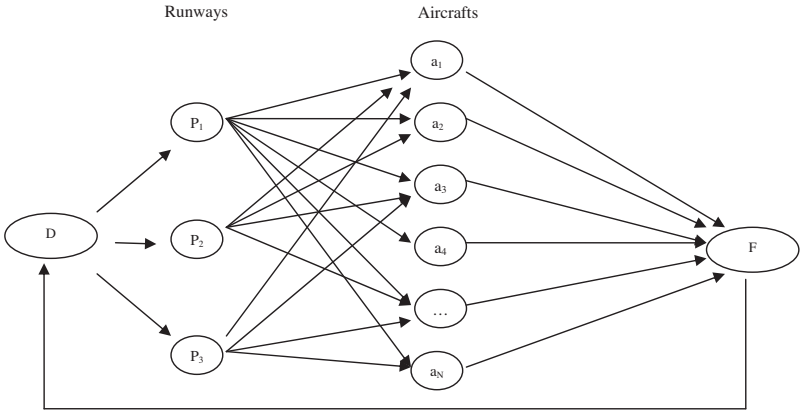


Figure 1 Graphical representation of the problem.

Table 5 presents the deviation of the solution values of the memetic algorithm from solution values of DALP – OPT, DALP – H1 and FCFS. This deviation is calculated as follows:

$$Dev_1 = 100\frac{Z_{dyn} - Z_{DALP-OPT}}{Z_{DALP-OPT}}; \quad Dev_2 = 100\frac{Z_{dyn} - Z_{DALP-H1}}{Z_{DALP-H1}};$$
$$Dev_3 = 100\frac{Z_{dyn} - Z_{FCFS}}{Z_{FCFS}}$$

We observe that for all instances, except instance 5, of 20 aircraft and 3 runways, the deviation of the memetic algorithm from FCFS algorithm is zero or a negative value. This means that in the worst case, the aircrafts are landed according to a FCFS procedure.

In comparison with the DALP H1 approach, the memetic algorithm provides 2 solutions where the penalty cost is lower; these are instances 1 and 5 of respectively 20 aircrafts and 2 runways, and 10 aircrafts and 1 runway. For 50% of the instances, the solution values are the same. And for the rest of instances, deviation value is varying from 9% to 145% and 1373% for the instance 8 of 50 aircrafts and 2 runways.

In comparison with the DALP OPT approach, 45% of the tests are the same (Bold values in Table 2). For the rest of instances, deviation value is varying from 9% to 161% and 2681% for the instance 8 of 50 aircrafts and 2 runways.

7. Conclusion

In this paper, we consider the aircraft landing problem (ALP) where we have to organize the landing of a set of planes in an airport that has one or more runways. This problem has been classified NP-complete. To solve the problem, we have to

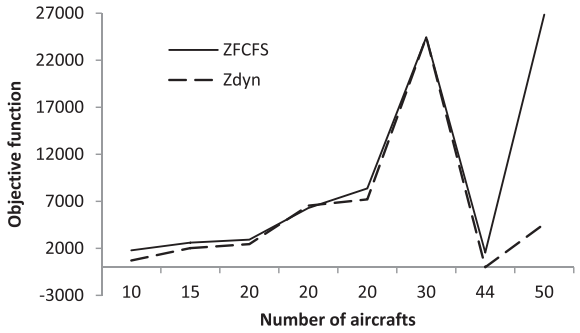


Figure 2 Comparison between FCFS and memetic algorithm, 1 runway.

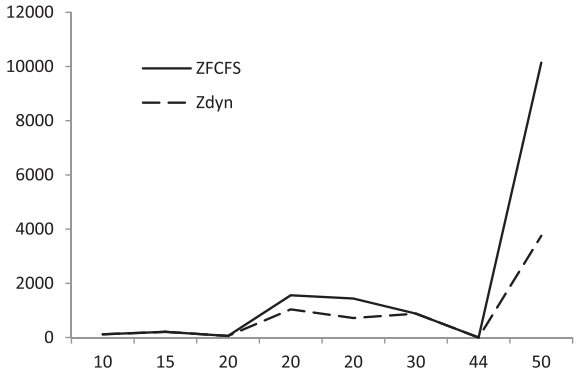


Figure 3 Comparison between FCFS and memetic algorithm, 2 runways.

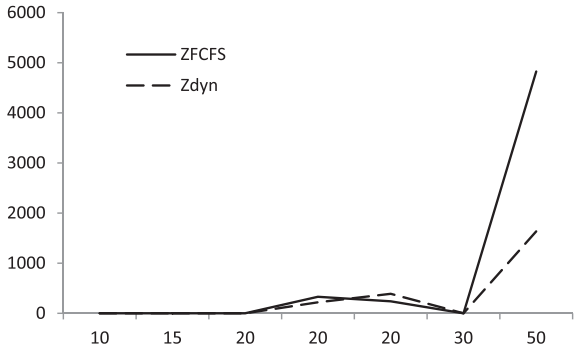


Figure 4 Comparison between FCFS and memetic algorithm, 3 runways.

(1) Assign a landing time to all aircrafts
(2) Affect aircrafts to the runways.

Taking into consideration certain precedence and safety constraints there are two cases of the problem, static and dynamic, in the static case of the problem, all the data are known in advance, which is not the case in the second case of the problem, dynamic. In this case, the number of aircraft and their characteristics are unknown, over time, new aircrafts may arise during planning, which makes the problem more complicated to solve.

Table 5 Deviation of the memetic's solutions from DALP – OPT, DALP – H1 and FCFS.

Benchmark	Dev_1 ($Z_{DALP\text{-}OPT}$) (%)	Dev_2 ($Z_{DALP\text{-}H1}$) (%)	Dev_3 (Z_{FCFS}) (%)
1	−4	−4	−60
	33	0	0
	0	0	0
2	17	9	−22
	0	0	0
	0	0	0
3	161	70	−16
	0	0	0
	0	0	0
4	142	145	4
	53	53	−33
	69	69	−33
	0	0	0
5	89	18	−14
	6	−33	−50
	63	63	63
	0	0	0
6	0	0	0
	9	0	0
	0	0	0
7	–	–	–
	0	0	0
8	130	58	−83
	2681	1373	−63
	Not defined	Not defined	−66

In this paper, we consider the aircraft landing problem in the dynamic case. We presented in the first section, a mathematical formulation of the problem as a linear integer program.

To solve the problem, we applied a memetic algorithm combining ant colony algorithm and a local improvement heuristic. The choice of the ACO algorithm resides in its constructive aspect. Indeed, when an ant builds its solution, it does not take into account information from all components, so, we can insert or delete nodes in the graph without this affecting his path.

This algorithm is an adaptation of the Improved ACO applied in the ALP in the static case (Bencheikh et al., 2011). Our algorithm has been tested on instances available online http://people.brunel.ac.uk/~mastjjb/jeb/orlib/airlandinfo.html, involving 10–50 aircraft and 1–5 runways. We compared our results with three approaches presented by Beasley et al. (2004), namely DALP – OPT, DALP – H1 and FCFS. To our knowledge, no comparison has been made with these results in the literature. However, to improve our results, we propose in further research to combine an exact method with the ant colony algorithm.

Appendix A

We present in this Annex a concrete formulation of the problem.

Data:

- $N = 4$
- $R = 1$

- $(a_i) = (54, 120, 14, 21)$
- $(ta_i) = (155, 258, 98, 106)$
- $(e_i) = (129, 195, 89, 96)$
- $(l_i) = (559, 744, 510, 521)$
-

$$(S_{ij}) = \begin{pmatrix} 0 & 3 & 15 & 15 \\ 3 & 0 & 15 & 15 \\ 15 & 15 & 0 & 8 \\ 15 & 15 & 8 & 0 \end{pmatrix}$$

- $(s_{ij}) = (0)_{i,j}$
- $(Pb_i) = (10, 10, 30, 30)$
- $(Pa_i) = (10, 10, 30, 30)$.

Variables: t_i, x_{ij}, y_{ir} and z_{ij}.

Constraints

Constraints (2): for each aircraft, its scheduled landing time must belong to the landing window, $[e_i, l_i]$

$$129 \leqslant t_1 \leqslant 559$$

$$195 \leqslant t_1 \leqslant 744$$

$$98 \leqslant t_1 \leqslant 510$$

$$106 \leqslant t_1 \leqslant 521$$

Constraints (3) there are two cases: i lands before j or j lands before i.

$$x_{12} + x_{21} = 1$$

$$x_{13} + x_{31} = 1$$

$$x_{14} + x_{41} = 1$$

$$x_{23} + x_{32} = 1$$

$$x_{24} + x_{42} = 1$$

$$x_{34} + x_{43} = 1$$

Constraints (4): The separation constraints must be respected:

$$t_1 \geqslant t_2 + 3z_{12} - M.x_{23}$$
$$t_1 \geqslant t_3 + 15z_{13} - M.x_{32}$$
$$\ldots$$
$$t_4 \geqslant t_3 + 8z_{43} - M.x_{34}$$

where: M is a great positive number.

Objective function: how to calculate the objective function

Let's suppose that the solution is $(t_i) = (150, 260, 98, 106)$

The penalty cost of deviation between the actual landing time of the aircrafts and their target landing times is calculated as follows:

For aircraft 1, its landing time is 150 and its target landing time is 155, so the penalty cost is: $10 \times (155 - 150) = 50$

For aircraft 2, its landing time is 260 and its target landing time is 258, so the penalty cost is: $10 \times (260 - 258) = 20$

For aircrafts 3 and 4, they land on their target landing time, so there is no penalty cost generated.

So the total penalty cost is: $50 + 20 = 70$.

References

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

Artiouchine, K., Baptiste, P., Dürr, C., 2008. Runway sequencing with holding patterns. Eur. J. Oper. Res. 189 (3), 1254–1266.

Baüerle, N., Engelhardt-Funk, O., Kolonko, M., 2007. On the waiting time of arriving aircrafts and the capacity of airports with one or two runways. Eur. J. Oper. Res. 177 (2), 1180–1196.

Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., Abramson, D., 2000. Scheduling aircraft landings – the static case. Transp. Sci. 34, 180–197.

Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., Abramson, D., 2004. Displacement problem and dynamically scheduling aircraft landing. J. Oper. Res. Soc. 55, 54–64.

Beasley, J.E., Sonander, J., Havelok, P., 2001. Scheduling aircraft landing at London Heathrow using a population heuristic. J. Oper. Res. Soc. 52, 483–493.

Bell, J.E., McMullen, P.R., 2008. Ant colony optimization techniques for the vehicle routing problem. Adv. Eng. Inform. 18, 41–48.

Bencheikh, G., El Khoukhi, F., Baccouche, M., Boudebous, D., Belkadi, A., Ait Ouahman, A., 2013. Hybrid algorithms for the multiple runway aircraft landing problem. Int. J. Comput. Sci. Appl. 10 (2), 53–71.

Bencheikh, G., Boukachour, J., EL Hilali Alaoui, A., 2011. Improved ant colony algorithm to solve the aircraft landing problem. Int. J. Comput. Theory Eng. 3 (2), 224–233.

Boukachour, J., Elhilali Alaoui, A., 2002. A genetic algorithm to solve a problem of scheduling plane landing. Adv. Comput. Syst. Part II, 257–263.

Ciesielski, V., Scerri, P., 1998. Real time genetic scheduling of aircraft landing times. In: Fogel, D. (Ed.), Proceeding of the 1998 IEEE International Conference on Evolutionary Computation (ICEC98). IEEE, New York, USA, pp. 360–364.

Costa, D., Hertz, A., 1997. Ants can colour graphs. J. Oper. Res. Soc. 48, 295–305.

Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M., 1994. Ant system for job shop scheduling. Belgian J. Oper. Res. Stat. Comput. Sc. (JORBEL) 34, 39–53.

Donati, A.V., Montemanni, R., Casagrande, N., Rizzoli, A.E., Gambardella, L.M., 2008. Time dependent vehicle routing problem with a multi ant colony system. Eur. J. Oper. Res. 185 (3), 1174–1191.

Dorigo, M. 1992. Optimization, learning and natural algorithms (Ph. D. thesis). Dip. Elettronica e Informazione, Politecnico di Milano, Italy.

Dorigo, M., Gambardella, L.M., 1997a. Ant colonies for the traveling salesman problem. BioSystems 43, 73–81.

Dorigo, M., Gambardella, L.M., 1997b. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. 1, 53–66.

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

Ernst, A.T., Krishnamoorthy, M., Store, R.H., 1999. Heuristic and exact algorithms for scheduling aircraft landings. Networks 34, 229–241.

Kong, M., Tian, P., Kao, Y., 2008. A new ant colony optimization algorithm for the multidimensional knapsack problem. Comput. Oper. Res. 35 (8), 2672–2683.

Liao, C.J., Juan, H.C., 2007. An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. Comput. Oper. Res. 34, 1899–1909.

Moser, I., Hendtlass, T. 2007. Solving dynamic single-runway aircraft landing problems with extremal optimisation. In: Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling, 206–211.

Pinol, H., Beasley, J.E., 2006. Scatter search and bionomic algorithms for the aircraft landing problem. Eur. J. Oper. Res. 127 (2), 439–462.

Randall, M., 2002. Scheduling aircraft landings using ant colony optimization. In: Sixth IASTED International Conference Artificial Intelligence and Soft Computing (ASC 2002), Banff, Alberta, Canada, pp. 129–133.

Reimann, M., Doerner, K., Hartl, R.F., 2004. D-Ants: savings based ants divide and conquer the vehicle routing problem. Comput. Oper. Res. 31, 563–591.

Soomer, M.J., Franx, G.J., 2008. Scheduling aircraft landings using airlines' preferences. Eur. J. Oper. Res. 190 (1), 277–291.

Soomer, M.J., Koole, G., 2008. Fairness in the aircraft landing problem. Proceedings of the Anna Valicek competition.