International Journal of Computer Science and Applications,
©Technomathematics Research Foundation
Vol. 10, No. 2, pp. 53 – 71, 2013

HYBRID ALGORITHMS FOR THE MULTIPLE RUNWAY AIRCRAFT LANDING PROBLEM

GHIZLANE BENCHEIKH, FATIMA EL KHOUKHI

Moulay Ismail University, Meknes, Morocco
{ghizlane_bencheikh, el_khoukhi_fatima}@yahoo.fr

MOHAMED BACCOUCHE, DALILA BOUDEBOUS

University of Le Havre, Le Havre, France
baccouchemohamed@voila.fr ,dalila.boudebous@univ-lehavre.fr

ABDELHAQ BELKADI, ABDELLAH AIT OUAHMAN

Cadi Ayyad University, Marrakech Morocco
abdelhaqb.m@hotmail.com, ouahman@uca.ma

Over the past few decades, air traffic has experienced a tremendous Growth. However, as the air traffic develops, the limitation of the runway becomes the bottleneck during the airport operation and scheduling aircraft landing present a complex daily task encountered by most air traffic control towers. In this paper, we study the Multiple Runway case of the Aircraft Landing Problem (MRALP), which is a heavily studied combinatorial optimization problem. Many approaches such as linear programming, Genetic Algorithm (GA) and local search have been applied to the MRALP. This paper describes four hybrid algorithms for solving ALP problems. The algorithm uses two computational heuristic search techniques, namely, Tabu Search (TS) and GA. Despite the combinatorial complexity of ALP problems, the hybrid algorithms find best solutions very quickly and with very high frequencies.

Keywords: Aircraft Landing; Assignment; Genetic Algorithm; Tabu Search; Hybridization.

1. Introduction

"The Air Traffic Control (ATC) system is currently organized in predefined routes, crossing airspace sectors. Each sector is handled by a team of two controllers (a planning controller and a radar controller)" [Eurocontrol (2001)]. Typically, the ATC will be in charge of determining approach paths, runway allocation and landing times of several aircrafts in the radar horizon. Currently, the system of the air traffic management can encounter some limitations which prevent a confident answer to user's expectations [Eurocontrol (2004), (2001)], such as the capacity of aerial navigation, the takeoff and landing capacities of the airports, the limitation of the runway and a slow integration of new technologies.

Thereby, an optimal scheduling of airport runway operations can play an important role in improving the safety and efficiency of the airspace system. In fact, upon entering within the air traffic controllers' radar range, the control tower must compute a landing time for each aircraft in the horizon. This time must exist within a specified time window which is dependent on the type of aircraft and bounding by an earliest time and latest time, which represent respectively, the time required if the plane flies at its maximum airspeed and that if it flies at its most fuel efficient airspeed while holding for the maximum allowable time. Within this time window, there is a target landing time, a preferred landing time, which corresponds to the time that aircraft could land if it flies at its cruise speed (the most economical speed of the aircraft). In addition, there are aerodynamic considerations that arise because of the turbulence created by landing aircraft, imposing a separation time between the landings of any two aircrafts.

In the Aircraft Landing Problem each aircraft requires to assign it a landing time close as possible to their target time and also an appropriate runway if there is more than one runway, in order to achieve effective runway use, while respecting predefined time window associated to an aircraft and separation time requirements with other aircrafts.

This paper focuses on the resolution of the MRALP. Our hybrid algorithms were evaluated using the standard OR Library benchmarks [Beasley (1990)] and on instances which are very challenging for mathematical programming approaches.

The rest of the paper is organized as follows. A literature review is given in section 2. Section 3 shows a mathematical formulation of the ALP. Section 4 describes how our GA works and section 5 outlines our Hybrid Algorithms. Finally, we conclude with the computational results.

2. Related works

Scheduling aircrafts landing has interested many researchers in air traffic management. The basic aircraft scheduling problem involves finding optimal departing or arrival times of aircraft subject to operational constraints. A detailed review of published work addressing the ALP is given in [Beasley *et al.* (2000)]. Authors in this study 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 aircrafts and 1 to 4 runways.

According to Beasley "The aircraft landing problem (ALP) is ′hard′ to solve since it can be viewed as a job machine scheduling problem with release times and sequence dependent processing time. The job machine scheduling problem has been proved to be NP-hard; hence the ALP is NP-hard". This is reflected for example, in the work of [Milan (1997)], who considered the problem of assigning priorities, based upon factors such as number of passengers, cost of passenger delays and proportion of transfer passengers, for

landing to aircraft in arrival batches. Computational results were presented for one example with 30 aircrafts and 1 runway. [Artiouchine *et al.* (2008)] are interested in the complexity of the problem; they discussed several cases solvable in polynomial time and presented a compact mixed integer programming formulation in order to solve large instances of the general problem when all time windows have the same size. They proposed a general hybrid branch and cut framework.

[Baccouche (2007)] has applied hybridization between GA a local research to multiple runways case. GA and its variants have been shown to be very successful on the ALP. Thus, in this category, [Ciesielski and Scerri (1998)] presented a GA for two runways; they update the problem after each landing/appearance of an aircraft to find the best solution. Computational results were presented for two data sets involving 28 and 29 aircrafts and two runways.

[Baüerle *et al.* (2007)] presented a model for the landing procedure of aircraft to reduce the waiting time of one or two runways. The ALP is more suitably addressed by simulation techniques. [Bolender and Slater (2000)] combined queuing theory and discrete event simulation. Assuming that aircraft appeared according to a Poisson process, the analysis focused on differing ways of allocating newly appeared aircraft to a runway for landing (when multiple runways are present). Once allocated to a runway, aircraft landed in First-Come/First-Served (FCFS) order. Computational results were presented for instances involving up to 3 runways. [Carr *et al.* (2000)] modified the standard FCFS approach to allow individual airlines to express priorities with regard to the landing of their aircraft. Computational results were presented for a number of simulated scenarios involving 3 runways.

For the MRALP a mathematical formulation with a linear and non linear objective functions are considered in the paper of [Pinol and Beasley (2006)], for resolution, they apply two approaches: the scatter search and bionomic algorithm. Computational results are presented involving up to 500 aircrafts and 5 runways.

Heuristics have been successfully applied by several authors. [Beasley *et al.* (2001)] developed a population heuristic for scheduling aircraft landing at London Heathrow airport. They used this algorithm to solve the ALP in the dynamic case in [Beasley *et al.* (2004)]. [Jung and Laguna (2003)] presented a new approach based on the segmentation of time, dividing the problem into sub-problems and each one is formulated as a 0-1 mixed-integer linear program and was optimally solved and tested on instances involving 10 to 50 aircraft and 1 to 4 runways. [Soomer and Franx (2008)] studied the airport arrival problem with a single runway, they presented a local search heuristic in order to assign a landing time to each flight taking into account the cost provided by airline. This cost is related to the arrival delays of the flight. Other previous methods are used to address such problems include exact, GAs and Ant Colony Optimization (ACO). For example, in [Ernst and Krishnamoorthy (2001)] an exact method based on Branch and Bound and a heuristic one based on GA are presented. A job-shop scheduling view of the ALP is adopting in the work of [Bencheikh *et al.* (2009)] that presented a formulation of the

MRALP problem into a Job Shop problem with partial order and alternative sequences and an hybrid method combining GA and ACO to solve the ALP based on a "and/or" graph representation. The hybrid algorithm is tested on variant instances involving up to 50 aircraft and 4 runways. In [Bencheikh *et al.* (2011)] the behaviour of the ACO was improved by introducing a local algorithm, this hybrid algorithm was tested on benchmarks from OR Library [Beasley (1990)] involving 50 aircrafts and 5 runways. Another application of the ACO for solving the ALP was used in [Randall (2002)].

Recently, [Tavakkoli-Moghaddam *et al.* (2012)] propose a fuzzy programming approach and an estimator to schedule the sequence of aircraft landings for a single runway, two conflict objectives, namely minimizing the total cost of the deviation from the target times and minimizing the completion time of the landing sequence are used. For the MRALP, the paper of [Salehipour *et al.* (2013)] developed a mixed integer goal programming model and a hybrid meta-heuristic applying simulated annealing framework. The computational results show that the proposed algorithm can obtain the optimal solution for instances up to 100 aircrafts, and also comparable solutions for the problems with up to 500 aircrafts and 5 runways.

In this paper, we aim to consider the static case of the MRALP problem, where all data are known in advance. We begin with a mathematical model for the problem. Then, for the resolution, we propose four hybrid algorithms based on two computational heuristic search techniques, namely, GA and TS.

3. ALP Formulation

When an aircraft enters an airport radar range (or radar horizon), it requires from ATC (Air Traffic Control) to assign it the authorization to land, a landing time and an appropriate runway if several runways are available. A penalty cost is associated to each earliness or delay from the target time. The objective is thus to minimize the total penalty cost with respect to the following constraints:

- A security interval between two successive landings on the same runway,
- A security interval that must separate two successive landings on different runways,
- Each aircraft must land within a predetermined time window [Earliest landing time, Latest landing time].

Here is a new mathematical formulation of the static case of the MRALP. This formulation is a modified version based on the classical one presented in [Beasley *et al.* (2000)].

3.1. *Parameters and decision variables*

N : number of aircrafts waiting to land
R : number of available runways
e_i : earliest landing time of the aircraft i
l_i : latest landing time of the aircraft i

ta_i : target landing time of the aircraft i
Pb_i : cost per unit of time if aircraft i lands before its target time
Pa_i : cost per unit of time if aircraft i lands after its target time
S_{ij} : separation time between aircraft i and aircraft j ($S_{ij} > 0$, $i \neq j$) on the same runway
s_{ij} : separation time between aircraft i and aircraft j ($s_{ij} \geq 0$, $i \neq j$) on different runways
t_i : scheduled landing time of the aircraft i
er_i : earliness of the aircraft i, $er_i = \max(0, ta_i - t_i)$
tr_i : delay of the aircraft i, $tr_i = \max(0, t_i - ta_i)$

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

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

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

3.2. *Constraints*

The scheduled landing time of each aircraft i belongs to its landing window $[e_i, l_i]$

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

There following constraints show that aircraft i lands before aircraft j or vice versa.

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

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

Between two successive landings on the same runway or on different runways, a separation time must be respected.

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

M is a great positive number.

We introduce the following constraint to express the fact that each aircraft i must land on just one runway:

$$\sum_{r=1}^{R} y_{ir} = 1 \qquad \forall i = 1,\ldots,N \tag{5}$$

The next constraint expresses the fact that the matrix (z_{ij}) is symmetric:

$$z_{ij} = z_{ji} \qquad \forall i, j = 1,\ldots,N,\ j > i \tag{6}$$

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

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

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 isn't guaranteed by (7). So, we have added the following constraint:

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

3.3. *Objective function*

The objective is to minimize the total cost of penalties. For each aircraft, the penalty cost is linearly proportional to its earliness and delay from the actually landing time and the target landing time.

$$\text{Min} \left(\sum_{i=1}^{N} \text{er}_i . \text{Pb}_i + \text{tr}_i . \text{Pa}_i \right) \qquad (9)$$

4. Genetic Algorithm

GA is a global search metaheuristic [Holland (1992)] [Dréo *et al.* (2003)] which aims to evolve a population of solutions through a process of selection, crossover and mutation during a number of iterations. The algorithm involves the following steps [Goldberg (1994)].

4.1. *Solution encoding*

A solution is coded using two vectors S_1 and S_2. The first one represents the runway assignments and the second one represents the ordering of the aircrafts. The initial population will be composed of Np solutions.

S_1	1	2	2	2	3	3	1	3	1	1	2	2	1	3	3
S_2	1	5	3	13	8	9	4	7	10	12	6	2	14	15	11

Fig. 1. A solution

4.2. *Evaluation and selection*

From a sequence of landings, we compute the corresponding landing times in order to evaluate the quality of the solution. To assign the landing times, we apply the Simplex method after fixing the variables corresponding to the order between aircrafts and assigning runways to aircrafts. This order is set according to a FCFS procedure.

Two different methods for selecting mates are implemented: wheel [Davis (1991)] and tournament [Baker (1987), Goldberg (1994), Goldberg (1989)] selection. Elitism selection is used to keep the best solution (or some best solutions) and ensures they will be copied with the next generation while other solutions are selected using wheel and tournament ranking methods.

4.3. *Crossover and mutation*

We use 4 crossover techniques: Partially Mapped (PMX), Order (OX), Linear Order (LOX) and Cycle (CX) Crossover. Though these operators are originally designed for the Travelling Salesman Problem (TSP) [Oliver *et al.* (1987), Moscato (1989), Baccouche (2007)], we adapt it to the ALP Problem. Also, 4 mutation operators were implemented in this work: InSertion (ISM), Swap (EM), Modify (MM) and Centre Inverse (CIM) Mutation [Goldberg and Lingle (1985), Goldberg (1989), Fogel (1988), Baccouche (2007)]. These genetic operators are applied to a landing sequence (vector S_2 of a solution) (see Fig. 2.).

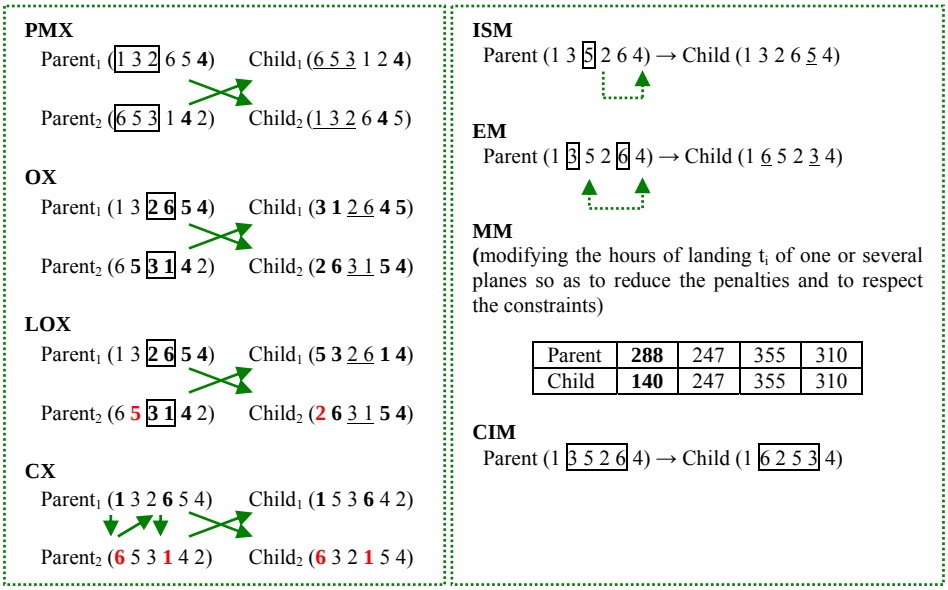

Fig. 2. Example operations of crossover (PMX, OX, LOX and CX) and mutation (ISM, EM, MM and CIM)

5. Hybrid Algorithms

To date the best results reported are not produced by a single algorithm but by a subset of algorithms. The subset contains hybrid methods which give good results for problems such as the TSP problem. However, hybridization needs to be used intelligently to avoid excessively costly calculations and degradation of results in a compromise between exploration and intensification. We propose two kinds of hybridization. In the first hybridization, TS is applied to improve some solutions obtained after the selection stage of GA. This hybrid algorithm is named “GARDE”. The second hybridization that we

propose is to apply TS to improve the final solution found by GA. We call this hybrid algorithm "AGIR"

Following the approach of [Vasconcelos *et al.* (2001)], we define a methodology which allows a choice between intensifying (local search) and diversifying (global search), based on a measure of genetic diversity σ, which is defined as:

$$\sigma = \frac{\text{Averageof the valuesof fitness}}{\text{Maximumof fitness}}$$

To formalize the choice between diversifying and intensifying research, we generate random numbers Φ_{min} and Φ_{max} between 0 and 1, defining thresholds which determine whether the search should be intensified or diversified.

We can thus synthesize the choice between intensifying or diversifying research according to:

- If $\sigma \geq \Phi_{max}$, diversification
- If $\sigma \leq \Phi_{min}$, intensification

5.1. *Tabu Search*

TS is a local search metaheuristic that has been successfully applied to solve many hard combinatorial optimization problems [Laguna *et al.* (1995)]. It is an iterative procedure that moves from one potential solution *x* to an improved one in the neighborhood of *x* noted *N(x)*. A short-term memory, usually identified as the tabu list, is used to avoid cycling by forbidding moves to solutions recently visited or sharing attributes with such solutions. More advanced memory and neighborhood structures may be advantageously used to attempt to reach higher quality solutions.

The algorithm creates at each iteration the set *N(S)* of neighbours of the current solution *S* by switching the positions of two planes. Then, the "best" neighbouring solution, not necessarily an improving one, is selected as the new current solution.

The algorithm selects the best neighbour *v* from N(S) by minimizing the term $\Delta(S_v, S) = OF(S_v) - OF(S)$. OF(S) is the objective function of S. For each permutation, we used a tabu matrix T of N lines and N columns to progressively store the number of iterations λ which will be prohibited. λ is a random value generated from the interval $[\lambda_{min}, \lambda_{max}]$, where λ_{min} and λ_{max} are fixed according to the experiments. The algorithm below describes the implementation of the TS algorithm:

Tabu Search algorithm

Generate an initial solution S
Create the Tabu matrix T
Initialize the best solution to the initial solution Bsol = S
Choose λ_{min} *and* λ_{max}

Repeat
Compute $\Delta(S_v, S)$ for each solution S_v from N(S)
Find the permutation p= (i,j) of the best solution S_v of N(S) - T // allow sometimes a taboo permutation to improve the best solution: aspiration criteria
$S=S_v$
if *$\Delta(S, Bsol)<0$,* ***Then*** *set Bsol =S*
Update T // delete the oldest entry, if necessary using FIFO rule
Forbid the permutation p during λ iterations // λ randomly generated in $[\lambda_{min}, \lambda_{max}]$
Until *(met Ng_{max} iterations) or (no improvement of the current solution)*

5.2. *Algorithm "GARDE"*

We thus thought of improving the process of selection by introducing TS before reproduction to diversify the space of solutions. We defined a tabu probability P_t in the interval [0.0, 0.2]. We fix Φ_{max} at $1 - P_t$. We apply TS if $\sigma \geq \Phi_{max}$

Algorithm GARDE

Generate an initial population P
Select the best solution Bsol of P
Repeat
Compute the value of σ
Generate P_t randomly in the interval [0.0, 0.2]
$\Phi_{max} = 1 - P_t$
Repeat
Select two solutions S et S' from P
if ($\sigma \geq \Phi max$) then $S_{T1}=TS(S)$ // apply taboo search to S
else $S_{T1}=S$
if ($\sigma \geq \Phi m$) then $S_{T2}=TS(S')$ else $S_{T2}=S'$
Crossover S_{T1} and S_{T2} to produce S_{C1}, S_{C2} with a probability P_c
Mutate S_{C1} to S_{M1} with a probability P_m
Mutate S_{C2} to S_{M2} with a probability P_m
Select two solutions among (S_{T1}, S_{T2}, S_{C1}, S_{C2}, S_{M1} and S_{M2}) to replace S and S'
Until (creation of the new population Npop)
P=Npop // new population
Select the best actual solution S_{act} of P
if (S_{act} is better than Bsol) then Bsol = S_{act}
Until (met Ng_{max} iterations) or (no improvement of the current solution)

5.3. *Algorithm "AGIR"*

We use a local exploration to improve quality of the solutions. We apply TS to avoid the prolongation of search. Thus, if the objective were not improved during the last *Ns* iterations, we apply a local TS to intensify search and ameliorate the best and the current solutions.

AGIR Algorithm

Generate an initial population P
i=0; j=0 // generation and stagnation count
Repeat
Choose the best actual solution S_{act} of P
Compute σ
Generate Φ_{min} randomly on the interval [0.0, 0.2]
Repeat
Select two solutions S et S' from P
Crossover S and S' to produce SC1 and SC2 with a probability Pc
Mutate S_{C1} to produce S_{M1} with a probability P_m
Mutate S_{C2} to produce S_{M2} with a probability P_m
Select two solutions among (S_{T1}, S_{T2}, SC1, SC2, SM1 and SM2) to replace S and S'
if (i=0) then Bsol = S_{act}
else
if S_{act} is better than Bsol then Bsol = S_{act} ; j=0
else j=j+1
if j=Ns and $\sigma \leq \Phi_{min}$ then S_{T1}=TS(S_{act}) ; S_{T2}=TS(Bsol) //apply Tabu search to S_{act} & Bsol
Select two solutions among (S_{T1}, S_{T2}, S_{act} and Bsol) to replace S_{act} and the worst solution of P
Bsol = the best solution found
Until (creation of the new population Npop)
P=Npop // replace the current population with the new population
i=i+1// next generation
Until (i=Ng_{max}) or (no improvement of the current solution)

5.4. *Hybridizing of crossover and mutation*

Another algorithm named SIMAGT was developed. This algorithm is the combination of SIMAG algorithm, with tabu crossover named CTRL and tabu mutation named MRT.

5.4.1. Hybrid crossover "CTRL"

CTRL is a crossover combined with local TS. The underlying principle of CTRL is that new cells are renewed from pre-existing cells by cellular division. In the majority of cases this renewal is carried out according in a process which includes a) the cytokinesis, which is division of the whole of the cell giving two girls cells and b) the mitosis, which is the division of the core during which the chromosomes are distributed in an identical way in the girl cells. The cells develop for a period, the interphase, ranging for two cellular divisions. In CTRL, the parent chromosomes break up into cells, the formed cells evolve according to TS and the children are finally produced by a combination of the cells.

The following example illustrates how the CTRL algorithm works. Given a number of aircrafts, a number of runways, earliest landing times, target landing times, the penalties cost and the matrix of security separation intervals:

N = 15 aircrafts R = 3 runways

Landing earlier (e_i) = |129 190 84 89 100 107 109 109 115 134 266 251 160 152 276|

A target time (ta_i) = |155 250 93 98 111 120 121 120 128 151 341 313 181 171 342|

Landing later (l_i) = |559 732 501 509 536 552 550 544 557 610 837 778 674 637 815|

The unit costs (Pa_i) and (Pr_i) for plane i :

$Pa_i = Pr_i$ = |10 10 30 30 30 30 30 30 30 30 10 10 30 30 10|

S_{ij} =

0	3	15	15	15	15	15	15	15	15	3	3	15	15	3
3	0	15	15	15	15	15	15	15	15	3	3	15	15	3
15	15	0	8	8	8	8	8	8	8	3	3	15	15	3
15	15	8	0	8	8	8	8	8	8	15	15	8	8	15
15	15	8	8	0	8	8	8	8	8	15	15	8	8	15
15	15	8	8	8	0	8	8	8	8	15	15	8	8	15
15	15	8	8	8	8	0	8	8	8	15	15	8	8	15
15	15	8	8	8	8	8	0	8	8	15	15	8	8	15
15	15	8	8	8	8	8	8	0	8	15	15	8	8	15
15	15	8	8	8	8	8	8	8	0	15	15	8	8	15
3	3	15	15	15	15	15	15	15	15	0	3	15	15	3
3	3	15	15	15	15	15	15	15	15	3	0	15	15	3
15	15	8	8	8	8	8	8	8	8	15	15	0	8	15
15	15	8	8	8	8	8	8	8	8	15	15	8	0	15
3	3	15	15	15	15	15	15	15	15	3	3	15	25	0

Step 1: Given a solution S.

S_1	1	2	2	2	3	3	1	3	1	1	2	2	1	3	3
S_2	1	5	3	13	8	9	4	7	10	12	6	2	14	15	11

S is evaluated using simplex method:

R							
1	N	1	4	10	12	14	**N_1=5**
	X_i	155	170	178	193	208	
2	N	5	3	13	6	2	**N_2=5**
	X_i	111	119	127	135	256	
3	N	8	9	7	15	11	**N_3=5**
	X_i	120	128	136	342	345	
R = 3	**The cost of solution =5280+1770+570=7620**						

Step 2: We divide the solution into 3 cells which represent the whole of planes by runways: C_i (i=1,…, R).

C_1					**C_2**					**C_3**				
1	4	10	12	14	5	3	13	6	2	8	9	7	15	11

Step 3: cytokinesis: the cells divide and give rise to 6 identical girls cells Cf (analogy with cellular division)

Cf_1					**Cf_2**					**Cf_3**				
1	4	10	12	14	5	3	13	6	2	8	9	7	15	11
Cf_4					**Cf_5**					**Cf_6**				
1	4	10	12	14	5	3	13	6	2	8	9	7	15	11

Step 4: interphase: The cells Cf_4, Cf_5 and Cf_6 evolve with TS to visit the neighbors' cells. The solutions before and after TS are:

Cf_1					**Cf_2**					**Cf_3**				
Initial solution					Initial solution					Initial solution				
1	4	10	12	14	5	3	13	6	2	8	9	7	15	11
155	170	178	193	208	111	119	127	135	256	120	128	136	342	345
5280					1770					570				
Cf_4					**Cf_5**					**Cf_6**				
the final solution					the final solution					the final solution				
1	4	10	12	14	5	3	13	6	2	8	9	7	15	11
98	136	151	171	313	93	111	120	181	250	120	128	136	341	344
190					0					470				

Step 5: By crossing the cells Cf_1, Cf_2, Cf_3, Cf_4, Cf_5 and Cf_6, we obtain eight solutions (S_0 initial =solution, S_7 = solution formed by TS, S_1 with S_6=solutions obtained by combining the cells)

R	1	2	2	2	3	3	1	3	1	1	2	2	1	3	3	Costs
S_0	**Cf_1**					**Cf_2**					**Cf_3**					7620
	1	4	10	12	14	5	3	13	6	2	8	9	7	15	11	
S_1	**Cf_1**					**Cf_2**					**Cf_6**					7520
	1	4	10	12	14	5	3	13	6	2	8	9	7	11	15	
S_2	**Cf_1**					**Cf_5**					**Cf_3**					5850
	1	4	10	12	14	3	5	6	13	2	8	9	7	15	11	
S_3	**Cf_1**					**Cf_5**					**Cf_6**					5750
	1	4	10	12	14	3	5	6	13	2	8	9	7	11	15	
S_4	**Cf_4**					**Cf_2**					**Cf_3**					2530
	4	1	10	14	12	5	3	13	6	2	8	9	7	15	11	
S_5	**Cf_4**					**Cf_2**					**Cf_6**					2430
	4	1	10	14	12	5	3	13	6	2	8	9	7	11	15	
S_6	**Cf_4**					**Cf_5**					**Cf_3**					760
	4	1	10	14	12	3	5	6	13	2	8	9	7	15	11	
S_7	**Cf_4**					**Cf_5**					**Cf_6**					660
	4	1	10	14	12	3	5	6	13	2	8	9	7	11	15	

The final step is to choose one or more solutions and introduce them into the population to replace the current worst one.

5.4.2. Hybrid mutation "MRT"

CTRL heuristic is mixed with the hybrid mutation "MRT". MRT is also based on TS. We describe the MRT mutation of the solution S7 below:

Step 1: permute two or several planes by changing their respective positions. In the S7 solution, we permute for examples plane 1 and 7 and plane 12 and 15:

	Cf_4					**Cf_5**					**Cf_6**				
S_7	4	**1**	10	14	**12**	3	5	6	13	2	8	9	**7**	11	**15**
S_7'	4	**7**	10	14	**15**	3	5	6	13	2	8	9	**1**	11	**12**

Step 2: The cells Cf_4, Cf_5 and Cf_6 evolve with a taboo search to visit the neighbors' cells.

Cf_1					**Cf_2**					**Cf_3**				
Initial solution					Initial solution					Initial solution				
4	7	10	14	15	3	5	6	13	2	8	9	1	11	12
98	121	151	171	342	93	111	120	181	250	120	128	155	341	344
0					0					310				
Cf_4					**Cf_5**					**Cf_6**				
the final solution					the final solution					the final solution				
4	7	10	14	15	3	5	6	13	2	8	9	1	12	11
98	121	151	171	342	93	111	120	181	250	120	128	155	313	341
0					0					0				

The solution after TS is:

S_1	1	2	2	2	3	3	1	3	1	1	2	2	1	3	3
S_2	4	7	10	14	15	3	5	6	13	2	8	9	1	12	11

This solution replaces the worst one in the population. A selected solution may be mutated directly without crossing. The final (optimal) solution after crossing and mutation is as follow:

R							
1	N	4	7	10	14	15	**N_1=5**
	X_i	98	121	151	171	342	
2	N	3	5	6	13	2	**N_2=5**
	X_i	93	111	120	181	250	
3	N	8	9	1	12	11	**N_3=5**
	X_i	120	128	155	313	341	
R = 3	**The cost of solution = 0**						

6. Experimental results

Tests were performed using benchmarks from the OR-library [Beasley *et al.* (2000)] with data varying from 10 to 500 planes and 1 to 5 runways. The benchmarks were designated as either small in size (10 to 50 planes) or large (100 to 500 planes) and to evaluate our approaches the results were compared to others that have been published [Pinol and Beasley (2006)]. The results[1] obtained are given in Tables 1, 2 and 3. They are indexed by the problem size, the benchmark number, the number of planes N, the number of runways R, the solutions respectively obtained by the solver CPLEX, by FCFS, by BA and SS algorithms [Pinol and Beasley (2006)] and by TS and by our four hybrid metaheuristics.

Several different and well known operators were tested on benchmarks of ALP. In terms of operator success, the results gave the following operator rankings from best to worst: OX and PMX for crossover, and EM and ISM for mutation in conjunction with TS. For both algorithms tested, the best combination of operators is OX and CIM and the results showed that experimental strategies improve the solution in any genetic type.

As shown below, the results also indicate that the search converges uniformly according to the importance of the crossover rate, the mutation rate, the size of population N_p and the elitism rate P_{el}.

Best solution (Bsol) : $N_p = 30$, $Ng_{max} = 100$, Crossover 1X, $P_c = 0.5$, Mutation IM, $P_m = 0.3$						
P_{el}		0.1	0.2	0.3	0.4	0.5
Objective function	N=10 aircrafts	810	860	940	1020	1040
	N=15 aircrafts	1850	1840	2140	2370	2410
	N=20 aircrafts	1230	1170	1440	1540	1610

The best solutions found for N=10, 15 and 20 correspond respectively to the elitism rates P_{el} =0.1, 0.2 and 0.2. The elitism increased the convergence of the GA, with an improvement of the best solutions. We notice that for rates (P_{el} >0.2 for N=10, $P_{el} \geq 0.3$ for N=15 and P_{el} >0.3 for N=20) that GA does not evolve and the quality of the solutions is degraded. We conclude that the elitism rates must adapt to the population size N_p and set it using the selection criterion:

Table 1 shows the effective parameters setting using by experimental design [Tagushi (1987)]. The settings selected here follow from the results presented earlier. In terms of stopping criterion, when the fitness of the best solution doesn't change from one generation to another during a fixed number of iterations, we stop the research. A fixed value of $N_S \in [20,50]$ stagnations before stopping is used according to the problem's size.

1 All tests were performed using a Pentium 4 2500 MHz computer

Table 1. Choice of parameters

	Best configuration	Variables Choice
Population size (N_p)	$N_p \in [20, 200]$	
Selection	Ranking	Wheel and ranking selection
Crossover type	*OX*	*OX* and *PMX*
Mutation type	*CIM*	*IM* et *CIM*
Crossover rate (P_{cross})	0.9	$P_{mut} \in [0.5, 0.9]$
Mutation rate (P_{mut})	0.2	$P_{mut} \in [0.1, 0.3]$
Elitism rate (P_{el})	$P_{el} = E\left(\frac{E(\frac{10}{N})}{N_p}\right)$ Where E is the (upper or lower) integer part	
Replacement Selection	Tournament selection	Tournament and ranking selection

The cost of the solution is equal to 0 if we have a sufficient number of the runways and each plane lands at its target time. We present the results of small instances the following table.

Table 2. Results of small instances

Instances	N	M	Z_{opt}	Z_{FCFS}	Z_{BA}	Z_{SS}	Z_{TSH}	Z_{SIMAG}	Z_{GARDE}	Z_{SIMAGT}	Z_{AGIR}
1	10	1	700	700	700	700	700	700	700	700	700
2	10	2	90	90	90	90	90	90	90	90	90
3	10	3	0	0	0	0	0	0	0	0	0
4	15	1	1480	1500	1480	1480	1480	1520	1480	1510	1480
5	15	2	210	230	210	210	210	210	210	210	210
6	15	3	0	0	0	0	0	0	0	0	0
7	20	1	820	1380	820	820	820	900	840	840	820
8	20	2	60	130	60	60	60	60	60	60	60
9	20	3	0	n/d	0	0	0	0	0	0	0
10	20	1	2520	2520	2520	2520	2520	2680	2620	2580	2520
11	20	2	640	640	640	640	640	690	680	680	640
12	20	3	130	130	130	130	130	130	130	130	130
13	20	4	0	0	0	0	0	0	0	0	0
14	20	1	3100	5420	3100	3100	3100	3240	3720	3180	3100
15	20	2	650	760	670	650	650	690	650	650	650
16	20	3	170	180	170	170	170	170	170	170	170
17	20	4	0	n/d	0	0	0	0	0	0	0

18	30	1	24442	24442	24442	24442	24442	24442	24442	24442	24442
19	30	2	554	882	554	554	554	600	554	600	600
20	30	3	0	0	0	0	0	0	0	0	0
21	44	1	1550	1550	1550	1550	1550	1550	1710	1648	1550
22	44	2	0	0	0	0	0	0	0	0	0
23	50	1	1950	unfeasible	2655	2965	2965	1950	2120	1950	1950
24	50	2	135	710	135	135	135	135	135	135	135
25	50	3	0	n/d	0	0	0	0	0	0	0

In order to compare the results obtained, we compute the average deviation from the optimal solution Av(Z). As we can see, AGIR algorithm (($Av(Z_{AGIR})$ =0.33) gives better results than other algorithms ($Av(Z_{TSH}, Z_{SIMAGT}, Z_{BA}, Z_{SIMAG}, Z_{GARDE}, Z_{SS}, Z_{FCFS})$ = {0.4, 0.88, 1.7, 1.82, 2.07, 2.1, 37.1}). So, AGIR algorithm is powerful for small instances. In addition, the tabu crossover CTRL and the tabu mutation MRT improve considerably the results of SIMAG.

Given the results of the algorithm AGIR for small problems, the following table 3 shows the results of the FCFS, SS, BA and AGIR algorithms for the large instances. The column Z_{best} contains the best knows solutions.

Table 3. Results of large instances

Instances	N	R	Z_{best}	Z_{FCFS}		Z_{SS}		Z_{BA}		Z_{AGIR}	
1	100	1	5612	7310	30.3%	7299	30.1%	6426	14.5%	6890	22.8%
2	100	2	453	1235	172.6%	479	5.7%	701	54.7%	453	0.0%
3	100	3	76	335	342.8%	76	0.0%	142	87.5%	76	0.0%
4	100	4	0	n/d	n/d	0	0.0%	n/d	n/d	0	0.0%
5	150	1	12329	20158	63.5%	17873	45.0%	16509	33.9%	14230	15.4%
6	150	2	1289	2622	103.5%	1390	7.9%	1623	26.0%	1320	2.4%
7	150	3	221	1440	552.2%	240	8.9%	653	195.9%	246	11.3%
8	150	4	34	1215	3473.5%	40	16.7%	134	292.4%	42	23.5%
9	150	5	0	n/d	n/d	0	0.0%	n/d	n/d	0	0.0%
10	200	1	12418	15019	20.9%	14647	18.0%	14488	16.7%	16790	35.2%
11	200	2	1541	3541	129.8%	1682	9.2%	2135	38.5%	1830	18.8%
12	200	3	281	2427	764.3%	341	21.6%	1096	290.1%	320	13.9%
13	200	4	55	2226	3947.9%	56	2.8%	316	474.5%	55	0.0%
14	200	5	0	n/d	n/d	0	0.0%	n/d	n/d	0	0.0%
15	250	1	16210	20146	24.3%	19800	22.2%	20032	23.6%	16242	0.2%
16	250	2	1961	4657	137.4%	2330	18.8%	2946	50.2%	1980	1.0%
17	250	3	290	2912	904%	341	17.5%	864	198.0%	324	11.7%
18	250	4	3	2126	70752.4%	11	271.6	400	13216.9%	3	0.0%
19	250	5	n/d	0	n/d	0	0.0%	n/d	n/d	0	0.0%
20	500	1	44832	47119	5.1%	46285	3.2%	45294	1.0%	44832	0.0%
21	500	2	5502	8633	56.9%	5707	3.7%	7564	37.5%	5502	0.0%
22	500	3	1109	6239	462.6%	1130	2.0%	3134	182.7%	1140	2.8%

23	500	4	188	4001	2027.9%	232	23.0%	2420	1186.8%	188	0.0%
24	500	5	7	3691	52628.7%	7	0.0%	1569	22308.4%	7	0.0%
					6830%		22.0%		1936.5%		6.6%

AGIR algorithm ($Av(Z_{AGIR})$=6.6) provides also better results for large instances than other algorithms ($Av(Z_{SS}, Z_{BA}, Z_{FCFS}) = \{22, 1936.5, 6830\}$).

The figure 3 shows the comparative performance of AGIR (Z_{AGIR}) and the scatter search SS (Z_{SS}), the next best performing alternative. The graph presents a comparative analysis against the Z_{opt} optimum and shows the proportion of the cases for which optimal solutions were obtained (which corresponds to a deviation of 0%). It shows the absolute and relative performance of AGIR, indexed by benchmark, compared to the optimal Z_{opt} and informs us of AGIR's performance relative to other published algorithms.

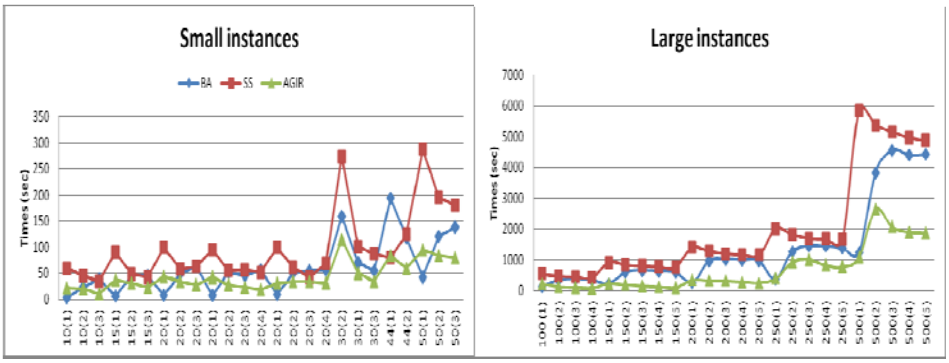

Fig. 3. A comparative performance of Z_{AGIR}, Z_{SS} and Z_{BA} (Small and Large instances)

As we can see AGIR algorithm outperforms SS for 13/24 benchmarks, provides the same results in 6/24 benchmarks and is outperformed by SS in 5/24 benchmarks, though only by relatively small margins. These results show that AGIR improves the quality of the solutions. In 12/24 benchmarks AGIR reaches the optimal solution and in other cases it is close to the optimum. From this we conclude that in AGIR is also an extremely powerful technique for solving large scale ALP problems.

Summarizing the results for both small and large scale tests, firstly we note that GARDE and AGIR generally yield superior results compared to the other algorithms. The reasons for the improvements are twofold; firstly because of the processes of intensification, diversification and memorizing, which improve the richness of the solution space and secondly the methods of exploring this space to reach close to optimal solutions. The performance of the algorithms also depends on the genetic operators and the optimization of the choice of the parameters using the experimental designs. Clearly, the combination of the GAs and TS significantly enhances the quality of the final solutions. In particular, they provide greater robustness compared to the solutions

produced when the algorithm SIMAG is used. Finally, AGIR has been shown to be an extremely powerful technique for the solution of both small and large scale ALP.

7. Conclusion

This paper shows efficient and robust hybrid GAs. These algorithms outperform, or are competitive with, some algorithms in terms of solution quality, robustness, and efficiency. Moreover, they are simple to understand and to implement. The paper also describes various tradeoffs between efficiency and solution quality obtained by changing some parameters. Currently, we focus on the dynamic case of the ALP to take into account uncertainties and stochastic elements.

References

Artiouchine K.; Baptiste P.; Dürr C. (2008): Runway sequencing with holding patterns, European Journal of Operational Research, 189 **(3)**: 1254-1266.

Baccouche M. (2007) : Métaheuristiques Hybrides D'optimisation D'atterrissage D'avions Multipistes, PhD Thesis, University of Le Havre, France

Baker J.E. (1987): Reducing Bias and Inefficiency in the selection Algorithm, Proceedings of the Second International Conference on Genetic Algorithms and their Application, pp. 14-21.

Baüerle N.; Engelhardt-Funk O.; Kolonko M (2007): 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.

Beasley J. E.; Krishnamoorthy M.; Sharaiha Y.M.; Abranmson D. (2004): Displacement problem and dynamically scheduling aircraft landings, Journal of the Operational Research Society, **55**:54-64.

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

Beasley J. E.; Krishnamoorthy M.; Sharaiha Y. M.; Abranmson D. (2000): Scheduling Aircraft Landings-the static case, Transportation Science **34(2)**, pp: 180-197.

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

Bencheikh G.; Boukachour J.; EL Hilali Alaoui A. (2011): Improved Ant Colony Algorithm to Solve the Aircraft Landing Problem, International Journal of Computer Theory and Engineering, **3, (2)**: 224–233.

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

Bolender M. A.; Slater G. L. (2000): Evaluation of scheduling methods for multiple runways, J Aircraft **37**: 410–416.

Carr G. C.; Erzberger H.; Neuman F. (2000): Fast-time study of airline-influenced arrival sequencing and scheduling, J Guidance Control Dyn **23**: 526–531.

Ciesielski V.; Scerri P. (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.

Davis L. (1991): Handbook of Genetics Algorithms, New York, USA, Van Rostrand Reinhold.

Dréo J. ; Pétrowski A. ; Siarry P. ; Taillard E. (2003) : Métaheuristiques pour l'optimisation difficile, Eyrolles Editions.

Ernst A. T.; Krishnamoorthy M. (2001): Algorithms for Scheduling Aircraft Landings, CSIRO Mathematical and Information Sciences Private Bag **10**, Clayton South MDC, Clayton VIC 3169, Australia.

Eurocontrol (2004): Operational Concept Document (OCD), Volume 1 (the vision), European Air Traffic Management Programme, Édition 2.1.

Eurocontrol (2001): Performance Review Commision. Performance review report, an assessment of air traffic management in Europe during the calendar year 2000, Technical report.

Fogel D. B. (1988): An evolutionary approach to the travelling salesman problem, Biological Cybernetics, **vol. 60**, n° 2, pp. 139-144.

Goldberg D. A. (1994): Algorithmes Génétique, Exploration, Optimisation et Apprentissage Automatique, Editions Addison Wesley.

Goldberg D. E. (1989): Genetic Algorithms in Search, Optimization and Machine Learning, Editions Addison Wesley.

Goldberg D. E.; Lingle R. (1985): Alleles, loci and the travelling salesman problem, Proceedings of the First International Conference on Genetic Algorithms and Their Applications, pp. 154-159.

Holland J. (1992): Genetic Algorithms, Scientific American, July: 66-72.

Jung G., Laguna M. (2003): Time segmenting heuristic for an aircraft landing problem, leeds school of Bisness, Univrsity of Colorado, Boulder, CO 80309, USA, working paper.

Laguna M.; Kelly J. P.; Gonzalez-Velarde J. L.; Glover F. (1995): Tabu search for the Multilevel Generalized Assignment Problem, European Journal of Operational Research **82**: 176-189.

Milan J. (1997): The flow management problem in air traffic control: a model of assigning priorities for landings at a congested airport, Transport Plann Technol **20**: 131–162.

Moscato P. (1989): On Genetic Crossover Operators for Relative Order Preservation, Caltech Concurrent Computation Program, C3P Report 778.

Oliver I. M.; Smith C. J.; Holland J. R. C. (1987): A study of permutation crossovers on the travelling salesman problem, Proceedings of the Second International Conference on Genetic Algorithms and their Applications, pp. 225-230.

Pinol H.; Beasley J. E. (2006): Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem, European Journal of Operational Research **127(2)**: 439-462.

Randall M. (2002): Scheduling aircraft landings using ant colony optimization, 6th IASTED International Conference Artificial Intelligence and Soft Computing (ASC 2002), Banff, Alberta, Canada: 129–133.

Salehipour A.; Modarres M.; Moslemi Naeni L. (2013): An efficient hybrid meta-heuristic for aircraft landing problem, Original Research Article, Computers & Operations Research, **Volume 40**, Issue 1, January 2013, Pages 207-213.

Soomer M. J.; Franx G. J. (2008): Scheduling aircraft landings using airlines' preferences, European Journal of Operational Research **190 (1)**: 277-291.

Tagushi G. (1987): System of Experimental Design, Traduction anglaise, Unipub Kraus International Publication.

Tavakkoli-Moghaddam R.; Yaghoubi-Panah M.; Radmehr F. (2012): Scheduling the sequence of aircraft landings for a single runway using a fuzzy programming approach, Journal of Air Transport Management, **Volume 25**, December 2012, Pages 15-18.

Vasconcelos J. A.; Ramirez J. A.; Takahashi R. H. C.; Saldanha R. R. (2001): Improvements in Genetic Algorithms, IEEE Trans. on AP, **37 (1)**: 3414-3417.