Journal homepage: www. jzs.uos.edu.krd
Journal of Zankoi Sulaimani
Part-A- (Pure and Applied Sciences)

Hybrid Bees Algorithm to Solve Aircraft Landing Problem

Tariq S. Abdul-Razaq[1] & Faez H. Ali[1]

1 *Mathematic department, College of Sciences, University of Al-Mustansiriya, Baghdad, Iraq*
e-mail: dr.tariqsalih@yahoo.com

Article info

Original: 04 Oct. 2014
Revised: 29 Nov. 2014
Accepted: 10 Dec. 2014
Published online:
20 March 2015

Key Words:
Aircraft Landing Problem
Swarm Intelligence
Bees Algorithm

Abstract

In this paper, we study the aircraft landing problem (ALP), which considered as one of a combinatorial optimization problems, in a single runway case. We present in the first part, a mathematical formulation of the problem with a linear objective function. In the second part, we consider the static case of the problem where all data are known in advance. We present a new heuristic for scheduling static case of aircraft landing; this heuristic is incorporated into Bees algorithm to solve this problem.

Introduction

Over the past few decades, air traffic (AT) has experienced a tremendous growth. Air transport has definitely established itself as one of the most important means of transport in the future [1]. It is well known that throughout the world AT is increasing. With these increases in traffic there is an increasing demand for decision support tools to make effective use of the limited capacity (airspace, runways, etc) available [2].

However, as the AT develops, the limitation of the runway becomes the bottleneck during the airport operation. For example, London Heathrow airport, one of the busiest airports in the world, has only two runways. When the number of approaching flights exceeds the airport capacity, some of these aircraft cannot be landed on its perfect Landing Time (LT). There is a cost mainly on the waste of fuel for each plane flying faster than its most economical speed. Airlines also experience different costs for delays of different flights [1].

Optimal scheduling of airport runway operations can play an important role in improving the safety and efficiency of the National Airspace System (NAS). Methods that compute the optimal landing sequence and LTs of aircraft must accommodate practical issues that affect the implementation of the schedule [3].

In this paper, we study the problem of landing planes at a busy airport. Given a set of planes in the radar horizon of an air-traffic controller (ATC), the problem is one of determining a LT for each plane such that each plane in this ATC horizon lands within a prespecified LT window and such that landing separation criteria specified for each pair of planes in this horizon are adhered to. This problem is a slightly unusual application that fits into the general category of machine-scheduling problems (MSP). Indeed, it is a special type of MSP with sequence-dependent processing times and with earliness and tardiness penalties for jobs not completed on target.

Psaraftis (1980) incorporated constrained position shifting (PS) within a dynamic programming recursion and considered the single runway static problem. This work is also considers the two-runway problem and builds upon the single runway solution approach by enumerating all possible partitions of the groups of planes between runways.

Dear and Sherif (1989,1991) discussed both the static and dynamic ALP and presented a heuristic algorithm (HA) for the single runway dynamic ALP based upon a technique they refer to as constrained PS.

Venkatakrishnan, et al. (1993) observed separation times adopted on landing at Logan Airport, Boston. Using these observed separation times, they modified in a heuristic manner to take account of earliest/latest times, to see the improvement that could result from better sequencing.

Beasley, et al. (2001) report on an investigation undertaken by National AT Services in the UK into improving runway utilization at London Heathrow. This investigation centered on developing an algorithm for improving the scheduling of aircraft waiting to land. The heuristic algorithm (HA) developed (a population heuristic) is discussed and results presented using actual operational data relating to aircraft landings at London Heathrow.

Min Wen (2005), in her thesis, attempts to develop a branch-and-price exact algorithm for the ALP, in which the column generation method is applied to solve the linear relaxation problem for each branch node throughout the branch-and-bound (BAB) procedure. She formulates the ALP as a set partitioning problem with side constraints on the number of available runways. She also presents a mixed integer formulation for the subproblem in column generation, which can be used to generate the columns with the minimum negative reduced cost.

Bencheikh et al. (2011) study the ALP in the multiple runway (MR) case. They present in the first part, a mathematical formulation of the problem with a linear and nonlinear objective function. In the second part, they consider the static case and they present a new HA for scheduling aircraft landing on a single runway, this HA is incorporated into an ant colony algorithm to solve the MR case.

In paper of Mahmoudian et al. (2013), a MR case of the static ALP is considered. It is assumed that one of the runways is not available. A mixed integer formulation approach is proposed to deal with the problem. The approach of solving this problem is breaking down the main problem into several sub problems.

This paper is organized as follows. In Section II, we describe the ALP. In section III, we present the single runway formulation of the ALP. In section IV, we demonstrate some types of heuristic and local search solution methods. Some techniques which are be helpful to improve the solution of ALP and reduce the computations to increase the speed of the good solution approach, in addition, we will discuss some special case of ALP, and all of these are introduced in section V. In section VI, we will improve the heuristic method; discuss the complete enumeration method using exhaustive search and the improved heuristic. Finally, we use a new local search to solve ALP. Computational results for the proposed methods for a number of test problems involving up to 50 planes are presented in this section.

We should also stress here that we are dealing only with the static case. In other words, we are dealing with the off-line case where we have complete knowledge of the set of planes that are going to land. The dynamic, or on-line, case, where decisions about the LTs for planes must be made as time passes and the situation changes (planes land, new planes appear, etc.) is the subject of a separate paper (Beasley et al., 1995 [12]).

Aircraft Landing Problem (ALP)

A. Problem Description

Upon entering within the radar range (radar horizon) of ATC at an airport, a plane requires ATC to assign it a LT, sometimes known as the broadcast time, and also, if more than one runway is in use, assign it a runway on which to land. The LT must lie within a specified time window, bounded by an earliest time and a latest time, these times being different for different planes. The earliest time represents the earliest a plane can land if it flies at its maximum airspeed. The latest time represents the latest a plane can land if it flies at its most fuel-efficient airspeed while also holding (circling) for the maximum allowable time.

Each plane has a most economical, preferred speed, referred to as the cruise speed. A plane is said to be assigned its preferred time, or target time, if it is required to fly in to land at its cruise speed. If ATC requires the plane to either slow down, hold, or speed up, a cost will be incurred. This cost will grow as the difference between the assigned LT and the target LT grows [7].

The time between a particular plane landing, and the landing of any successive plane, must be greater than a specified minimum (the separation time) which is dependent upon the planes involved. Separation times are bounded below by aerodynamic considerations. A Boeing 747, for example, generates a great deal of air turbulence (wake vortices) and a plane flying too close behind could lose aerodynamic stability. Indeed a number of aircraft accidents are believed to have been caused by this phenomenon (Mullins, 1996). For safety reasons, therefore, landing a Boeing 747 necessitates a (relatively) large time delay before other planes can land. A light plane, by contrast, generates little air turbulence and, therefore, landing such a plane necessitates only a (relatively) small time delay before other planes can land. Planes taking off impose similar restrictions on successive operations [7].

B. ALP Objective

In this paper, we shall assume that we are minimizing total cost (the total earliness and tardiness cost), where the cost for any plane is linearly related to deviation from its target time. Fig.1 illustrates the variation in cost within the time window of a particular plane. Note here that, in Fig.1, we have a cost of zero for the plane landing at its target time.

Note here that the objective function used is related to the issue of the viewpoint we adopt. If we are the airport operator, we may be interested in improving the long-term utilization of finite runway capacity by better scheduling. If we are an individual airline, we may be more interested in the individual plane costs incurred. In this paper, we implicitly assume, through the form of Fig.1, that we are interested in minimizing total plane costs.

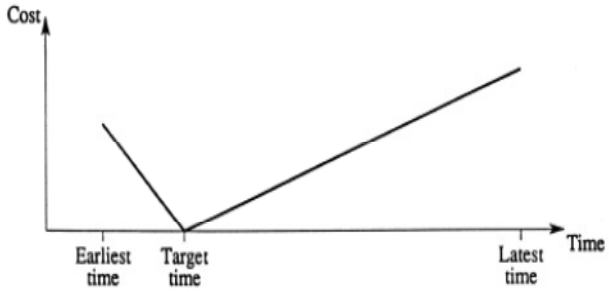

Fig.1: Variation in cost for a plane within its time window.

Single Runway Formulation of ALP

In this section, we present an initial mixed-integer zero–one formulation of the static single runway ALP.

A. Notations

N: the number of planes.

P : set of N planes, P={1,2,…,N}.

E_i: the earliest landing time (ELT) for plane $i \in P$.

L_i: the latest landing time (LLT) for plane $i \in P$.

T_i: the target (preferred) landing time (TLT) for plane $i \in P$.

S_{ij}: the required separation time (≥ 0) between plane i landing and plane j landing(where plane i lands before plane j), $i,j \in P$, $i \neq j$.

g_i: the penalty cost (≥ 0) per unit of time for landing before the target time T_i for plane $i \in P$.

h_i: the penalty cost (≥ 0) per unit of time for landing after the target time T_i for plane $i \in P$.

The variables are:

t_i: the scheduled (actual) landing time (SLT) for plane $i \in P$.

α_i: how soon plane $i \in P$ lands before T_i, mathematically the earliness $\alpha_i = \max\{0, T_i - t_i\}$.

β_i: how soon plane $i \in P$ lands after T_i, mathematically the tardiness $\beta_i = \max\{0, t_i - T_i\}$.

$$\delta_{ij} = \begin{cases} 1 \text{ if plane } i \text{ lands before plane } j \; \forall i, j \in P, i \neq j. \\ 0 \text{ otherwise.} \end{cases}$$

Without significant loss of generality, we shall henceforth assume that the times E_i, L_i, and S_{ij} are integers.

B. ALP Constraints [7]

To clarify some of the constraints that will be given in this section, we provide a diagram (Fig.2), which depicts overlapping time windows for planes i and j.

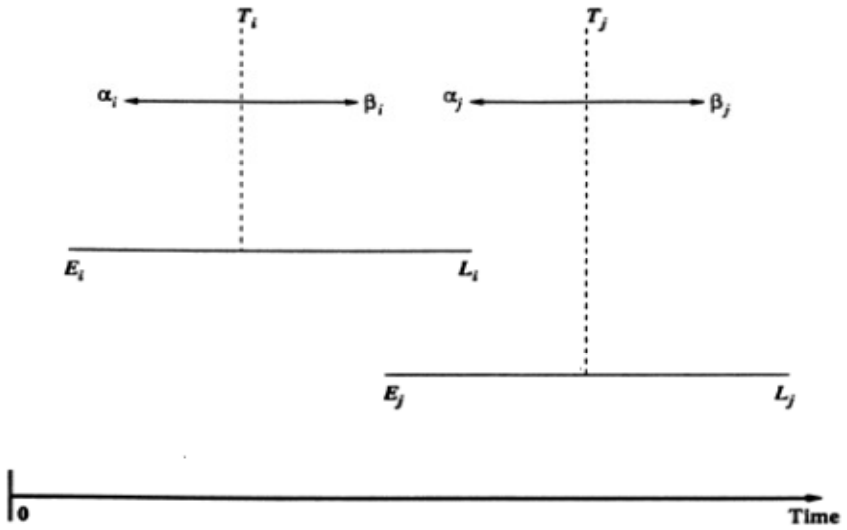

Fig.2: An example of overlapping time windows.

The first set of constraints are

$$E_i \leq t_i \leq L_i, \; \forall i \in P, \qquad (1)$$

which ensure that each plane lands within its time window. Now, considering pairs (i,j) of planes we have that

$$\delta_{ij} + \delta_{ji} = 1, \; \forall i,j \in P, \; i<j, \qquad (2)$$

In words, either plane i must land before plane j ($\delta_{ij}=1$) or plane j must land before plane i ($\delta_{ji}=1$). It is trivial to see that, for certain pairs (i,j) of planes, we can immediately decide whether $\delta_{ij}=1$ or whether $\delta_{ji}=1$. For example, if the time windows for two planes i and j are [10,50] and [70,110], respectively, then it is clear that plane i must land first (i.e., that $\delta_{ij}=1$). However, even if we know, for a pair of planes (i,j), the order in which they land, it does not mean that the separation constraints (SC) is automatically satisfied.

Continuing the example given above then, if the separation time $S_{ij}=15$, the SC is automatically satisfied, regardless of when within their respective time windows ([10,50] and [70,110]) planes i and j land. However, if $S_{ij}=25$, the SC is not automatically satisfied, i.e., there exist LTs for i and j (within their respective time windows) that violate the SC. Hence we need to define three sets:

U: the set of pairs (i,j) of planes for which we are uncertain whether plane i lands before plane j.

V: the set of pairs (i,j) of planes for which i definitely lands before j (but for which the SC is not automatically satisfied).

W: the set of pairs (i,j) of planes for which i definitely lands before j (and for which the SC is automatically satisfied).

Then, we can define the set W by:

$W=\{(i,j)|L_i<E_j \text{ and } L_i+S_{ij}\leq E_j, \forall i,j\in P, i\neq j]\}$ (3)

In words, i must land before j ($L_i<E_j$) and the SC is automatically satisfied ($L_i+S_{ij}\leq E_j$).

We can define the set V by:

$V=\{(i,j)|L_i<E_j \text{ and } L_i+S_{ij}>E_j, \forall i,j\in P, i\neq j]\}$ (4)

In words, i must land before j ($L_i<E_j$) but the SC is not automatically satisfied ($L_i+S_{ij}>E_j$).

Some thought reveals that the pairs of planes (i,j) for which uncertainty exists with respect to which plane lands first must have overlapping time windows. Hence, we can define the set U as

$U=\{(i,j)|i,j=1,\ldots,N, i\neq j; E_j\leq E_i\leq L_j$ or

$E_j\leq L_i\leq L_j, E_i\leq E_j\leq L_i$ or $E_i\leq L_j\leq L_i\}$ (5)

The definition of U means that one of the end points of the time window of one plane falls within the time window of the other.

We therefore have the constraint,

$\delta_{ij}=1 \ \forall(i,j)\in W\cup V$ (6)

We need a SC for pairs of planes in V, and this is

$t_j\geq t_i+S_{ij} \ \forall(i,j)\in V$ (7)

which ensures that a time S_{ij} must elapse after the landing of plane i at t_i before plane j can land at t_j.

$t_j\geq t_i+S_{ij}\delta_{ij}-(L_i-E_j)\delta_{ji} \ \forall(i,j)\in U$ (8)

Finally, we need constraints to relate the α_i, β_i, and t_i variables to each other. The variables α_i and β_i and the constraints presented below are necessary simply to ensure that we have a linear objective function and are:

$\alpha_i \geq T_i - t_i, \quad i\in P,$ (9)

$0 \leq \alpha_i \leq T_i - E_i, \ i\in P,$ (10)

$\beta_i \geq t_i - T_i, \quad i\in P,$ (11)

$0 \leq \beta_i \leq L_i - T_i, \ i\in P,$ (12)

$t_i = T_i - \alpha_i + \beta_i, \ i\in P,$ (13)

Equations 9 and 10 ensure that α_i is at least as big as zero and the time difference between T_i and t_i, and at most the time difference between T_i and E_i (see Fig. 2). Equations 11 and 12 are similar equations for β_i. Equation 13 relates the LT (t_i) to the time i lands before (α_i), or after (β_i), target (T_i).

C. Objective Function [7]

We now need only to set up the objective function, minimize the cost of deviation from the target times (T_i), and this is

$$\text{minimize } \sum_{i=1}^{N}(g_i\alpha_i + h_i\beta_i) \quad (14)$$

The complete formulation (model) of the single runway problem is therefore to minimize function (14) subject to Eqs. 1, 2, and 6-13.

Some Solution Methods of ALP

A. First Come, First Serve Ordering [9]

In order to prevent a trailing aircraft from losing aerodynamic stability because of turbulence (wake vortices) generated by a leading aircraft, the Federal Aviation Administration (FAA) has fixed minimum separation requirements between aircraft. Thus, to guarantee safety, the FAA has generally partitioned aircraft into three weight classes: Small, Large, Heavy; and fixed in-trail separation requirements. The time separation requirements are then a function of the plane speed and the length of the final approach path. An example for a 6 miles long final approach path is given in Table.1, and was derived in Odoni [11],[10]. We use these numerical values for time separation in our simulations.

Table.1: Time separation requirements (in s) and classes speed (in knots).

	Small	Large	Heavy	Speed(kts)
Small	82	69	60	110
Large	131	69	60	130
Heavy	196	157	96	150

If we consider one unique class, then the separation times between all successive aircraft are equal, since all planes of the same weight class should enter the Terminal Maneuvering Area (TMA) at the same speed. In this case, the "First Come, First Served" (FCFS) landing order (that is, ordering with respect to ELT, ensures an optimal LT for the whole set of planes. Now if we consider several classes, the FCFS strategy, despite being fair and easy to implement, is no longer optimal due to the assymetry in the separation requirements, underlined above. That said, if we consider a particular landing sequence, applying the FCFS strategy within each class minimizes the makespan, corresponding to this particular sequence.

B. Parallel Improving Technqiue

a. Aircraft Sequencing [6]

The order between aircraft is set up according to priority rules (PR's) which are based on the variables:

- $\mathbf{E_i}$: The priority is given to the aircraft which has the sooner ELT; this PR can be used if we are interested by the maximal exploitation of the runway.
- $\mathbf{T_i}$: The priority is given to the aircraft which has the earliest TLT; We can choose this priority if we want to minimize the advance or late made by aircraft
- $\mathbf{L_i}$: The priority is given to the aircraft which has the earliest LLT, in order to avoid that, we assign a LT after the LLT of an aircraft.
- $\mathbf{E_i/g_i}$: The priority is given to the aircraft which has the soonest earliest time and which causes the most important advance penalty.
- $\mathbf{L_i/h_i}$: The priority is given to the aircraft which has the soonest latest time and which causes the most important lateness penalty.
- $\mathbf{Ti\ /(g_i+h_i)}$: The priority is given to the aircraft which has the soonest target time and which causes the most important advance and lateness penalty.
- $\mathbf{1/(g_i+h_i)}$: The priority is given to the aircraft which causes the most important advance and lateness penalty.

b. Adjusting Aircraft LTs [6]

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

c. Parallel Improving Algorithm (PIA)[6]

If we increase (reduce) the aircraft LT, we check the interval of Improving the landing aircraft time during the initialization: in this algorithm, we adjust the LT of each aircraft during its time assignment. Let O be the set of aircraft which have previously been assigned a LT.

Parallel Improving Algorithm:

Let P be the list of aircraft set up according to a PR and O={ }.

1. $t_{P1} \leftarrow T_{P1}$; $P_1 \in O$.

2. **FOR** i = 2 : N

$t_{Pi} \leftarrow \max\{(E_{Pi} \text{ or } T_{Pi}), \max_{j \in O} (t_j + S_{j,Pi})\}$

END {FOR i}

3. **REPEAT**

IF ($t_{Pi} > T_{Pi}$)

Reduce the LT by 1 unit of time

ELSE

Increase the LT by 1 unit of time

END {IF}

IF (the solution is unfeasible)

Reject the change and keep the last feasible solution.

BREAK.

END {IF}

WHILE (there is decrease of penalty cost)

Classical Bees Algorithm (CBA)

The artificial intelligence is used to explore distributed problem solving without having a centralized control structure. This is seen to be a better alternative to centralized, rigid and preprogrammed control. Real life swarm intelligence (SI) can be observed in ant colonies, beehives, bird flocks and animal herds.

The most common examples of SI systems: Ant Colony Optimization, Particle Swarm Optimization and Marriage in Honey Bees Optimization (MBO).

The main processes in MBO are: the mating flight of the queen bee with drones, the creation of new broods by the queen bee, the improvement of the broods' fitness by workers, the adaptation of the workers' fitness, and the replacement of the least fit queen with the fittest brood [4].

The challenge is to adapt the self–organization behavior of the colony for solving the problems. The **Classical Bees Algorithm (CBA)** is an optimization algorithm inspired by the natural foraging behavior of honey bees to find the optimal solution.

The algorithm requires a number of parameters to be set, namely:

N: Number of scout bees (*n*).

m: Number of sites selected out of n visited sites (*m*).

e: Number of best sites out of m selected sites (*e*).

nep: Number of bees recruited for best e sites (*nep*).

nsp: Number of bees recruited for the other (*m–e*) selected sites (*nsp*).

ngh: Initial size of patches (*ngh*) which includes site and its neighborhood and stopping criterion.

The pseudo code for the CBA is shown below in its simplest form [5].

Classical Bees Algorithm

INPUT: *n*, *m*, *e*, *nep*, *nsp*, Max. iterations.

1. Initialize population with random solutions.
2. Evaluate fitness of the population.
3. **REPEAT**
4. Select sites for neighborhood search**.**
5. Recruit bees for selected sites (more bees for best e sites) and evaluate fitness.
6. Select the fittest bee from each patch**.**
7. Assign remaining bees to search randomly and evaluate their fitness's.
8. **UNTIL** stopping criterion is met.

OUTPUT: Optimal or near optimal solutions**.**

END.

Techniques to Improve the Solution and Reduce the Computations

In this section we demonstrate two types of methods which are contribute in improving the solution and speed the approach to the good solution. In addition, we will discuss some special cases of ALP.

A. Time Window Tightening (TWT) [7]

Let Z_{UB} be any upper bound on the optimal solution to the problem. Then, it is possible to limit the deviation from target for each plane. Specifically, for plane i, if we assume that all other planes make a zero contribution to the objective function (Eq. 14) value, we can update E_i using

$$E_i = \max\{E_i, T_i - Z_{UB}/g_i\},\ i \in P, \quad (15)$$

because, if we land more than Z_{UB}/g_i time units before target, we would exceed the upper bound on the optimal solution. Similarly we have that

$$L_i = \min\{L_i, T_i + Z_{UB}/h_i\},\ i \in P, \quad (16)$$

because, if we land more than Z_{UB}/h_i time units after target, we would exceed the upper bound on the optimal solution.

Using equations 15 and 16, the time window $[E_i, L_i]$ for each plane $i \in P$ can be tightened in a preprocessing step. The benefit of tightening (closing) the time windows is that (potentially) the sets U and V can be reduced in size, thereby giving a smaller problem to solve.

Example (1): For N=3, lets have the following ALP information.

	P_1	P_2	P_3
E_i	129	195	89
T_i	155	258	98
L_i	559	744	510
g_i	10	10	30
h_i	10	10	30

and

	S_{ij}		
	1	2	3
1	0	3	15
2	3	0	15
3	15	15	0

The time window tightening of example.1 using Eq. (15) and (16) for instance, Z_{UB}=1060 is as shown in table. 2.

Table.2: time window tightening of example (1): for Z_{UB}=1060.

	P_1	P_2	P_3
E_i	129	195	89
T_i	155	258	98
L_i	261	364	133
g_i	10	10	30
h_i	10	10	30

In table.3, for example.1, we show the difference between the ALP information before and after TWT using exhaustive search method will mention in subsection (VI.B.a).

Table.3: The difference between the ALP information before and after TWT.

i	Seqπ	Before TWT		After TWT	
		t_i	Cost Z	t_i	Cost Z
1	3,2,1	98,195,198	1060	98,258,261	1060
2	3,1,2	98,155,258	0	98,155,258	0
3	2,3,1	195,210,225	4690	No FS	-----
4	2,1,3	195,198,213	4510	No FS	-----
5	1,2,3	155,195,210	3990	No FS	-----
6	1,3,2	129,144,258	1640	No FS	-----

FS is feasible solution.

B. *Successive Rules (SR)*

Reducing the current sequence is done by using several SR's. When, for each i ($i\in P$), and with its cost given in the objective function (14), we can derive successive rules (SR) that restrict the search for an optimal solution. Such rules can be used in any optimization algorithms. These improvements lead to very large decrease in the number of solutions to obtain the optimal solution [8].

The ALP is treated as a scheduling problem, so we may find SR's can be useful to eliminate number of solutions to reducing the computations time.

Definition (1): Let $W_i=[E_i,L_i]$ be the time window interval of plane $i\in P$, if $W_i\bigcap W_j=\phi$ (time windows are disjoint) and $L_i<E_j$ we denote for the interval W_i proceed the interval W_j in line number by $W_i \Im W_j$.

Definition (2): We say that plane i proceeds the plane j (we write $i\rightarrow j$ or $(i,j)\in P$) or j proceeds the plane i if $W_i\cap W_j=\phi$, for $i\neq j$.

Remark (1):

1. $t_i < t_j$ iff $i\rightarrow j$, $\forall i,j\in P$, $i\neq j$.
2. if $E_i\leq E_j\leq L_i$ or $E_i\leq L_j\leq L_i$, then $W_i\bigcap W_j\neq\phi$ for $i\neq j$, we say that W_i and W_j are overlapped.

Proposition(1): if $W_i \Im W_j$, then $t_i\in W_i< t_j\in W_j$, $\forall i,j\in P$, $i\neq j$.

Proof: since $W_i \Im W_j$, then $t_i\notin W_j$ and $t_j\notin W_i$. Suppose $t_i\geq t_j$, for $t_i=t_j$, $t_j=t_i\in W_i$, C!.

For $t_i>t_j$, if $t_j\in W_i$ C!. Take $t_j\notin W_i$. $\therefore$ $t_j\in$ another interval say W_k, s.t. $W_k \Im W_j$, but $t_j\in W_j$ and that is a contradiction since there is no integer belong to two disjoint intervals in the same time. $\therefore$ $t_i<t_j$ □

Remark (2): if $W_i\bigcap W_j=\phi$, then $L_i < E_j$ or $L_j < E_i$ for $i\neq j$.

Definition (3): We say that $i\rightarrow j$ if one of the following conditions is satisfied:

1. $L_i < E_j$ for $i\neq j$.
2. For $L_i \geq E_j$, if $L_i < E_j+S_{ij}$ for $i\neq j$.

Conditions of SR are shown in fig.3.

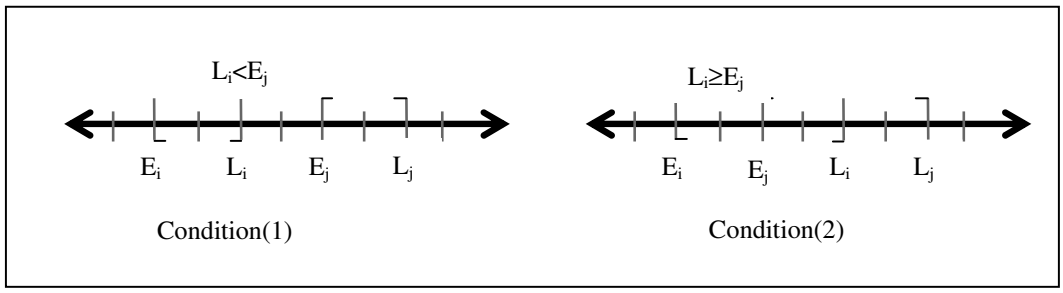

Fig.3: Conditions of dominiance rules.

Example (2): For N=5 lets have the following ALP information:

	P_1	P_2	P_3	P_4	P_5
E_i	129	89	96	111	123
T_i	155	98	106	123	135
L_i	191	110	118	135	147
g_i	10	30	30	30	30
h_i	10	30	30	30	30

and

S_{ij}	1	2	3	4	5
1	0	15	15	15	15
2	15	0	8	8	8
3	15	8	0	8	8
4	15	8	8	0	8
5	15	8	8	8	0

From definition (3), condition (1) we obtain the following SR's:

$2\rightarrow1$, $2\rightarrow4$, $2\rightarrow5$, $3\rightarrow1$, $3\rightarrow5$.

From condition (2), we have $3\rightarrow4$ because of $E_4+S_{34}=111+8=119 > L_3=118$, and $4\rightarrow1$ because of $E_1+S_{41}=129+15=144 > L_4=135$. Fig. 4 shows the SR's of example (2).

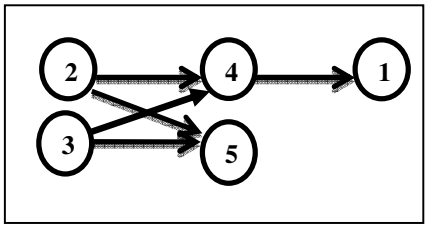

Fig.4: Graph of SR of example (1).

The adjacency matrix A of the graph shown above is:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & a_{15} \\ 1 & 0 & a_{23} & 1 & 1 \\ 1 & \bar{a}_{23} & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & a_{45} \\ \bar{a}_{15} & 0 & 0 & \bar{a}_{45} & 0 \end{bmatrix} \end{array}$$

Note: the sequencing problem of this ALP can solved by 2^3=8 possible and no need to try 5!=120 possible.

C. Special Cases

Definition (4): Let $S=\max\{S_{ij}\}$, $\forall i,j\in P, i\neq j$, then W_i is called **logical time window** if the length ℓ_i of W_i, for $i\in P$ is $\ell_i=L_i-E_i+1\geq 2S$ and $T_i=(E_i+L_i)/2$.

Example(3): let $W_1=[10,20]$ and $W_2=[25,50]$ $S_{12}=10$, $S=10$. Note that $\ell_1=11$ and $\ell_2=26$, W_2 is logical time window but W_1 is not.While if $W_1=[10,15]$ and $W_2=[16,24]$, $S_{12}=15$, $S=15$. Note that both W_1 and W_2 are not logical time windows, since if $t_1=E_1=10$, then $t_2<t_1+S_{12}=10+15=25>L_2=24$, that mean is not logical time definitely, not satisfies the SC.

Case (1): Let $W_{i1},W_{i2},\ldots,W_{iN}$ are all disjoint logical time windows in this sequence s.t. $W_{i_k}\bigcap W_{i_j}=\phi$,$\forall i_k,i_j\in P$, $i_k\neq i_j$, then the optimal solution is Z=0 at $t_{i_1}=T_{i_1}<t_{i_2}=T_{i_2 i_N}<\cdots<t_{i_N}=T_{i_N}$ and $i_1\rightarrow i_2\rightarrow\ldots\rightarrow i_N$.

Proof: Without loosing the generality, let N=3 for sequence π=(1,2,3).

Since W_1,W_2 and ,W_3 are logical time windows this mean $S=\max\{S_{ij}\}$, $\forall i,j\in P$.

Let $t_1=T_1$, $T_1+S\leq L_1<E_2<T_2$, then take

$$t_2=T_2>T_1+S=t_1+S \qquad (a)$$

$\therefore$ $t_1=T_1$ and $t_2=T_2$ satisfy the window and separation conditions (WSC's). By applying (a) again for t_2 and t_3 we obtain that: $t_2=T_2$ and $t_3=T_3$ satisfy the WSCs.

$\therefore$The optimal solution is the cost Z=0 and $1\rightarrow 2\rightarrow 3$.

Naturally, this case can be applied for N aircraft and for any sequence π. □

Case (2): Let $W=W_1=W_2=\ldots=W_N$ be the same large time window, then the optimal solution Z=0 at $t_{i_k}=T_{i_k}$ if T_{i_k} satisfies the SC $\forall i_k\in P$ and $i_1\rightarrow i_2\rightarrow\ldots\rightarrow i_N$.

Proof: let's take any arbitrary sequence π=(1,2,…,N). Since T_{i_k} satisfy the SCs, this means:$T_1\leq T_2-S_{12},T_2\leq T_3-S_{23},\ldots,T_{N-1}\leq T_N-S_{N-1,N}$. If we take $t_{i_k}=T_{i_k}$, then the LTs t_{i_k} satisfy the SC $\forall i_k\in P$.

$\therefore$The optimal solution with cost Z=0 and $1\rightarrow 2\rightarrow\ldots\rightarrow N$. □

Of course, this case can be applied for any sequence π.

Solving Aircraft Landing Problem

We first observe that the ALP involves two decision problems: (1) a sequencing problem (which determines the sequence of plane landings) and (2) a scheduling decision problem (which determines the precise LTs for each plane in the sequence, subject to the SCs). We observe that if we are given the sequence in which planes land, then we can develop an algorithm for resolving the scheduling decisions.

A. Modified PIA

As mentioned before, the PIA, found by [6], implements the sequencing using one of the PR's and the scheduling represented by increases (decreases) the t_i value by one unit if it less (larger) than T_i. In this section we attempt to modify the PIA to improve the efficiency of PIA. We called the new PIA by **Modified PIA** (MPIA).

The modification is represented by checking the interval of improving the landing aircraft time during the initialization before increase (reduce) the aircraft LT by one unit and calculating the penalty cost (say Z_0) then we increase (reduce) the aircraft LT by one unit as mentioned in ALP, then calculating the penalty cost (say Z), if $Z>Z_0$, we ignore Z and keeping Z_0 otherwise we take Z.

MPIA:

Let P be the list of aircraft set up according to a PR and O={}.

1. $t_{P1} \leftarrow T_{P1}$; $P_1 \in O$.
2. **FOR** i = 2 : N

   $t_{Pi} \leftarrow \max(T_{Pi}, \max_{j \in O} (t_j + S_{j,Pi}))$

   END {FOR i}
3. **REPEAT**

   Calculate penalty Cost Z

   IF ($t_{Pi} > T_{Pi}$)

   Reduce the LT by 1 unit of time

   ELSE

   Increase the LT by 1 unit of time

   END {IF}

   IF (the solution is unfeasible)

   Reject the change and keep the last feasible solution.

   BREAK.

   END {IF}

WHILE (there is decrease of penalty cost)

Example (4): The information of this ALP example obtained from [6] for N=10, we take the sequence π=(3,4,5,6,7,8,9,1,10,2). Table.4 shows the difference between the results obtained from implementation of PIA and MPIA for this example.

Table.4: Difference between the results obtained from PIA and MPIA for example (4).

stage	Alg.	3	4	5	6	7	8	9	1	10	2	Cost Z
1	PIA	98										0
	MPIA	98										0
2	PIA	98	106									0
	MPIA	98	106									0
3	PIA	98	106	123								0
	MPIA	98	106	123								0
4	PIA	98	106	123	135							0
	MPIA	98	106	123	135							0
5	PIA	98	106	123	134	142						150
	MPIA	98	106	123	131	139						150
6	PIA	98	106	122	130	138	146					360
	MPIA	98	106	122	130	138	146					360
7	PIA	98	106	121	129	137	145	153				510
	MPIA	98	106	122	130	138	146	154				480
8	PIA	98	106	120	128	136	144	152	167			630
	MPIA	98	106	122	130	138	146	154	169			620
9	PIA	98	106	118	126	134	142	150	165	180		700
	MPIA	98	106	118	126	134	142	150	165	180		700
10	PIA	98	106	118	126	134	142	150	165	180	258	700
	MPIA	98	106	118	126	134	142	150	165	180	258	700

Table.5: Comparison results between PIA and MPIA.

N	Alg.	Penalty cost according to the PR's							Best Z	Opt.	CP U/s
		E_i	T_i	L_i	$1/(g_i+h_i)$	$T_i/(g_i+h_i)$	E_i/g_i	L_i/h_i			
10	PI	1280	700	3470	930	930	930	1140	700	700	0.003
	MPI	1280	700	2360	880	880	880	1090	700		
15	PI	1810	1550	1830	2290	1530	1530	1550	1530	1480	0.006
	MPI	1790	1500	2210	2210	1480	1480	1770	1480		
20	PI	2000	1820	8680	7170	840	900	2730	840	820	0.008
	MPI	1790	1730	5890	5000	820	880	2140	820		
20	PI	5410	3620	15140	4420	3620	3010	4960	3010	2520	0.008
	MPI	4890	2520	6100	4210	2520	2650	3260	2520		
20	PI	9810	6730	10370	4390	3040?	3220	3160	3040	3100	0.006
	MPI	6470	5420	7060	4060	3100	3250	3220	3100		
30	PI	67525	67525	67525	116972	46270	46270	43670	43670	24442	0.01
	MPI	24442	24442	24442	89766	31194	31194	37956	24442		
44	PI	39058	39058	39058	79946	49528	42738	50793	39058	1550	0.03
	MPI	1550	1550	1550	73536	40330	33820	43443	1550		
50	PI	19055	2600	113580	271650	91185	103650	147940	2600	1950	0.02
	MPI	18790	2530	61215	215025	152270	168505	141570	2530		
Tot.	PI	1	3	1	0	4	1	0	10		
	MPI	2	5	2	0	4	1	0	14		

In table.5, we demonstrate the comparison total penalty cost (Z) results between PIA obtained from [6] and MPIA for sequencing using different priorities. The best results are the shaded cells. From table.5, the best priorities is T_i and $T_i/(g_i+h_i)$ for PIA and MPIA. The cost(20)=3040 signed with (?) obtained from PIA may be incorrect since its less than the optimal value 3100.

From the results of table.5, although the MPIA is heuristic, it has good performance so it may be able to be used to improve any results of solving ALP obtained from any method or algorithm e.g. local search methods.

B. Complete Enumeration Method (CEM)

When using CEM, in sequencing stage we will try all the possible permutation of N planes which equal to N!, while in scheduling stage we will apply two methods:

- Exhaustive Search Method (ESM): in this method we try all possibilities starting from E_i to L_i. The total number of all possibilities for scheduling is $\prod_{i=1}^{N}(L_i - E_i + 1)$.
- MPIA.

Note that the total complexity (C(N)) for sequencing and scheduling N-planes using CE is:

$$C(N)=N!*\prod_{i=1}^{N}(L_i - E_i + 1) \qquad (17)$$

For $E_i=T_i=L_i$, (Z=0) $\forall i \in P$, then C(N)=N!.

Remark (3):

In general, if R is the number of aircraft satisfy the SR's and D be the number of craft which are not submitted to SR (represented by the variables a_{ij} in matrix A) where $R+D=C_2^N=N*(N-1)/2$, for the ALP we have 2^D sequences can be try to find the best sequence. In some ALP, N! may be larger than 2^D and vice versa. Table.6 shows samples of ALP examples with C(N) and 2^D for different N.

Table.6 samples of ALP examples with C(N) and 2^{ND}.

N	**C(N)**	**N!**	$\mathbf{C_2^N}$	**R**	**D**	$\mathbf{2^D}$
8	9.7×10^{16}	40320	28	17	11	2048*
9	2.5×10^{20}	362880*	36	17	19	524288
10	1.9×10^{23}	3628800*	45	23	22	4194304
15	5.1×10^{41}	1.3×10^{12}*	105	44	61	2.3×10^{18}

The cells signed with (*) means is better to be used in search to find good sequence.

a. Complete Enumeration using ESM

In this subsection, we introduce the results of optimal penalty cost Z using the exact ESM for N=3,…,9. The results obtained after applying TWT and find SR. The TWT very important factor to reduce the calculations, since it implies to:

1. Increase the number of R (i.e. decrease the number D).
2. Reduce the complexity of CE.

These two factors will reduce the CPU time of CE to approach the optimal solution. In table.7, we show the influence of TWT on R, C(N=6) and CPU time for different choices of Z_{UB}.

Table.7: the influence of TWT on R, C(6) and CPU time for different Z_{UB}.

i	$\mathbf{Z_{UB}}$	**R**	**C(N)**	**NF**	**CPU/s**
1	0	15	6!=720	1	0.2
2	480	11	3.529993×10^{12}	8	110
3	600	9	8.627828×10^{12}	18	306
4	770	7	2.362645×10^{13}	25	1071
5	1080	6	9.397747×10^{13}	48	6427

Where NF is the number of best feasible solutions in specific sequencing. Table.8 shows the results of applying CE using ESM for N=3,4,…,9, with CPU time.

Where OS is the optimal sequence and OC is the optimal scheduling, which are both corresponding to optimal penalty cost Z. When using ESM, the wasting of CPU time done in scheduling stage especially when NF is high.

Table.8: The results of applying CE using ESM for N=3,4,…,9, with CPU time.

N	N!	OS	OC	Z_{UB}	Opt. Z	C(N)	NF	CPU/s
3	6	3,1,2	98,155,258	0	0	6	1	0.06
4	24	3,4,1,2	98,106,155,258	0	0	24	1	0.06
5	120	3,4,5,1,2	98,106,123,155,258	0	0	120	1	0.07
6	720	3,4,5,6,1,2	98,106,123,135,155,258	0	0	720	1	0.2
7	5040	3,4,5,6,7,1,2	98,106,123,131,139,155,258	150	150	7.800409×10^{11}	1	3
8	40320	3,4,5,6,7,8,1,2	98,106,122,130,138,146,161,258	420	420	$9.689287e\times10^{16}$	32	1456
9	362880	3,4,5,6,7,8,9,1,2	98,106,122,130,138,146,154,258,266	620	620	2.486859×10^{20}	>136	>3hs

b. Complete Enumeration using MPIA

In this subsection, we introduce the results of best penalty cost Z using the heuristic MPIA for N=3,…,9. The results obtained after applying TWT and SR. Table.9 shows the results of applying CE using MPIA for N=3,…,9, with CPU time.

Where BS is the best sequence and BC is the best scheduling, which both are corresponding to best penalty cost Z. Note the difference in CPU time for CE using ESM and MPIA.

Table.9: The results of applying CE using MPIA for N=3,…,9, with CPU time.

N	BS	BC	Best Z	NF	CPU/s
3	3,1,2	98,155,258	0	1	0.05
4	3,4,1,2	98,106,155,258	0	1	0.05
5	3,4,5,1,2	98,106,123,155,258	0	1	0.05
6	3,4,5,6,1,2	98,106,123,135,155,258	0	1	0.05
7	3,4,5,6,7,1,2	98,106,123,131,139,155,258	150	1	0.14
8	3,4,5,6,7,8,1,2	98,106,122,130,138,146,161,258	420	19	1.1
9	3,4,5,6,7,8,9,1,2	98,106,122,130,138,146,154,258,266	620	129	6

c. Solving ALP using Classical BA (CBA)

Classical BA Description

The problem therefore is to decide values for the t_i which lie in the time windows $[E_i,L_i]$ and satisfy the separation criteria, whilst attempting to ensure that aircraft land at (or before) their target time. Below we discuss the elements of our BA relating to:

- representation, how we represent a solution to the problem in our BA;
- fitness, assessing the value of a solution;
- dealing with the separation constraints (SC's);
- population replacement, placing the new bees in the population.

Each of these elements is discussed separately below.

For **CBA representation**, the first decision in any BA is how to represent a solution to the problem. To answer this question, we should leave a bee to be an integer vector, or permutation of $\pi_k=(i_1,...,i_N)$ integers, $i_j \in P$ and $k \in NB$ (number of bees in population).

For the initial population, the process begins with a random set of n bees consisting of permutations of the integers 1,…,n, then it applies such sample, and assesses the "fitness" of each bee.

Fitness in BAs relates to assessing the value (worth) of an individual. This issue relates to the overall objective, namely what are we trying to achieve when we schedule a set of aircraft to land. All of these functions were based on the difference between the SLT and the TLT. Let α_i and β_i be the difference (deviation) between the t_i and the T_i for aircraft i. Note in particular that, if $T_i - t_i > 0$, then an aircraft lands before its target time and if $T_i - t_i<0$ then an aircraft lands after its target time, i.e. it is delayed. Then the fitness function is:

$$\text{Fitness} = Z = \sum_{i=1}^{N} (g_i\alpha_i + h_i\beta_i) \qquad (18)$$

This fitness function is linear: large deviations receive a disproportionately larger weighting. Since we seek to minimize fitness this function implies that we would prefer to land aircraft near their target time and dislike landing aircraft faraway their target time.

For the **SCs**, Probably the most fundamental difficulty in applying BAs to the problem of scheduling aircraft landings is to try and ensure that the BA generates solutions that are feasible (satisfy the constraints). As discussed above the representation we've adopted automatically means that the time window constraints are satisfied. This therefore leaves the SCs. In our BA for scheduling aircraft landings we dealt with the problem of attempting to ensure that the SCs are satisfied by separating the evaluation of fitness and infeasibility.

In scheduling aircraft landings the SCs may be conveniently expressed mathematically as

$$t_j - t_i \geq S_{ij}, \text{ if } t_i < t_j,\ i,j \in P, i \neq j$$

i.e. that the difference in the SLT's is at least S_{ij} if i lands before j ($t_i \leq t_j$). Given a BA solution we can compute the SLT's t_i(i=1,…,N) as indicated previously. These may, or may not, satisfy the SCs. The quantity

$$\sum_{i=1}^{N} \sum_{\substack{j=1 \\ i \neq j, t_i \leq t_j}}^{N} \max\{0, S_{ij} - (t_j - t_i)\} \qquad (19)$$

will be zero if the SLT's satisfy the SCs and nonzero (strictly positive) if the SLT's do not satisfy the SCs. This quantity is the unfitness associated with any BA solution and is a measure of constraint infeasibility. Informally the higher the unfitness value the more infeasible a solution.

In this paper we will try to apply CBA in such as optimization of a function. This ALP was chosen according to different factors such as representation of the problem can be applied more efficiently. Furthermore, this problem chosen since its own a high complexity (the size and the shape of the search space), which its, cannot be solved using traditional known searches, like ESM.

The fitness rating helps the CBA for ALP achieve solving by awarding scores according to the number of times.

Nevertheless, the fitness rating of BA does correlate well with the correctness of a bee. To show this we applied the following steps:

1. Find the SR and the calculate matrix A of the ALP for N aircraft.
2. Check the generated CBA (π_k), if its satisfies the SR and A(i,j)=1, where i,j$\in\pi_k$ or equal the variable $a_{ij}\in\{0,1\}$ then added to the population (Π), $\pi_k\in\Pi$, otherwise (where A(i,j)$\neq$1), ignore and generated another one or swapping all the pairs (i,j), i,j$\in$P, (i,j)$\notin$U and make $\forall$(i,j)$\in$U to guarantee that π_k satisfies SR.
3. After be certain that π_k satisfies SR by, we apply MPIA to find the sequence $\sigma(t_i)=(t_1,t_2,\ldots,t_N)$ which satisfies the SC.
4. Lastly, we compute the fitness $Z(\sigma(t_i))$.

From above, we suggest to make a hybrid between the CBA and the new heuristic MPIA, we called the suggested algorithm by Improvement CBA (ICBA)

For **parameters selection**, as the goal of this study is to verify the impact of the choice of social topologies in the behavior of the algorithm, the tuning parameters are fixed. They are set to the values that are widely used by the community and that are deemed the most appropriate ones. Table.10 shows the different parameters which are used in solving ALP.

Table.10: Parameters selection of ICBA to solve ALP.

Parameters	Symbol	Value
Number of planes	N	≥ 5
Population size (Number of Bees)	NB	10–20
Number of selected sites	m	3 – 5
Number of elite sites out of m selected sites	E	2
Number of bees for elite sites	Nep	5
Number of bees other selected points	Nsp	3
Number of Generations	NG	≥ 100

We demonstrate some related subalgorithms in order to implement ICBA on ALP:

1. **Subalgorithm Initialization** (Start)

```
{Initialize population with random solutions}
FOR k=Start : NB
    πk = RANDOM(1..n);
    IF πk satisfies SR THEN
        πk∈Π ;        { Sequencing }
    END;
END;
```

CALL MPIA (Π) { *Scheduling* }

2. **Subalgorithm CAL-Fitness**

{*Calculate fitness function for vector* π_k}

FOR k=1: NB

$$Fit_k(t_k,\pi_k)=f(\pi_k)=\sum_{i=1}^{N}(g_i\alpha_i+h_i\beta_i)(\pi_k);$$

END;

3. **Subalgorithm** Improve-Keys(np,me)

{*Improve each solution* Key_i *for m sites*}

FOR k=1: me

FOR j=1: np

r=**RANDOM**(n-1);

New_π_k =**SWAP**(r,r+1);

New_Fit_k(New_t_k,New_π_k)=*f* (New_π_k)**;**

IF New_Fit_k > Fit_k **THEN**

Fit_k = New_Fit_k; π_k = New_π_k;

END;

END;

END;

Fig.5 shows an algorithm description of solving ALP using ICBA.

Algorithm ICBA
Step (1): **READ** ALP information,
READ n, NB, m, e, nep, nsp, NG.
{*BA parameters*}
Step (2): Find DR of ALP.
Step (3): **CALL Initialization (1).**
{*population of Bees*}
Step (4): **REPEAT**
Step (5): **CALL CAL-Fitness.**
Step (6): **CHOOSE** Best m fitness Keys.
Step (7): **CALL Improve-Keys** (nep,e);
CALL Improve-Keys (nsp,m-e).
Step (8): **CALL Initialization** (m+1).
UNTIL (Reach NG).
Step (9): **OUTPUT**: Best Fit(t_i,π).

Fig.5: ICBA algorithm.

2. Experimental Results of Solving ALP using ICBA

In this section, a number of experimental results are presented which outline the effectiveness of ICBA described above. Firstly, the best way to illustrate the ICBA in operation is to look at the development of the population over time. We apply ICBA for any chosen N, we have population of NB random Π N-sequence, in chosen iterations, we choose a sequence (π) with best fitness in the population. Table.11 shows the decrease of fitness value till approach the best fitness using NG=100, NB=20 for N=8 for one run.

Table.11: The decrease of fitness value for N=8 using ICBA.

Iter.	**Best π in iter.**	**t_i**	**Fitness**	**CPU/s**
1	4, 1, 8, 7, 2, 6, 3, 5	105,128,143,151,194,209,217,225	10270	0.04
3	3, 4, 6, 8, 5, 7, 1, 2	98,106,120,128,136,144,159,258	1420	0.31
4	3, 4, 7, 5, 8, 6, 1, 2	98,106,124,132,140,148,163,258	1160	0.36
6	3, 4, 5, 8, 6, 7, 1, 2	98,106,119,127,135,143,158,258	690	0.46
7	3, 4, 5, 6, 8, 7, 1, 2	98,106,123,132,140,148,163,258	470	0.51
8	3, 4, 5, 6, 7, 8, 1, 2	98,106,122,130,138,146,161,258	420	0.55

The shaded cell means this π is the best. Fig.6 shows the decreasing slope of the curve of the fitness values for N=8 using ICBA.

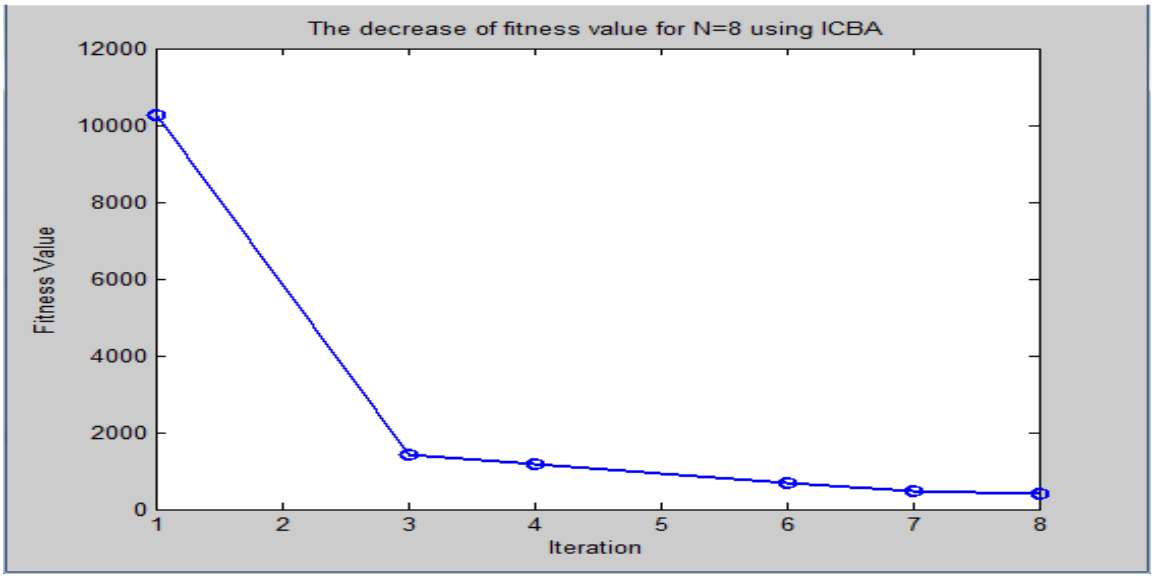

Fig.6: the decrease of fitness values for N=8 using ICBA.

In table.12 we will show the implementation of ICBA to solve ALP. We used NG=100, NB=20 for average of (5) runs for different N.

Fig.7 shows the optimal values and the values obtained by the MPIA and ICBA.

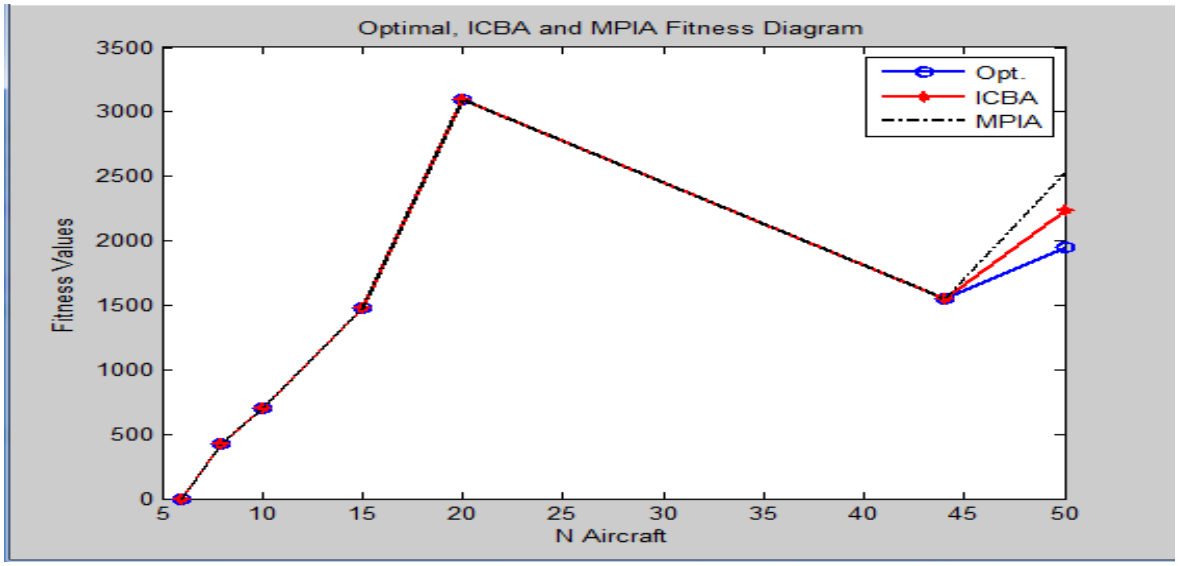

Fig.7: the optimal values and the values obtained by the MPIA and ICBA.

Table.12: the implementation of ICBA to solve ALP for different N.

N	Sequencing (Best π)	Scheduling (Best t_i)	CPU/s	Fitness		
				ICBA	MPIA	Opt.
5	3,4,5,1,2	98,106,123,155,258	0.2	0	0	0
6	3,4,5,6,1,2	98,106,123,135,155,258	0.1	0	0	0
7	3,4,5,6,7,1,2	98,106,123,132,140,155,258	0.34	150	150	150
8	3,4,5,6,7,8,1,2	98,106,122,130,138,146,161,258	0.55	420	420	420
9	3,4,5,6,7,8,9,1,2	98,106,122,130,138,146,154,169,258	0.5	620	620	620
10	3,4,5,6,7,8,9,1,10,2	98,106,118,126,134,142,150,165,180,258	0.6	700	700	700
15	2;3,4,5,6,8,7,9,10,14,13,1 ,2,12,11,15	90,98,106,114,122,130,138,151,171,181, 196,250,313,339,342	0.35	1480	1480	1480
20	2;1,6,8,4,12,10,9,11,19,20, 3,2,7,15,5,18,14,13,17,16	82,100,108,116,124,132,140,149,160,169, 184,197,229,258,261,287,316,335,338,409	0.4	820	820	820
20	1,2,5,9,8,6,7,13,16,12,19, 18,17,15,3,4,10,14,11,20	82,90,98,106,114,122,130,138,146,154, 162,170,178,186,201,270,280,291,295,357	0.3	2520	2520	2520
20	3,4,5,8,6,7,9,10,14,19,17,13, 18,20,1,2,12,15,11,16	82,90,98,106,114,122,130,138,146,154, 162,170,178,186,201,246,280,301,307,393	0.19	3100	3100	3100
30	1,2,3,4,5,6,7,8,9,10,11, 12,13,14,15,16,17,18,19, 20,21,22,23,24,25,26,27, 28,29,30	0,96,192,392,464,560,760,832,1032, 1112,1184,1280,1461,1591,1671,1751, 1831,1903,1999,2180,2252,2348,2576, 2656,2728,2928,2998,3098,3170,3266	0.65	24442	24442	24442
44	1,2,3,4,5,6,7,8,9,10,11, 12,13,14,15,16,17,18,19, 20,21,22,23,24,25,26,27, 28,29,30,31,32,33,34,35, 36,37,38,39,40,41,42,43,44	0,96,296,376,456,528,624,720,920,992, 1192,1264,1464,1544,1616,1816,1896,1968, 2168,2248,2320,2416,2616,2688,2888,2968, 3048,3128,3200,3296,3496,3568,3768, 3840,3936,4136,4216,4296,4368,4464, 4560,4656,4752,4952	1.72	1550	1550	1550
50	1,6,8,4,12,10,9,11,3,19,20, 2,7,15,5,24,18,14,13,23,17, 50,26,25,43,16,35,22,27,44, 45,49,28,32,29,33,47,34,37, 38,48,21,30,39,46,31,36,40, 41,42,82,100,	108,116,124,132,135,149,152,160,175,197, 229,258,261,269,287,316,325,333,341, 345,378,389,392,397,412,427,430,441, 456,471,498,512,516,524,527,556,564, 571,579,619,634,654,673,688,717,725, 740,763	41.45	**2235**	**2530**	**1950**

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

Conclusion:

In this paper, we proposed, first a modification on PIA to obtain new HA called modified parallel improving algorithm MPIA which gives better results from PIA. MPIA is proposed for scheduling aircraft LTs on a single runway from an order determined by a PR. We compared several PR's for test our heuristic. Secondly, we adapted the Bee algorithm to our problem in the single runway case, we have combined with the MPIA in the assignment of aircraft LTs level.

From table.12, all fitness values for different N, all the values are coincidence accept for N=50 the fitness value obtained from ICBA is (2235) which is near the optimal, and its better than the fitness value obtained from MPIA, this mean that the sequence π obtained from ICBA is not one of the PR's mentioned in subsection (*V.B*).

Table.12 shows a remarkable improvement in results after the incorporation of the heuristic algorithm (MPIA) with the CBA.

Our algorithms have been tested on instances available online http://people. brunel. ac.uk/mastjjb~/jeb/info.html,

involving 10 to 50 aircraft for single runway. We compared our results with optimal solutions obtained by ILOG Cplex.

It is of interest to see if some of the results that we have described in this paper can be used to solve the practical problem of scheduling planes in a "dynamic" ATC horizon, in which some planes land (and, hence, depart from the ATC horizon) and new planes appear (into the ATC horizon) at a steady rate.

References

[1] Min Wen, "Algorithms of Scheduling Aircraft Landing Problem", Master thesis, Department of Informatics and Mathematical Modeling, Technical University of Denmark, (2005).

[2] Beasley, J. E., Sonander, J. and Havelock, P., "Scheduling Aircraft Landings at London Heathrow using A Population Heuristic", Journal of the Operational Research Society (2001) 52, 483-493, (2001).

[3] Balakrishnan H. and Chandran B., "Scheduling Aircraft Landings under Constrained Position Shifting", AIAA Guidance, Navigation and Control Conference and Exhibit, Keystone, Colorado, USA, August 21-24, (2006).

[4] Ashraf A., Michael P. and Marco C., "Bees Algorithm", Manufacturing Engineering Center, Cardiff University, Wales,UK, (2009).

[5] Pham D. T., Ghanbarzadeh A., Koc E., Otri S., and Zaidi M. "The Bee's Algorithm – a Novel Tool for Complex Optimization Problems". In: Pham D.T., Eldukhri E., Soroka A. J. ed(s) 2nd Virtual International Conference on Intelligence Production Machines and Systems (IPROMS 2006). Elsevier, Oxford, pp 454-459, (2006).

[6] Bencheikh Gh., Boukachour J. and EL Hilali Alaoui A., "Improved Ant Colony Algorithm to Solve the Aircraft Landing Problem", International Journal of Computer Theory and Engineering, Vol. 3, No. 2, ISSN: 1793-8201, (April 2011).

[7] Beasley, J. E., Krishnamoorthy M., Sharaiha Y. M. and Abramson D., "Scheduling Aircraft Landings—The Static Case", Transportation Science, INFORMS 0041-1655 /00/ 3402-0180 $05.00, Vol. 34, No. 2, pp.180–197, 1526-5447 electronic ISSN, (May 2000).

[8] Abdul-Razaq T. S. and Ali F. H., "Constructing of an Artificial Neural Networks to Minimize Total Completion Time and Total Tardiness", IOSR Journal of Mathematics (IOSR-JM) e-ISSN: 2278-3008, p-ISSN:2319-7676. Volume 10, Issue 2 Ver. VI, PP 25-37, (Mar-Apr. 2014).

[9] Bojanowski L., Harikiopoulo D. and Neogi N., "Multi-Runway Aircraft Sequencing at Congested Airports", 2011 American Control Conference on O'Farrell Street, San Francisco, CA, USA, (June 29 - July 01, 2011).

[10] De Neufville R. and Odoni A. R., "Airport Systems: Planning, Design, and Management", Mc Graw-Hill, (2002).

[11] Venkatakrishan C. S., Barnett A., and Odoni A. R., "Landings at Logan Airport: Describing and Increasing Airport Capacity," Transportation Sciences, vol. 27, pp. 211–217, (1993).

[12] Beasley E., Krishnamoorthy M., Sharaiha Y. M., and Abramson D., "Dynamically Scheduling Aircraft Landings—the Displacement Problem", Working Paper, available from the first author at The Management School, Imperial College, London, England, (1995).