Harmony Search Algorithm for the Multiple Runways Aircraft Landing Scheduling Problem

Omar Salim Abdullah [1, 2], Salwani Abdullah[1], Hafiz Mohd Sarim[1]
[1] *Centre for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia.*
[2]*University of Diyala, College of Pure Science. Iraq.*
omarsalim@siswa.ukm.edu.my

***Abstract*— This paper proposes a Harmony Search (HS) algorithm to solve the multiple runways aircraft landing scheduling (ALS) problem. ALS is a combinatorial optimization problem that has been recognized as an NP-hard problem. It deals with assigning landing times and runways for a set of arrival aircrafts. Each aircraft has its predefined target landing time within a time window, and a separation time between each successive pairs of aircrafts. The objective of ALS problem is to minimize the deviation from the target landing time of each aircraft subject to a set of constraints. The performance of the proposed algorithm is evaluated on thirteen benchmark instances ranging from 10 to 500 aircrafts, and 1 to 5 runways. The results show that the proposed algorithm works considerably well on small-sized instances.**

***Index Terms*—Aircraft Landing Scheduling Problem; Combinatorial Optimization Problem; Harmony Search Algorithm**

I. INTRODUCTION

According to the aviation reports, there is a great increase in the use of air transportation over the last years. Thus, air transportation industries facing the problem of airspace congestion in the terminal area especially in busy airports. The increase of the number of passengers using air transportation on one hand and the fixed number of airports or runways in the airports on the other hand necessitates the discovery of an efficient method to deal with congestion problems. Air traffic controllers (ATC) have the responsibility of giving the landing order of the arrival aircrafts. Usually, they use the first come first save (FCFS) strategy to generate the landing schedule. FCFS is a fair strategy but it is not efficient when air space is congested. Therefore, ATC needs to look for sufficient strategies for managing the aircraft landing scheduling (ALS) problem. The ALS problem deals in generating a landing schedule for a set of aircrafts by assigning landing time and specific runway for each aircraft, by taking into consideration the safety constraints between each successive pair of aircrafts. The goal is to minimize the total deviation from the predefined target landing time and thus increasing the runway throughput.

Different optimization approaches have been used to tackle the ALS problem. Exact methods provide optimal solution in many research papers with a small instance of the problem [1]. While with large instances, exact methods are not the suitable choice due to the time consumption growing exponentially with the instances size. Metaheuristics represent the suitable choice in this class of problem. Different metaheuristic algorithms have been proposed for the ALS problem in literature. For example, scatter search [2], iterated local search [3] hybrid particle swarm optimization with local search [4] and hybrid simulated annealing with variable neighborhood search [5]. These approaches show a better performance on a number of instances.

In this work, we propose a harmony search (HS) algorithm to tackle the ALS problem. HS is a well-known algorithm and widely used to solve optimization problems [6-11]. Harmony Search (HS) algorithm is a metaheuristic search algorithm inspired from the musical performance by [12], where musicians cooperate in tuning the pitches of their music tools to generate a euphonious harmony. HS is a population based stochastic algorithm with a simple design, clear concept and few parameters. Due the efficiency and simplicity of HS, it has been adopted in several optimization applications such as engineering [13], scheduling [11]; [14], clustering [15] and so many others [16]. To the best of our knowledge, HS has not been used in ALS problem. This motivates us to investigate the performance of the HS algorithm in ALS problem.

The structure of this paper is organized as follows: Section 2 presents the problem description and formulation of the ALS problem. The fundamentals of the Harmony Search (HS) algorithm is described in Section 3, followed by the algorithm for ALS in Section 4, experimental results in Section 5, and conclusion and future work in section 6.

II. PROBLEM DESCRIPTION AND FORMULATION

The problem description employed in this work is adopted from [3]. The input for the ALS problem can be stated as below:

n: the number of the arrival aircrafts.
m: the number of runways
S_{ij}: the separation time ($S_{ij} > 0$) between aircrafts i and j if they are assigned to the same runway.
S_{ij}: the separation time ($Sij = 0$) between aircrafts i and j if they are assigned to the different runways.
T_i: the preferred landing time (target time) of aircraft i.
E_i: the earliest landing time of aircraft i.
L_i: the latest landing time of aircraft i.
$C1_i$: the incurred penalty per unit of time for late landing of aircraft i.
$C2_i$: the incurred penalty per unit of time for early landing of aircraft i.

The decision variables used as follows:
x_i: the assigned landing time of aircraft i ($i = 1, 2... n$).
y_{ij}: equal to 1 if aircraft i is assigned to land before aircraft j. Otherwise equals 0.
y_{ir}: equal to 1 if aircraft i is scheduled to land on a runway r ($r = 1,2,..., m$). Otherwise equals 0.
δ_{ij}: equal to 1 if aircrafts i and j are scheduled to land on the same runway. Otherwise equals 0.
a_i: the tardiness of landing when aircraft i is scheduled to land after the target time, $a_i = \max(0, x_i - T_i)$.
b_i: the earliness of landing when aircraft i is scheduled to land before the target time, $b_i = max(0, T_i - x_i)$.

The objective function is to minimize the total penalty incurred from landing before or after the target time formulated as follows:

$$Minf = \sum_{i=1}^{n} (aiC1 + biC2) \quad (1)$$

This is subject to the following:
Time window: landing time for aircraft i must be with its time window:

$$Et_i \le x_i \le Lt_i \qquad i = (1, 2 .., n) \quad (2)$$

Separation time: for aircrafts i and j land on the same runway, there is at least Sij time that must be considered between them:

$$(xj - xi) \ge Sij \qquad i,j = 1,2,\ldots,n \quad i \ne j \quad (3)$$

The priority of landing for each successive pairs of aircrafts i and j, either i or j lands first:

$$y_{ij} + y_{ji} = 1 \quad (4)$$

Each aircraft allocated to only one runway:

$$\delta_{ij} \ge y_{ir} + y_{jr} - 1, \quad i,j = 1,2,\ldots,n \quad i \ne j, \quad r=1,2,\ldots,m \quad (5)$$

Each aircraft must be allocated to only one runway:

$$\sum_{r=1}^{m} yir = 1 \qquad i = 1, 2..., n \quad (6)$$

Note that, if two aircrafts land on different runways, no separation time is required between them.

III. Fundamentals of the Harmony Search (HS) Algorithm

HS algorithm is a population-based algorithm found by Geem and Kim [12]. The inspiration of the HS algorithm mimics a team of musicians in generating pleasing harmony by interacting the harmonies of their music tools. The quality of the resulted harmony is determined by the esthetic standard. For the optimization process, HS algorithm can be represented as an optimization technique to generate (near) optimal solutions determined by the objective function.

Assume that there are three musicians playing different musical instruments, such as a saxophone, a double bass, and a guitar. Each musical instrument represents a decision variable in the optimization problem as (x1, x2,x3). Each pitch generated by any of these musical instruments has a specific range. As example the musical instruments is the saxophone with the range = {La, Si, Do} corresponding to the decision variable x1 = {1, 2, 3}, the double bass with the range = {Do, Re, Mi} corresponding to the decision variable x2 = {3, 4, 5} and the guitar with the range = {Mi, Fa, Sol} corresponding to the decision variable x3 = {5, 6, 7}. In the optimization process, each musician represents a decision variable. Therefore, if La is selected by the saxophone, Do is selected by the double bass and Mi is selected by the guitar, the new harmony will be (La, Do, and Mi). This new harmony is evaluated by esthetic standards. Similarly, in the optimization process, if the values of the decision variables selected are 1, 3 and 5 respectively, then the new solution will be represented as (1, 3 and 5). This new solution will be evaluated by the objective function as a maximization or minimization function according to the optimization problem in hand.

The general scheme of HS algorithm described in Figure 1. HS consists of five steps, as follows [18]:
Step 1: Parameters initialization,
Step 2: Harmony memory (HM) initialization and evaluation,
Step 3: Improvise new harmony,
Step 4: HM update,
Step 5: Termination Criterion.

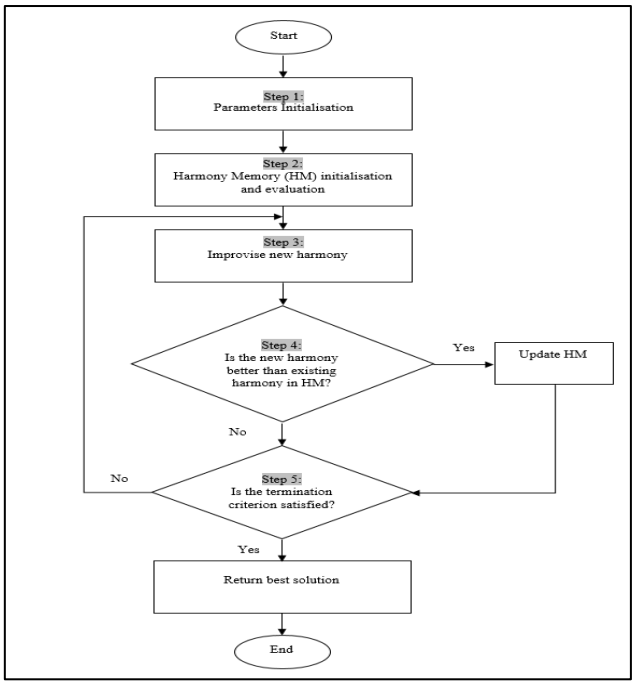

Figure 1: General Scheme of a HS Algorithm

Step 1: Initializing the algorithm parameters
The parameters of HS algorithm are initialized. HS algorithm has the following parameters:

- Harmony Memory Size (HMS): This parameter is equal to the population size in the metaheuristic algorithm. In the HS algorithm, a number of solutions is stored in the harmony memory (HM). The HM represented in our work is a two-dimensional matrix (as in Figure 2) where

the rows represent the solutions and the columns represent the decision variables. The size of the row and the column are equal to HMS and the decision variables of the problem in hand, respectively. $f(x^1)$, $f(x^2)$..., and $f(x^n)$, as shown in Figure 2 represents the deviation from a preferred target time of each aircraft based on Eq. (1).

$$HM=\begin{vmatrix} x_1^1 & x_1^2 & \cdots & x_1^{n-1} & x_1^n & f(x^1) \\ x_2^1 & x_2^2 & \cdots & x_2^{n-1} & x_2^n & f(x^2) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{HMS}^1 & x_{HMS}^2 & \cdots & x_{HMS}^{n-1} & x_{HMS}^n & f(x^n) \end{vmatrix}$$

Figure 2: HM Representation [15]

- Harmony Memory Consideration Rate (HMCR): This parameter is used in the improvisation process to determine whether the value of decision variable of a new harmony will be selected from HM, or it will be generated at random from the possible range between [0, 1]. The probability of randomly selecting the decision variable value from the possible range is given as 1-HMCR.
- Pitch Adjustment Rate (PAR): This parameter is used to decide whether to maintain or to modify the value of the selected decision variables from HM to its neighboring value. The value of PAR is between [0,1]
- Number of Improvisations (NI): This parameter represents the number of iterations of the HS algorithm.

Step 2: HM initialization
This step represents the generation of a population of solutions. The number of solutions in this population is equal to the HMS. The size of the solution is equal to the size of the decision variables and evaluated by the objective function f(x). The solution is generated at random by assigning random values of each decision variable from the range bounded by the lower and upper bounds as;

$$x_i^j = Lx_i + rand[0,1]*(Ux_i - Lx_i) \quad (7)$$

where rand returns a random number between [0,1], Uxi and Lxi are the upper and lower bounds of the decision parameter, respectively.

Step 3: Improvising new harmony
In this step, a new harmony (solution) Xnew is generated according to the following procedure:
Assuming k is the number of the decision variables. Solution X in HM can be represented as X = {x1, x2, x3,..,xk}. To generate a new harmony Xnew = {x1new, x2new, x3new,.., xknew }, the improvisation process will undergo to the following procedure:

- Generate a random number between [0,1].
- If the generated random number is less than HMCR, the decision variables of the new solution Xnew will be assigned from the HM. Otherwise, it will be randomly assigned from its range according to the probability of 1-HMCR.
- The decision variables assigned according to the HMCR will be adjusted according to the value of PAR. The adjusting process starts by generating a random number between [0,1]. If the generated random number is less than PAR, the decision variables assigned by HMCR will be adjusted. Otherwise, the decision variables will not change.

Step 4: Updating the HM
The new harmony (generated in Step 3) will replace the worst harmony in the HM if its fitness is better than the worst harmony.

Step 5: Termination criteria
The process of HS algorithm will terminate when the maximum number of improvisation NI is reached. Otherwise, repeat Step 3 and Step 4.

IV. THE ALGORITHM

This section presents the initial solution construction, and the application of the HS on the ALS problem.

A. Initial Solution Construction

In the ALS problem, a solution is represented as a sequence of landing aircrafts, landing times and runways allocated for all the aircrafts in the sequence. The initial solution algorithm is described in Figure 3.

Initial solution construction

1. *Let n: number of the aircrafts, m: number of runways*
2. *Let S_{ij}: separation time between aircraft i and aircraft j, x_i : landing time of aircraft i*
3. *Let T: target landing time $\{T_1,T_2,...,T_n\}$*
4. *Generate m sequences $\{seq_1, seq_2, ..., seq_m\}$ // sequence for runway*
5. *Randomly select one seq for each runway*
6. ***If** there is an aircraft a is not assigned to runway*
7. *Add a to d1 // d1 ϵ m*
8. ***Else**, return $seq_1, seq_2,.., seq_m$*
9. ***Endif***
10. *Sort $seq_1, seq_2,.., seq_m$ according to their target landing time T in ascending order. $\{T_1<T_2<T_3..., T_n\}$*
11. ***For** i=1 to n do*
12. ***If** $T_a \leq T_b + S_{ab}$ // a,b ϵ n*
13. *Set $x_b = T_b$*
14. ***Else***
15. *$x_b = T_a + S_{ab}$*
16. ***Endif***
17. ***Endfor i***
18. *Return the generated solution.*

Figure 3: Initial Solution Construction

The algorithm starts by assigning the values for the number of the aircrafts (n), number of runways (m) and target landing time (Tn). Then, we divide the aircrafts into a number of sequences equal to the number of runways. For each runway d, randomly select one sequence, seq, of aircrafts. If there is a modulus or remainder from the division, we add it to the first runway (d1). Next, aircrafts are sorted in ascending order based on their target landing time. For each aircraft i, assign landing time (x) based on their target landing time (T) with respect the separation time constraint. Note that, in assigning the landing time (x) for each aircraft, the separation time between each successive pair of aircrafts (Sij) must be respected in order to maintain the feasibility of the solution.

Thus, the landing time (x) for each aircraft equals to its target landing time (T) if it does not violate the separation time constraint. Otherwise, the landing time (x) is equal to the target landing time (T) plus the separation time (of the preceding aircraft) and end the for loop statement. Finally, return the generated initial solution. Figure 4 shows the representation of the solution. The first row represents the sequence of aircraft. The landing time for each aircraft is represented in the second row. Each aircraft is allocated to a specific runway as in the third row. Based on Figure 4, a sequence of three aircrafts (denoted as A1, A2 and A3) are assigned to land on Runway 1. Landing time for aircraft A1 is 150, landing time for aircraft A2 is 160 and landing time of A3 is 782. The same procedure is repeated for the other sequences of the aircrafts.

Aircraft id	A1	A2	A3	…	...	...	An
Landing time (*x*)	150	160	782	…	...	...	..
Runway no	Runway1			Runway...			Runway *m*

Figure 4: Initial Solution Representation

For a better understanding, here we give an example how the assignment of aircrafts to runways is carried out. Assume that there are 10 aircrafts and 3 runways. Firstly, we divide the number of aircrafts with the number of runways (i.e., 10/3). The result of this division is three aircrafts for each runway, and there are 1 remaining aircraft which has not yet been assigned. Thus, the remaining aircraft will be assigned to Runway 1. In case there are 2 remaining aircrafts, each of them will be assigned to Runway 1 and Runway 2.

B. HS Algorithm for ALS Problem

In this section, we describe the implementation of the HS algorithm in solving the ALS problem as in Figure 5.
Firstly, the HS parameters are initialized, and an empty solution S^i is generated. Improvising the new solution is based on the values of the HMCR and PAR as follows: The decision variables of the ALS problem which are the landing time *x* and the runways number *m* for each aircraft. Based on HMCR and the random number generated as in the third step of the HSA in the section 3, the improvisation process will be repeated up to the maximum number of improvisation NI that identified in Step 1. According to the number of the decision variables, random number *r*1 will be generated between [0,1]. If the value of the random number *r1* is less then HMCR, the decision variables will be selected from the HM. Next, random number *r2* will generated between [0,1]. If *r2* is less than PAR, the selected decision variables according to HMCR will be adjusted i.e., swap two aircrafts that are randomly selected and generate a new landing time for them. Note that one swap operation is carried out when decision variables need to be adjusted. Otherwise, no change will be made. The HM is updated if the new solution is better than the worst solution in the HM. The procedure continues until the stopping criterion, which is the number of improvisations is met.

HS algorithm for ALS

1. *Input: population of solutions, HMCR, PAR, x, m // x: landing time of aircraft, m: the total number of runways,*
2. ***for*** *i = 1 to NI*
3. $S^i = \Phi$
4. ***for*** *j = 1 to the number of decision variables*
5. *r1 = uniform random number between [0,1]*
6. ***if*** *r1 < HMCR*
7. S^i_j *= randomly select aircraft p and assign landing time x and runway m from HM // p ϵ n*
8. *r2= uniform random number between [0,1]*
9. ***if*** *r2 < PAR*
10. S^i_j *= randomly select aircrafts p1,p2 based on the probability of PAR + bw*
11. *swap [p1, p2]*
12. *assign new landing time p1(x), p2(x)*
13. ***else*** *maintain [p1,p2] in same order*
14. ***endif***
15. ***else*** *select x, m from the range with the probability of [1-HMCR]*
16. ***endif***
17. *Add* S^i_j *to HM*
18. ***endfor***

Figure 5: Improvising New Solution in ALS Problem on HS Algorithm

V. EXPERIMENTAL RESULTS

Benchmark instances, parameter setting and experimental results are presented in this section. The proposed method is coded in the Java programming language and run on a personal computer with 3.4 GHz Core i5 CPU and 4 GB of RAM

A. Benchmark Instance

We used 13 well-known instances (with 49 instance id) to verify the performance of the proposed methods that we introduced by [17] and is available in OR Library and can be freely download from OR-library ([http://people.brunel.ac.uk/~mastjjb/jeb/orlib/airlandinfo.html] **.** The number of aircrafts ranges from 10 to 500 and the number of runways ranges from 1 to 5. There are two categories of datasets i.e., small instance (Airland1 to Airland8) and large instance (Airland9 to Airland13).

B. Parameters Setting

There are five parameters that need to be determined in advance. We conduct some preliminary tests to set the parameter values for the HMCR, PAR, HMS and NI, where the values of all parameters are determined one by one manually, by changing one value while fixing the rest. The values of the parameters of HS algorithm are presented in Table 1.

Table 1
The Parameter Settings of the HS Algorithm

No	Parameter	Suggested Value
1	HMCR	0.95
2	PAR	0.5
3	HMS	20
4	NI	1000
5	bw	1

C. Result and Comparisons

The performance of the HS algorithm is compared with the best-known solutions reported by the state of the art algorithms for ALS instances. The algorithms in comparison are as presented in Table 2.

The results are presented in Table 3 in terms of the percentage gap from the best known values in the literature (BKV) which is calculated as follows: Δ(%)=((B BKV)/BKV)) *100, where B is the best result returned by the tested algorithm over 30 runs and the BKV results are taken from [11]. Δ(%) = 0 means the obtained solution is equal to the BKV, -Δ(%) indicates the obtained solution superior than the BKV, and +Δ(%) indicates that the obtained solution is inferior than the BKV. The number of aircrafts (N) and the number of runways (m) in each instance reported in Table 3. In the table, the best results obtained by the compared algorithms are indicated in bold. From the results reported in Table 3, it can be observed that:

- Comparing the results of the HS algorithm with BKV, HS algorithm matches the best-known results on small-sized instances (10 out of 49 instances id) i.e., instance# 3,5,6,8,9,12,13,17,20 and 25.
- Comparing the results of the HS algorithm with other algorithms in comparison also show similar results in which it works considerably well on small-sized instances.
- The HS algorithm shows weak performance on large-sized instances in comparison with best-known solutions and other compared algorithms. This is expected, since in this work a basic HS algorithm is employed without any modification in comparison with other algorithms that include an improvement mechanism on the basic algorithm. For example, local search is hybridized with the particle swarm optimization in HPSO-LS, different perturbation operators are employed in the iterated local search by [3], hybridization between simulated annealing and variable neighbourhood descent and variable neighbourhood search in [5]

The performance of the HS algorithm possibly can be further improved by considering the following:
Employ an online/adaptive instead of offline parameter tuning, control the diversity of the solution in the population during the optimisation process and possibly and hybridise with a local search.

Table 2
State of the Art Algorithms in Comparison

#	Algorithm Symbol	References	Description
1	SS	[11]	Scatter search algorithm with linear objective function
2	ILS	[12]	An iterated local search algorithm with multiple perturbation operators and time varying perturbation strength
3	HPSO-LS	[2]	Hybrid particle swarm optimization algorithm in a rolling horizon
4	SA-VND	[13]	Simulated annealing with variable neighborhood descent
5	SA-VNS	[13]	Simulated annealing with variable neighborhood search

VI. CONCLUSIONS AND FUTURE WORK

The aircraft landing scheduling problem has been studied in this work. A Harmony Search (HS) algorithm has been proposed to solve the problem. Numerical experiments on 13 well-known datasets ranged from 10 to 500 aircrafts, and from 1 to 5 runways have been presented to show the effectiveness of the HS in producing a minimum deviation from the target landing time. In addition, the results showed that HS is competitive in terms of Δ **(%)** when compared to best known results on small-sized instances. For future work, we intend to investigate an adaptive mechanism to control parameters of the proposed algorithm. We believe that better solutions can be obtained if the parameter is adaptively modified during the optimization process.

ACKNOWLEDGMENT

This work was supported by the Ministry of Higher Education, Malaysia (FRGS/1/2015/ ICT02/UKM/01/2), and the Universiti Kebangsaan Malaysia (DIP-2016-024).

Table 3
The Computational Results of HS Algorithm Compared to the State of the Art Algorithms

Instance name	Instance no.	N	m	BKV	**HS**	**HPSO-LS**	**ILS**	**SS**	**SA-VND**	**SA-VNS**
					Δ (%)	**Δ (%)**	**Δ (%)**	**Δ (%)**	**Δ (%)**	**Δ (%)**
Airland1	1.	10	1	700	72,857.14	**0**	**0**	**0**	**0**	**0**
	2.		2	90	33,333.33	**0**	**0**	**0**	**0**	**0**
	3.		3	0	**0**	**0**	**0**	**0**	**0**	**0**
Airland2	4.	15	1	1,480	37,162.16	**0**	**0**	**0**	**0**	**0**
	5.		2	210	**0**	**0**	**0**	**0**	**0**	**0**
	6.		3	0	**0**	300	**0**	**0**	100	100
Airland3	7.	20	1	820	179.27	**0**	**0**	**0**	**0**	**0**
	8.		2	60	**0**	16,667	**0**	**0**	16.66	16.66
	9.		3	0	**0**	100	**0**	**0**	100	100
Airland4	10.	20	1	2,520	77,777.78	**0**	**0**	**0**	**0**	**0**
	11.		2	640	6.25	3,125	**0**	**0**	3.12	3.12
	12.		3	130	**0**	23,076.90	**0**	**0**	23.07	27.02
	13.		4	0	**0**	300	**0**	**0**	100	100
Airland5	14.	20	1	3,100	129,677.40	**0**	**0**	**0**	**0**	**0**
	15.		2	650	67,692.31	**0**	**0**	**0**	**0**	**0**
	16.		3	170	41,176.47	**0**	**0**	**0**	**0**	**0**
	17.		4	0	**0**	300	**0**	**0**	100	100
Airland6	18.	30	1	24,442	23,774.65	**0**	**0**	**0**	**0**	**0**
	19.		2	554	18,050.54	**0**	**0**	**0**	**0**	**0**
	20.		3	0	**0**	**0**	**0**	**0**	**0**	**0**
Airland7	21.	44	1	1,550	3,264.78	**0**	**0**	**0**	**0**	**0**
	22.		2	0	2820	**0**	**0**	**0**	**0**	**0**
Airland8	23.	50	1	1,950	116,923.10	**0**	**0**	52.05	**0**	**0**
	24.		2	135	18,518.52	**0**	**0**	**0**	**0**	**0**
	25.		3	0	**0**	100	**0**	**0**	100	100
Airland9	26.	100	1	5,611.70	147.84	**0**	**0**	30.06	8.55	8.55
	27.		2	452.92	120,096.30	**-1,947.36**	-1.74	5.67	**-0.58**	**0**
	28.		3	75.75	245,610.60	0	**-2.31**	**0**	**0**	**0**
	29.		4	0	909.20	**0**	**0**	**0**	**0**	**0**
Airland10	30.	150	1	12,329.31	168,350.90	**-0.30**	-0.06	44.96	**0**	**0**
	31.		2	1,288.73	127,812.70	11,227.332	**-1.37**	7.87	**-5.39**	**0**
	32.		3	220.79	411.64	**-6,576.38**	-9.41	8.88	**-6.49**	**0**
	33.		4	34.22	1,456.08	3,097.60	**-6.16**	16.74	3.09	3.09
	34.		5	0	3,499.40	0.6	**0**	**0**	100	100
Airland11	35.	200	1	1,2418.32	129,977.20	0	**-0.05**	17.95	**0**	**0**
	36.		2	1,540.84	179,751.30	**-13,624.39**	-8.49	9.19	**-8.04**	**0**
	37.		3	280.82	576,946.10	**-9,881.77**	-3.46	21.59	**-2.81**	**0**
	38.		4	54.53	1.529.195	0	**-6.47**	2.77	**0**	**0**
	39.		5	0	5.642.3	**0**	**0**	**0**	**0**	**0**
Airland12	40.	250	1	16,209.78	120,460.20	**-0.54**	**0**	22.15	**0**	**0**
	41.		2	1961.39	159,539.40	**-13,550.08**	**0**	18.80	**0**	**0**
	42.		3	290.04	719,004.30	**-23,469.18**	**-6.21**	17.48	**-3.56**	**0**
	43.		4	3.49	34,595.13	**-30,085.96**	**-2.57**	271.63	**0**	**0**
	44.		5	0	7,770.80	**0**	**0**	**0**	**0**	**0**
Airland13	45.	500	1	44,832.38	105,281.70	**-17,327.37**	**-7.70**	3.24	**-7.54**	**0**
	46.		2	5,501.96	157.074.20	**-28,727.94**	**-0.79**	3.72	**-0.47**	**0**

47.	3	1,108.51	470,884.30	**-39,211.19**	**0**	1.98	**-32.79**	**0**
48.	4	188.46	2.105.232	**-52,271.04**	**-50.71**	22.98	**-46.62**	**0**
49.	5	7.35	40,371.84	**-100**	**-59.18**	**0**	**-48.16**	**0**

REFERENCES

[1] Ernst, A.T., Krishnamoorthy, M. and Storer, R.H., 1999. Heuristic and exact algorithms for scheduling aircraft landings. *Networks*, *34*(3), pp.229-241.

[2] Pinol, H. and Beasley, J.E., 2006. Scatter search and bionomic algorithms for the aircraft landing problem. *European Journal of Operational Research*, *171*(2), pp.439-462.

[3] Sabar, N.R. and Kendall, G., 2015. An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem. *Omega*, *56*, pp.88-98.

[4] Girish, B.S., 2016. An efficient hybrid particle swarm optimization algorithm in a rolling horizon framework for the aircraft landing problem. *Applied Soft Computing*, *44*, pp.200-221.

[5] Salehipour, A., Modarres, M. and Naeni, L.M., 2013. An efficient hybrid meta-heuristic for aircraft landing problem. *Computers & Operations Research*, *40*(1), pp.207-213.

[6] Jaddi, N.S. and Abdullah, S., 2017. A cooperative-competitive master-slave global-best harmony search for ANN optimization and water-quality prediction. *Applied Soft Computing*, *51*, pp.209-224.Moh, O. et al., 2011.

[7] Zeng, B. and Dong, Y., 2016. An improved harmony search based energy-efficient routing algorithm for wireless sensor networks. *Applied Soft Computing*, *41*, pp.135-147.

[8] Zheng, Y.J., Zhang, M.X. and Zhang, B., 2016. Biogeographic harmony search for emergency air transportation. *Soft Computing*, *20*(3), pp.967-977.

[9] Wang, G.G., Gandomi, A.H., Zhao, X. and Chu, H.C.E., 2016. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Computing*, *20*(1), pp.273-285.

[10] Sun, W. and Chang, X., 2015. An improved harmony search algorithm for power distribution network planning. *Journal of Electrical and Computer Engineering*, *2015*, p.5.

[11] Gao, K.Z., Suganthan, P.N., Pan, Q.K., Chua, T.J., Cai, T.X. and Chong, C.S., 2016. Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. *Journal of Intelligent Manufacturing*, *27*(2), pp.363-374.

[12] Geem, Z.W., Kim, J.H. and Loganathan, G.V., 2001. A new heuristic optimization algorithm: harmony search. *Simulation*, *76*(2), pp.60-68.

[13] Fesanghary, M., Mahdavi, M., Minary-Jolandan, M. and Alizadeh, Y., 2008. Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer methods in applied mechanics and engineering*, *197*(33), pp.3080-3091.

[14] Yuan, Y., Xu, H. and Yang, J., 2013. A hybrid harmony search algorithm for the flexible job shop scheduling problem. *Applied Soft Computing*, *13*(7), pp.3259-3272.

[15] Moh'd Alia, O., Al-Betar, M.A., Mandava, R. and Khader, A.T., 2011, December. Data clustering using harmony search algorithm. In *International Conference on Swarm, Evolutionary, and Memetic Computing* (pp. 79-88). Springer Berlin Heidelberg.

[16] Geem, Z.W., 2010. State-of-the-art in the structure of harmony search algorithm. In *Recent Advances In Harmony Search Algorithm* (pp. 1-10). Springer Berlin Heidelberg.

[17] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M. and Abramson, D., 2000. Scheduling aircraft landings—the static case. *Transportation science*, *34*(2), pp.180-197.

[18] Al-Betar, M.A. and Khader, A.T., 2009, August. A hybrid harmony search for university course timetabling. In *Proceedings of the 4nd multidisciplinary conference on scheduling: theory and applications (MISTA 2009), Dublin, Ireland* (pp. 157-179).

[19] Turky, A.M. and Abdullah, S., 2014. A multi-population harmony search algorithm with external archive for dynamic optimization problems. *Information Sciences*, *272*, pp.84-95.