Hybrid method for aircraft landing scheduling based on a Job Shop formulation

G. Bencheikh †, ††, J. Boukachour ††, A. El Hilali Alaoui †, and F. El Khoukhi †, ††

† Modeling Laboratory and Scientific Computing,
Faculty of Science and Technology, Fez, B. P. 2202 Route D'Imouzzer Fez Morocco
†† CERENE
ISEL, Quai Frissard BP 1137 -76063 Le Havre Cedex France

Summary
This paper aims to study the multiple runway case of the static Aircraft Landing Problem (ALP), where all data are known in advance. In the first part of this work, we propose a formulation of the problem as a mathematical programming model in order to reduce the number of constraints (which can positively reduce the computing time) and to give a more rigorous formulation. In the second part, we formulate the ALP as a Job Shop Scheduling Problem (JSSP) based on a graphical representation. The interest of this second formulation is to show the relation between the ALP as a specific scheduling problem and the NP-hard JSSP as a more general Scheduling.
Finally, to resolve the ALP, we propose a hybrid method combining Genetic Algorithms with Ant Colony Optimization Algorithm. The hybrid algorithm is tested on variant instances of the ALP, involving up to 50 aircraft and 4 runways.

Key words:
Scheduling, Metaheuristic, Job Shop Scheduling, Aircraft landing, Mathematical Programming, Genetic Algorithm, Ant Colony Optimization Algorithm

1. Introduction and related works

The ALP has been studied in many works, including J. E. Beasley et al. [5]. They have presented a mathematical formulation of the problem as a mixed linear program; they used a resolution method based on relaxation of binary variables and introducing additional constraints. Computational results are presented involving 10 to 50 aircraft and 1 to 4 runways. G. Jung and M. Laguna have presented a new approach in [26] based on the segmentation of time. Their method has been tested on instances involving 10 to 50 aircraft and 1 to 4 runways. It consists of dividing the problem into smaller sub-problems and each one is formulated as a linear program (as in [5]) and optimally resolved. In [20], A.T. Ernst and M. Krishnamoorthy proposed resolutions methods, an exact method based on Branch and Bound and a heuristic one based on Genetic Algorithm. In [21], A.T. Ernst et al. proposed a heuristic and an exact method for the static and dynamic ALP. A Genetic Algorithm and a branch and bound algorithm has been applied by J. Abela et al. [1]. J. Boukachour and A. El Hilali Alaoui [10] applied a Genetic Algorithm for the single runway case. In [13], V. Ciesielski and P. Scerri presented a Genetic Algorithm for two runways. The landing times are allocated by specifying a 30 seconds time slot and infeasible individuals are not eliminate but penalized. A particular case has been discuss in [7], where J.E. Beasley et al. used a heuristic method based on population called 'heuristic population' for scheduling aircraft landing in London Heathrow airport. Their algorithm solves the dynamic case [6]. In [30] H. Pinol and J.E. Beasley presented a mathematical formulation of the ALP with two types of objective functions: linear and nonlinear one. They presented two genetic approaches to solve the ALP, Scatter Search and bionomic algorithm. Their computational results involve 10 to 500 aircraft and 1 to 5 runways.
In this paper, we deal with the multiple runway of the static ALP. In the first part, we give a new mathematical formulation of the ALP in order to reduce the number of constraints. In the second part, we formulate the ALP as a JSSP with a partial order and alternative sequences. Finally, we present a hybrid resolution method combining Genetic Algorithm and Ant Colony Optimization [17].

2. Problem description

When an aircraft arrives in an airport radar range (or radar horizon), it requires from air traffic control an authorization to land, a landing time and an appropriate runway if several runways are available. The landing time must be between an earliest landing time, which corresponds to the time at witch the aircraft could land if it flies at its fastest speed (witch is not economical for aircraft), and a latest landing time.
Within this time window, there is a target landing time, a preferred landing time, which corresponds to the time that

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

aircraft could land if it flies at its cruise speed (the most economical speed of the aircraft). It corresponds to the time announced to passengers. Any deviation (earliness or delay) from the target time causes disturbances at 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;
- An interval of security that must separate two successive landings on different runways must be respected;
- Each aircraft must land within a predetermined time window [Earliest landing time, Latest landing time].

3. 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 [5].

3.1 Notations

We use for the formulation, the following notations:

3.1.1. Variables

N : the number of aircrafts 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 : cost by one unit of time for aircraft i if it lands before its target time,
Pa_i : 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 $s_{ij} = s_{ji}$)

3.1.2. Decision variables

t_i : the scheduled landing time for aircraft i
er_i : earliness of the aircraft i, $\max(0, ta_i - t_i)$
tr_i : delay of the aircraft i, $\max(0, t_i - ta_i)$

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

Note that a similar variable has been used in some previous papers [5] [10] [21]:

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

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

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

3.2. Constraints

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

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

Note that this constraint is equivalent to $0 \leq y_i \leq 1$ where $t_i = e_i + y_i(l_i - e_i)$

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

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

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

Note that in [5], we found similar constraints:

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

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

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:

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

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

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

Where M is a great positive number and $x_{ij} = \begin{cases} 1 & \text{if aircraft i lands before j} \\ 0 & \text{otherwise} \end{cases}$

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 :
$$t_i \geq t_j + S_{ji} - M$$

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 $\frac{N^2 - N}{2}$ relations (expressed by $\forall i, j = 1,..., N, i < j$).

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

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

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

$$tr_i \geq t_i - ta_i \quad \forall i = 1,..., N \tag{7}$$

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

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

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

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

Then, the matrix (z_{ij}) is symmetric:

$$z_{ij} = z_{ji} \quad \forall i, j = 1,..., N, j > i \tag{11}$$

Constraint (12) links the variables y_{ir}, y_{jr}, and z_{ij}:

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

However, if one of the aircraft i or j lands on runway r and the other doesn't, we must have z_{ij} =0. This case isn't satisfied by the last constraint (12). To avoid this problem, we provide the following constraint:

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

Indeed, 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	$1 \leq z_{ij} \leq 1$

3.3. Objective function

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

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

However, there are other objectives (see [5] and [30]) like landing the aircraft as soon as possible, compute an efficient scheduling or reduce the aircraft's consummation.

4. Formulation as a Job Shop scheduling problem

The JSSP [8] [25] is one of the most complex problems encountered in real shop floor [19]. It consists generally of ordering a set of jobs to be processed in a set of machines such as:

- Each job is composed by a predefined sequence of unprompted operations ;
- Each job must be proceeded by some machines in a specified order not necessarily the same for all jobs ;
- Each machine can process only one job at a time.

The JSSP is NP-hard (see [32]) and has several variations, for example:

1. Classical Job Shop Scheduling problem
2. Flexible Job Shop Scheduling problem
3. Job Shop scheduling problem with partial order and processing alternatives
4. Job Shop scheduling problem with separable setup times
5. Job Shop problem with transportation times
6. Dynamic Job Shop problem

The JSSP has captured the interest of a great number of researchers; C. Dimopoulos and A. M. S. Zalzala in [15] illustrate recent developments in the field of evolutionary computation for manufacturing optimization, by considering the classical Job Shop scheduling and a wide range of optimization problems. J. Boukachour and A. Benabdelhafid in [9], T. Yamada and R. Nakano in [34], B.

Joo Park et al. in [11] and Dirk C. Mattfeld and Christian Bierwirth in [16], have developed resolution methods based on Genetic Algorithms. A wide survey on scheduling problems with setups times or costs are provided by A. Allahverdi et al. in [2] covering more than 300 papers. An approach entitled « adaptive Branch and Bound » is presented by Jose M. Framinan in [22] for transforming Job Shop scheduling problems into Flow Shops. In [27] T. Kis has studied an extension of the Job Shop scheduling problem namely the JSSP with partial order and processing alternatives. An application of Ant Colony Optimization on a JSSP is presented by A. Colorni et al. in [14] (see also the work of Kuo-Ling Huang and Ching-Jong Liao in [28]). Many authors have studied others variants of the problem, see M. Mastrolilli and L. M. Gambardella in [29] and A. Rossi and G. Dini in [31] for the flexible Job Shop problem, J. B. Chambers in [12] for both classical and flexible Job Shop scheduling problems, J. Hurink and S. Knust in [24], G. El Khayat et al. in [19] for the Job Shop problem with transportation times and recently in 2008, V. Vinod and R. Sridharan in [33] present a simulation-based experimental study of scheduling rules for the dynamic Job Shop problem with sequence-dependent setups.

In this section, we propose a formulation of the ALP as a JSSP.

Throughout the remainder of this section and in order to better explain our formulation, we consider ten aircraft (N=10) and two runways (R=2). Figure1 shows the aircrafts time windows and their target times.

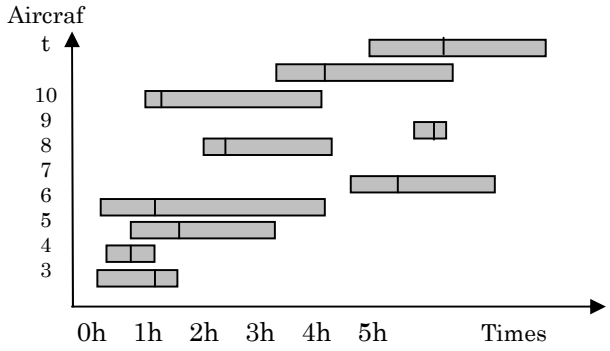
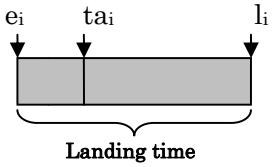


Figure 1: Aircraft time Windows and Target times

Each rectangle in the chart represent e_i (the earliest landing time for aircraft i), l_i (the latest landing time of aircraft i) and the target time ta_i.



We begin to define the Jobs and their components. First, we divide the aircrafts in subsets as shown in the figure 2 below:

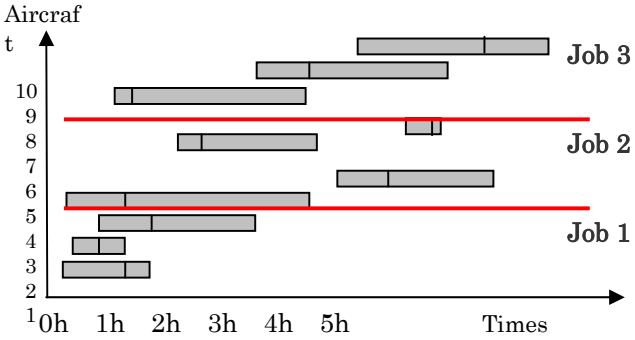


Figure 2: Random decomposing of Aircrafts in subsets

In this stage, we can build the jobs as follows:

- The first job J1 corresponds to the landing of the aircrafts: {1, 2, 3}
- The second job J2 corresponds to the landing of the aircrafts: {4, 5, 6, 7}
- The third job J3 corresponds to the landing of the aircrafts: {8, 9, 10}

We consider here the runways like machines and the landing of an aircraft like an operation (landing) of a job i (aircraft i) on a machine (runway).

Second, we define an order between operations of each job by supposing that, when the time windows of two aircrafts are separate, an order exists between them, hence the appearance of a partial order at the job operations level.

So, we have a JSSP composed by three jobs J1, J2 and J3, where J1 is composed by three operations, J2 four operations and J3 three operations.

Now, our aim is to determine the jobs sequence on the set of machines M, in our case M = {set of runways}. This task consists to assign a starting time of each operation (a landing time) and to allocate a resource (a runway) respecting the time constraints such as:

- The earliest starting time and the due date (the earliest and latest landing times).

- Setup times of machines depend on the nature of the operations (the intervals of security between aircrafts landings).
- The order of operations in jobs (the partial order as predefined previously).

4.1. Graphical Representation

To solve the JSSP, we often use a conjunctive-disjunctive graph [8] [25]. The vertices or nodes represent the operations of the job including two special nodes (fictitious nodes), representing the beginning and the end of the schedule. Conjunctive arcs represent constraints of precedence and disjunctive arcs represent pairs of operations that must be performed on the same machines. In our case, we use a disjunctive graph « and/or » based on two kinds of graphs (*for more details on « and/or » graph representation, see [27]*):

- **« and » graph** describes partial order at the level of jobs (the partial order between operations in the same job is expressed by an ‘’ and ‘’ relation).
- **« or » graph** describes the alternation relation between operations with alternatives resources (the possibility to land an aircraft on a runway among a set of candidates runways (multiple runways appearance) is expressed by an ‘’ or ‘’ relation).

In the « and/or » graph, we have two kinds of edges:

- Dotted edges connecting the operations of an « and » graph for a partial order relation ‘’ and ‘’
- Dotted edges connecting the operations of an « or » graph for an alternation relation ‘’ or ‘’.

The following figure illustrates an example of an « and/or » graph.

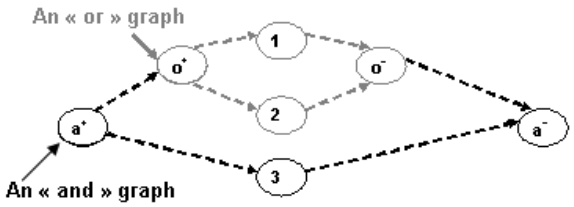


Figure 3 Example of an « and/or » graph

The following notations are adopted afterwards:

- $\mathbf{a}^+$: the dummy beginning of an « and » graph
- $\mathbf{a}^-$: the dummy end of an « and » graph
- $\mathbf{o}^+$: the dummy beginning of an « or » graph
- $\mathbf{o}^-$: the dummy end of an « or » graph
- **D** : the source node
- **F** : the sink node

We introduce the operation O_{ir} to designate the landing of aircraft i on the runway r. So, for each aircraft i corresponds an « or » graph whose branches are formed by the operations O_{ir}, for $r = 1,..., R$ (knowing that exactly one operation must be chosen during scheduling to respect the fact that an aircraft can land on a single runway through the set of runways) and for each job j corresponds an « and » graph whose branches are formed by « or » subgraphs corresponding to the operations in partial order knowing that each branch can have one « or » subgraph or more connected by precedence relations where they exist.
For the previous example with 10 aircraft and two runways, adopting the cited notations, the jobs become:

$$J1 = \{O_{11}, O_{12}, O_{21}, O_{22}, O_{31}, O_{32}\}$$
$$J2 = \{O_{41}, O_{42}, O_{51}, O_{52}, O_{61}, O_{62}, O_{71}, O_{72}\}$$
$$J3 = \{O_{81}, O_{82}, O_{91}, O_{92}, O_{101}, O_{102}\}$$

Note that the landing time windows of plane 1 and plane 2 are not separated; hence we haven’t order between operations which are associated. Likewise for aircraft 1 and 3, and aircraft 2 and 3.
We observe that the landing time windows of the pair of aircraft (4, 5), (4, 7), (6, 5) and (6, 7) are separated; hence we have an order between operations associated with each pair. The operations with corresponding pairs of aircraft (8, 10) and (9, 10) are connected by precedence relations.

The « and /or» graphs below show the processing steps of, respectively, Jobs J1, J2 and J3, as nodes.

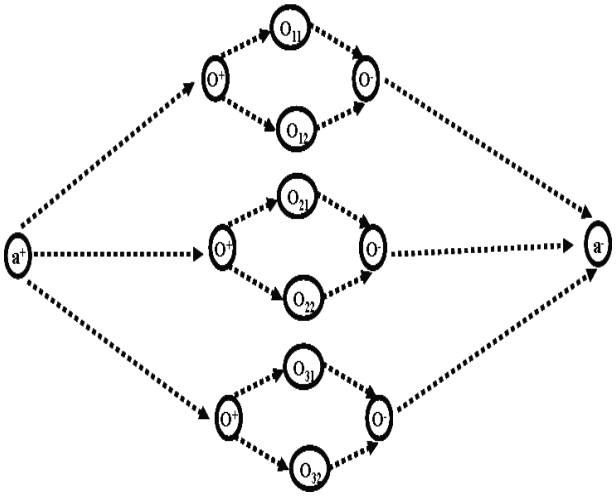


Figure 4: Job J1

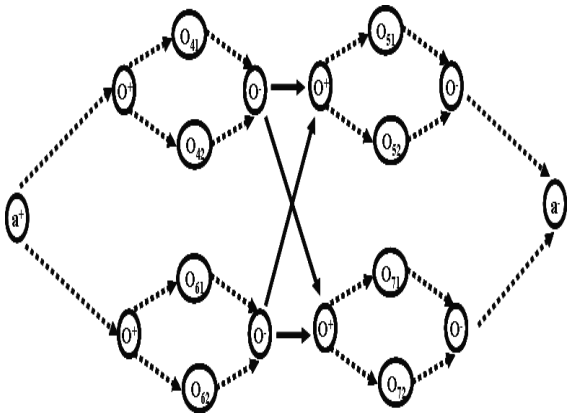


Figure 5: Job J2

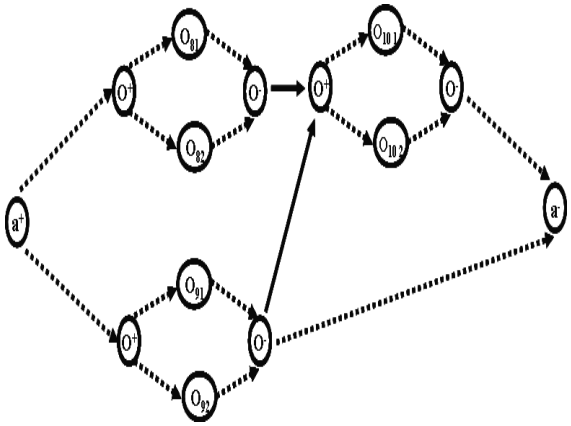


Figure 6: Job J3

Note that:

- A precedence relation between two operations is represented by a continuous conjunctive arc.
- There is no adjacency between pairs of operations (Oir, Oir').
- The other operations are all connected by disjunctive arcs that we have not represented in the graphs but we consider them implicitly.

The figure 7 below illustrates the complete « and/or » graph associate to the problem:

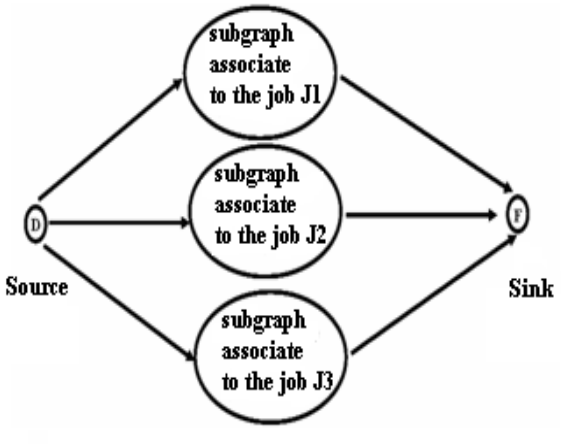


Figure 7: Complete « and/or » graph associate to the problem

We evaluate by:

- S_{ij}, the arcs (O_{ir}, O_{jr}) for $i \neq j$
- s_{ij}, the arcs (O_{ir}, O_{jk}) for $r \neq k$
- 0, the arcs :

$(O^{+}, O_{ij}), (O_{ij}, O^{-}), (a^{+}, O_{ij}), (O_{ij}, a^{-}), (D, a^{+})$

and (a^{-}, F)

Thus, we get the graph associated to a JSSP with partial order and alternative sequences.

5. Our hybrid method

To resolve the ALP, we propose a hybrid method, called ACOGA (Ant Colony Optimization Genetic Algorithm) which combines two metaheuristics: Genetic Algorithm (GA) and Ant Colony Optimization (ACO).
The ACO is used to generate an initial population of feasible solutions for the GA. The purpose is to increase the chances of generating some goods solutions in the initial population. Starting from a good initial population helps the GA to find goods solutions faster. The performance of a genetic algorithm depends strongly on the initial population.

First, we start by describing below in more detail our genetic algorithm in terms of coding method, fitness function, crossover as well as mutation.

5.1. Coding method

To resolve the ALP, there are two tasks. First, give a landing sequence and secondly compute landing times for a set of aircraft. The landing times are computed based on

the landing sequence. In addition to those lists (landing sequence and landing times), we use a third list to represent the assignment of runways to aircraft.

Here's an example of code for 10 aircraft and 3 runways

3	1	4	2	6	7	8	5	9	10
3	2	3	2	2	3	2	1	2	1
100	115	140	150	300	320	400	552	650	580

In this solution composed of three lists, aircraft 3 lands on the 3rd runway at 100 minutes from midnight, the aircraft 2 lands on the second runway at 115 and so on.

5.2. Generating the initial population

The initial solution plays a critical role in determining the quality of final solution in any local search and the initial good schedules can be evolved into the better schedules (see [11]). However, it is possible to generate the initial population randomly, but it's generally difficult to generate a number of feasible solutions with a small cost of penalties in a good time. That's why we propose to apply an ACO algorithm by using the graph previously generated (see graph of Figure 7). The number of ants is equal to the desired initial population size.

Note that to generate a population of solutions, we can suggest two strategies:
1. If one has a population of solutions, we propose a strategy that consists of fixing the number of ants to the desired initial population size and applying the ACO algorithm below.
2. In order to get a population of higher quality solutions, we propose a second strategy that consists of considering a random number of ants and fixing the maximum number of iterations of the ACO algorithm to the size of the initial population, and keep at each iteration the best solution given by the family of ants to build an initial population for the GA.

To obtain a population of solutions in a short time, we choose the second strategy which generates a landing sequence and an assignment of runways (first and second lists of a solution) by applying only one cycle of the ACO algorithm. In order to complete the solutions of the ALP, after building the previous lists, we have to compute a landing time to each aircraft (list 3).

5.2.1. Landing sequence

The ACO algorithm is described as follows:

1. Initialization
For each ant k,

- *Initialize the $Tabu_k$ list with the source D of the graph*
- *Initialize the $Candidate_k$ list with the first operation of each job J_i*
- *Initialize the pheromone trace with τ_0*

2. Construction of a population of m solutions
For each ant k = 1, …, m do
Initialize $i \leftarrow D$
While ($Candidate_k$ is not empty) do

- *From a current node i, choose the next node $j \notin Tabu_k$ among the $Candidate_k$ nodes according to a transition rule:*

$$\mathbf{j} = \begin{cases} \arg\max_{\mathbf{l} \notin \mathbf{Tabu_k}} [(\tau_{\mathbf{il}})^{\alpha} (\eta_{\mathbf{il}})^{\beta}] & \mathbf{if\ q \le q_0} \\ \mathbf{J} & \mathbf{if\ q \succ q_0} \end{cases}$$

J is chosen randomly among $Candidate_k$

$$\eta_{il} = \begin{cases} \dfrac{1}{\left(\left|ta_l - ta_i\right| * S_{il}\right) + 1} & if\ i \text{ and } l \text{ land on the same runway} \\ \dfrac{1}{\left(\left|ta_l - ta_i\right| * s_{il}\right) + 1} & if\ i \text{ and } l \text{ land on different runways} \end{cases}$$

- *Insert j in $Tabu_k$*
- *Updating the $Candidate_k$ list*
- *Local updating of pheromone trails τ_{ij} using the formula :*

$$\tau_{ij} = (1 - \rho_1)\tau_{ij} + \rho_1 \tau_0$$

End while
End for

Where, m is the number of ants, $Tabu_k$ is the list of visited nodes, $Candidate_k$ is the list of nodes candidates to be visited in order to respect precedence relations imposed by the technological sequences, α and β are two parameters that control the relative importance of τ_{ij} and η_{ij}, q is a random variable uniformly distribute on [0, 1], q_0 is a parameter set between 0 and 1, τ_0 is the initial value of pheromone trails and ρ is the coefficient of vaporization

We note that in the previous algorithm, the information heuristic (η_{il}) depends on two parameters:

- Separation time (S_{il}) or (s_{il}) : generally, $s_{il} << S_{il}$ ($\forall i,l = 1,\ldots,N$). So, the aircraft landing on different runways will be privileged than those landing on the same runway. This decision will provide a good repartition of aircraft on runways.

- $|ta_i - ta_j|$: the aircraft closed in term of target time are privileged than the others in order to land as soon as possible to their target landing time to reduce the cost penalty.

5.2.2. Computation of landing times

For each solution generated by ACO method, we compute the landing times of aircraft based on the landing sequence and the assignment of runways, respecting the aircraft landing windows and security's intervals between them.

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

5.3. Reproduction

The reproduction, crossover followed by a mutation, is applied after a roulette-wheel selection. We apply the 1X crossover [17] on the first list and second one of a solution. These lists correspond to the sequence of aircrafts and their allocated runways. The crossover point is the same for both lists and it is chosen randomly.

To avoid a repetition of an aircraft, when constructing Construction of children from a pair of parents, we proceed as follows:

Let P1 and P2 be the lists of the parents:

Crossover point

3	9	1	5	4	10	7	6	2	8	Parent1
3	2	3	2	2	3	2	1	2	1	

6	3	7	10	8	9	2	1	4	5	Parent2
1	1	2	3	1	2	3	1	3	1	

After choosing a crossover point, we copy the left part of the first parent into the first child and the left part of the second parent into the second child

Child1

3	9	1	5	4	10				
3	2	3	2	2	3				

Child2

6	3	7	10	8	9				
1	1	2	3	1	2				

Then we complete the first child with the missing aircraft and their assigned runways from the second parent (not already assigned) in the same order in which they appear in the second parent) beginning by the first element on the left side.

3	9	1	5	4	10	6	7	8	2	Child 1
3	2	3	2	2	3	1	2	1	3	

Then the other child will be similarly produced by exchanging the role of the two parents.

The third list (landing times) is calculated according to the process described in 5.2.2.

Finally, the mutation selects two or more aircraft at random and swaps their positions as shown in the example below with 10 aircraft and 3 runways. Unlike crossover, mutation acts just on the landing sequence (the first list of as solution).

Before

6	3	7	10	8	9	2	1	4	5
1	1	2	3	1	2	3	1	3	1

After

6	9	7	10	8	3	2	1	4	5
1	1	2	3	1	2	3	1	3	1

Then, the third list (landing times) is calculated according to the process described in 5.2.2.

6. Numerical Results and Discussion

The hybrid algorithm was implemented in C++, and tested on a Pentium 4, 2.66 GHz CPU with 240 Mo of RAM. All benchmarks can be downloaded from the web page: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/airlandinfo.html [4].

Table 1 summarizes the results acquired for 25 benchmark instances. First, we can see clearly from Table 1 that our hybrid algorithm works much better that a genetic algorithm alone. Secondly, in the majority of cases, our algorithm finds or approaches the optimal solution.

Benchmarks	N	R	Optimal values	ACGA Values	CPU(s) (ACGA)	GA Values	CPU(s) (GA)
1	**10**	**1**	700	700	1.42	820	12.40
		2	90	90	1.14	90	10.36
		3	0	0	1.09	0	10.32
2	**15**	**1**	1480	1720	2.44	1720	19.97
		2	210	210	2.25	220	16.82
		3	0	0	1.48	10	15.87
3	**20**	**1**	820	850	3.64	1750	28.41
		2	60	60	3.03	570	26.41
		3	0	0	3.02	320	22.32
4	**20**	**1**	2520	4480	3.51	6580	28.11
		2	640	680	3.49	1770	25.19
		3	130	130	3.45	600	24.06
		4	0	0	3.04	330	25.19
5	**20**	**1**	3100	4800	3.36	5800	27.70
		2	650	720	4.02	1650	29.15
		3	170	240	3.39	560	24.75
		4	0	0	3.06	440	22.88
6	**30**	**1**	24442	24442	3.08	*	*
		2	554	554	3.86	*	*
		3	0	0	4.38	*	*
7	**44**	**1**	1550	1550	5.67	*	*
		2	0	200	4.41	*	*
8	**50**	**1**	1950	3240	7.33	*	*
		2	135	160	10.47	*	*
		3	0	0	10.09	*	*

Table 1 Computational results

At the level of CPU time, we observe that GA takes more time than ACGA. This is due to the fact that in GA the landing times are generated randomly from the landing windows. Indeed, each time, we assign a random landing time to an aircraft while respecting the intervals of security with previous ones. If it's not the case, this landing time is regenerated until all the intervals of security are respected. This task increases the CPU time, especially when the number of aircraft increases.

7. Conclusion

This paper has discussed the multiple runway case of the static Aircraft Landing Problem, where data are known in advance. We have expressed the ALP as a JSSP with partial order and alternative sequences through an « and/or » graph. In

terms of resolution, we enhanced a hybrid method which combines two metaheuristics, namely Genetic Algorithm and Ant Colony Optimization.

Finally, our ongoing researches will concern, in one hand, the improvement of our algorithm to handle instances of large size (100 to 500 aircrafts) and on the other hand, the study of the dynamic case to face dynamic disturbances.

References

[1] J. Abela, D. Abramson, M. Krishnamoorthy, A. De Silva and G. Mills (1993), « *Computing optimal schedules for landing aircraft* », Proceeding 12th National ASOR Conference, Adelaide, Australia, pp 71-90, 1993.

[2] A. Allahverdi, C.T. Ng, T.C.E. Cheng and Mikhail Y. Kovalyov (2008), « *A survey of scheduling problems with setup times or costs* », European Journal of Operational Research, Volume 187, pp 985-1032, Issue 3, 16 June 2008.

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

[4] J. E. Beasley (1990), OR-library: Distributing test problems by electronic mail, Journal of the Operational Research Society 41 (1990) 1069–1072, Available from: <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/airlandinfo.html >.

[5] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha and D. Abramson (2000), « *Scheduling aircraft landings – The static case* », Transportation Science, 34, pp 180-197, 2000.

[6] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha and D. Abramson (2004), « *Displacement problem and dynamically Scheduling aircraft landing* », Journal of the Operational Research Society, 55, pp 54-64, 2004.

[7] J. E. Beasley, J. Sonander, and P. Havelok (2001), « *Scheduling aircraft landing at London Heathrow using a population heuristic* », Journal of the operational Research Society, 52, pp 483-493, 2001.

[8] J. Blazewicz, W. Domschke and E. Pesch (1996), « *The job shop scheduling problem: conventional and new solution techniques* », European Journal of Operational Research, 93, pp 1-33, 1996.

[9] J. Boukachour and A. Benabdelhafid (2000), « *Résolution d'un problème d'ordonnancement de type job-shop par les algorithmes génétiques* », Troisième Conférence Internationale de Mathématiques Appliquées et des Sciences de l'Ingénieur, CIMASI Octobre 2000, Casablanca, Maroc, 2000.

[10] J. Boukachour and A. Elhilali Alaoui (2002), « *A Genetic Algorithm to solve a Problem of Scheduling Plane Landing* », Advanced Computer Systems part II, pp 257-263, 2002.

[11] B. Joo Park, H. Rim Choi and H. Soo Kim (2003), « *A hybrid genetic algorithm for the job shop scheduling problems* », Computers & Industrial Engineering, Volume 45, pp 597-613, Issue 4, December 2003.

[12] J. B. Chambers (1996): «*Classical and flexible job shop scheduling by Tabu search* », Department of computer Science, University of Texas, 1996.

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

[14] A. Colorni, M. Dorigo, V. Maniezzo and M. Trubian (1994), «*Ant System for Job-Shop Scheduling* » , Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL), 34, pp 39-53, 1994.

[15] C. Dimopoulos and A. M. S. Zalzala (2000), « *Recent Developments in Evolutionary Computation for Manufacturing Optimization: Problems, Solutions, and comparisons* », IEEE Transactions on Evolutionary Computation, Volume 4, No. 2, pp 93-113, July 2000.

[16] Dirk C. Mattfeld and Christian Bierwirth (2004), « *An efficient genetic algorithm for job shop scheduling with tardiness objectives* », European Journal of Operational Research, Volume 155, pp 616-630, Issue 3, 16 June 2004.

[17] J. Dréo, A. Pétrowski, P. Siarry and E. Taillard (2003): « *Métaheuristiques pour l'optimisation difficile* », Eryolles 2003.

[18] N. Durand (2004) : « *Algorithmes génétiques et autres outils d'optimisation appliqués à la gestion du trafic aérien* », HDR de l'Institut Polytechnique de Toulouse, 2004.

[19] G. El Khayat, A. Langevin and D. Riopel (2006), « *Integrated production and material handling scheduling using mathematical programming and constraint programming* », European Journal of Operational Research, Volume 175, pp 1818-1832, Issue 3, 16 December 2006.

[20] A.T. Ernst and M. Krishnamoorthy (2001), « *Algorithms for Scheduling Aircraft Landing* », CSIRO Mathematical and Information Sciences Private Bag 10, Clayton South MDC, Clayton VIC 3169, Australia. 2001.

[21] A.T. Ernst, M. Krishnamoorthy and R.H. Store (1999), « *Heuristic and exact algorithms for scheduling aircraft landings* », pp 229-241, Networks 34; 1999.

[22] Jose M. Framinan (2007), « *An adaptive branch and bound approach for transforming job shops into flow shops* », Computers & Industrial Engineering, Volume 52, pp 1-10, Issue 1, February 2007.

[23] J. V. Hensen (2004), « *Genetic Search methods in air trafic control* », Computers and Operations Research, 31, pp 445-459, 2004.

[24] J. Hurink and S. Knust (2002), « *A tabu search algorithm for scheduling a single robot in a job-shop environment* », Discrete Applied Mathematics, Volume 119, pp 181-203, Issues 1-2, 15 June 2002.

[25] A. S. Jain and S. Meeran (1999), « *Deterministic job-shop scheduling: past, present and future* », European Journal of Operational Research, 113, pp 390-434, 1999.

[26] G. Jung and M. Laguna (2003), «*Time segmenting heuristic for an aircraft landing problem* », leeds school of Bisness, University of Colorado, Boulder, CO 80309, USA. Working paper, 2003.

[27] T. Kis (2003), « *Job-Shop scheduling with processing alternatives* », European Journal of Operational Research, Volume 151, pp 307-332, Issue 2, 1 December 2003.

[28] Kuo-Ling Huang and Ching-Jong Liao (2008), « *Ant colony optimization combined with taboo search for the job shop scheduling problem* », Computers & Operations Research, Volume 35, pp 1030-1046, Issue 4, April 2008.

[29] M. Mastrolilli and L. M. Gambardella (2000), « *Effective neighbourhood function for the flexible job shop problem* », Journal of Scheduling, 3(1), pp 3-20, 2000.

[30] H. Pinol and J. E. Beasley (2006), «*Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem*», Europeen Journal of Operational Research, 127(2), pp 439-462, 2006.
[31] A. Rossi and G. Dini (2007), « *Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimization method* », Robotics and Computer-Integrated Manufacturing, Volume 23, pp 503-516, Issue 5, October 2007.
[32] B. Roy and B. Sussmann (1964): « *Les problèmes d'ordonnancement avec contraintes disjonctives* », Note DS NO 9 Bis, SEMA, Montrouge, 1964.
[33] V. Vinod and R. Sridharan (2008), « *Scheduling a dynamic job shop production system with sequence-dependent setups: An experimental study* », Robotics and Computer-Integrated Manufacturing, Volume 24, pp 435-449, Issue 3, June 2008.
[34] T. Yamada and R. Nakano (1992), « *A genetic algorithm applicable to large scale job shop problems* », in Proc. 2end Int. Conf. PPS from Nature, Männer and Manderick, Eds. Amsterdam, The Netherlands: Elsevier Science, pp. 281-290, 1992.

Ghizlane Bencheikh is a PhD student of the Laboratory of Modeling and Scientific Calcul and CERENE laboratory, she is a member of Operational Research and Computer group at the Faculty of Sciences and Techniques of Fez, Morocco. She works on scheduling problems and metaheuristics.

Jaouad Boukachour is an Associate Professor of Computer Sciences at Le Havre University, France. His research interests include: Scheduling Problems, Operational Research, and Supply Chain Management. He has supervised a number of PhD researchers in areas such as logistics and scheduling aircraft landings. Currently, he is supervising six PhD students working on traceability, modelling road traffic, job shop scheduling, scheduling aircraft landings and vehicle routing. He has published more than 30 referred research papers. Within the French CPER 2006 (State-Region Project Contract), he was responsible for Modelling Optimisation and Simulation of physical and information flows in an industrial logistics project. Currently, he heads two projects about tracking container shipments, funded by French National Research Agency (ANR) and CPER 2008.

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

Fatima El Khoukhi is a PhD student of the Laboratory of Modeling and Scientific Calcul and CERENE laboratory, she is a member of Operational Research and Computer group at the Faculty of Sciences and Techniques of Fez, Morocco. She works on scheduling problems and metaheuristics.