Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 742653, 8 pages
http://dx.doi.org/10.1155/2013/742653

Research Article

A Hybrid Metaheuristic for Multiple Runways Aircraft Landing Problem Based on Bat Algorithm

Jian Xie,[1] Yongquan Zhou,[1,2] and Hongqing Zheng[1]

[1] *College of Information Science and Engineering, Guangxi University for Nationalities, Nanning, Guangxi 530006, China*
[2] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning, Guangxi 530006, China*

Correspondence should be addressed to Yongquan Zhou; yongquanzhou@126.com

Received 17 May 2013; Accepted 11 July 2013

Academic Editor: Xin-She Yang

The aircraft landing problem (ALP) is an NP-hard problem; the aim of ALP is to minimize the total cost of landing deviation from predefined target time under the condition of safe landing. In this paper, the multiple runways case of the static ALP is considered and a hybrid metaheuristic based on bat algorithm is presented to solve it. Moreover, four types of landing time assignment strategies are applied to allocate the scheduling time, and a constructed initialization is used to speed up the convergence rate. The computational results show that the proposed algorithm can obtain the high-quality and comparable solutions for instances up to 500 aircrafts, and also it is capable of finding the optimal solutions for many instances in a short time.

1. Introduction

Airport runway scheduling optimization is an ongoing challenge for air traffic controllers. The increasing demand is a challenge, which leads to many airports and routes are congested. Although investment in infrastructure may increase capacity at airports, it is an effective solution to improve planning and scheduling on current infrastructure. The aircraft landing Problem (ALP) is a kind of typical NP-hard problem in airport runway scheduling optimization [1]. The ALP consists of determining an optimal schedule of landing aircrafts on runways and assigning the landing time of each arriving aircraft. The objective is to minimize the total cost of landing deviation from predefined target time under the condition of safe landing. A predefined time window and separation time requirements with other aircraft must meet.

The first-come first-served (FCFS) is one of the most common solving methods for ALP, especially, in the single runway situation. A detailed review of published work addressing the ALP can be found in [1–3]. Generally speaking, for ALP, those methods can be broadly classified into three categories: exact methods (e.g., dynamic programming [4], branch-and-bound [1], and branch-and-price [5]), queueing theory [6], and heuristic or metaheuristic. Metaheuristic includes genetic algorithms [7], ant colony optimization [8], simulated annealing [9], scatter search and bionomic algorithms [2], cellular automata optimization [10], and other hybrid metaheuristics [9, 11]. Briskorn and Stolletz resent integer programming models an aircraft landing problems with aircraft classes [12].

Recently, more and more metaheuristics inspired by nature or social phenomenon are proposed and these algorithms are increasingly applied to different fields. The bat algorithm (BA) is one of the most popular algorithms, which is inspired by the intelligent echolocation behavior of microbats when they are foraging [13]. Many researchers applied BA to solve various optimization problems. For example, Gandomi et al. focus on solving constrained optimization tasks [14]. Yang and Gandomi apply BA to solve many global engineering optimizations [15]. Mishra at al. use BA to update the weights of a functional link artificial neural network classifier, a model proposed for classification [16]. Meanwhile, some researchers have improved BA and applied it to various optimization problems. Xie et al. proposed a bat algorithm based on differential operator and the Lévy flights trajectory (DLBA) to solve function optimization and nonlinear equations [17]. Wang et al. proposed a new bat algorithm with mutation (BAM) to solve the uninhabited combat air vehicle (UCAV) path planning problem [18]. In this paper, the multiple runways case of the static ALP is considered; a hybrid metaheuristic based on bat algorithm (HBA, for short) is presented to solve it.

The rest of this paper is organized as follows. Section 2 presents the mathematical model of ALP and original bat algorithm. Hybrid metaheuristic based on bat algorithm is proposed in detail to solve multiple runways aircraft landing problem in Section 3. The experimental results of the HBA and comparisons with other previous algorithms are shown in Section 4. In the last section, we conclude this paper and point out some future work.

2. Problem Definitions and Bat Algorithm

2.1. The Mathematical Formulation of ALP. The ALP aims at finding the best arrangement of sequences, runway, and corresponding landing time for a given set of landing aircraft to minimize total cost by following separation requirements. To illustrate the mathematical formulation, some notations and decision variables are defined as follows.

Notations:

n = the number of planes;

m = the number of runways;

S_{ij} = the separation time (≥ 0) between plane i landing and plane j landing, (where the planes i and j land on the same runways), $i \neq j \in \{1, 2, \ldots, n\}$;

s_{ij} = the separation time ($0 \leq s_{ij} \leq S_{ij}$) between plane i landing and plane j landing, (where the planes i and j land on the different runways), $i \neq j \in \{1, 2, \ldots, n\}$;

T_i = the target landing time (target time) of plane i, $i \in \{1, 2, \ldots, n\}$;

E_i = the earliest landing time of plane i, $i \in \{1, 2, \ldots, n\}$;

L_i = the latest landing time of plane i, $i \in \{1, 2, \ldots, n\}$;

$\vec{c}_i$ = the cost of late landing of plane i, $i \in \{1, 2, \ldots, n\}$;

$\overleftarrow{c}_i$ = the cost of early landing of plane i, $i \in \{1, 2, \ldots, n\}$.

Decision variables:

t_i = the scheduled landing time of plane $i, i \in \{1, 2, \ldots, n\}$;

$$y_{ij} = \begin{cases} 1, & \text{if plane } i \text{ lands before plane } j \\ 0, & \text{otherwise,} \end{cases} \quad i \neq j \in \{1, 2, \ldots, n\};$$

$$\gamma_{ir} = \begin{cases} 1, & \text{if plane } i \text{ lands on runway } r \\ 0, & \text{otherwise,} \end{cases} \quad i \in \{1, \ldots, n\}, r \in \{1, \ldots, m\};$$

$$\delta_{ij} = \begin{cases} 1, & \text{if planes } i \text{ and } j \text{ land on the same runway} \\ 0, & \text{otherwise,} \end{cases} \quad i \neq j \in \{1, 2, \ldots, n\};$$

$$\begin{aligned} \vec{\tau}_i &= \text{the delay of landing plane } i \\ &\quad (\text{landing after the target time}); \\ &\quad \text{thus } \vec{\tau}_i = \max(0, t_i - T_i); \\ \overleftarrow{\tau}_i &= \text{the earliness of landing plane } i \\ &\quad (\text{landing before the target time}); \\ &\quad \text{thus } \overleftarrow{\tau}_i = \max(0, T_i - t_i). \end{aligned} \tag{1}$$

The mathematical formulation of a mixed-integer programming for this problem is as follows [1, 2, 9]:

$$\min \quad Z = \sum_{i=1}^{n} \left(\vec{\tau}_i \vec{c}_i + \overleftarrow{\tau}_i \overleftarrow{c}_i \right) \tag{2}$$

$$\text{s.t.} \quad E_i \leq t_i \leq T_i, \; i = 1, 2, \ldots n; \tag{3}$$

$$y_{ij} + y_{ji} = 1, \; \forall i, j \in \{1, 2, \ldots, n\}, i \neq j; \tag{4}$$

$$\begin{aligned} &t_j \geq t_i + S_{ij}\delta_{ij} + s_{ij}\left(1 - \delta_{ij}\right) - M\delta_{ij}; \\ &M = \left(L_i + \max\left(S_{ij}, s_{ij}\right) - E_j\right), \; \forall i \neq j \in \{1, 2, \ldots, n\}; \end{aligned} \tag{5}$$

$$\sum_{r=1}^{m} \gamma_{ir} = 1, \; \forall i \in \{1, 2, \ldots, n\}; \tag{6}$$

$$\delta_{ij} \geq \gamma_{ir} + \gamma_{jr} - 1, i \neq j \in \{1, 2, \ldots, n\}, \; r \in \{1, 2, \ldots, m\}; \tag{7}$$

$$\begin{aligned} &t_i - T_i = \vec{\tau}_i - \overleftarrow{\tau}_i, \; i = 1, 2, \ldots, n; \\ &0 \leq \vec{\tau}_i \leq T_i - E_i, \; \vec{\tau}_i \geq T_i - t_i, \; i = 1, 2, \ldots, n; \\ &0 \leq \overleftarrow{\tau}_i \leq L_i - T_i, \; \overleftarrow{\tau}_i \geq t_i - T, \; i = 1, 2, \ldots, n; \end{aligned} \tag{8}$$

$$\begin{aligned} &y_{ij}, \gamma_{ir}, \delta_{ij} \in \{0, 1\}, i \neq j \in \{1, 2, \ldots, n\}, \; r \in \{1, 2, \ldots, m\}; \\ &t_i, \vec{\tau}_i, \overleftarrow{\tau}_i \geq 0, \; i = 1, 2, \ldots n. \end{aligned} \tag{9}$$

Objective function (2) minimizes the total cost of landing deviation from target time. The constraints (3) ensure that the scheduled landing time of each aircraft lies within its time window; constraints (4) consider the landing order; either aircraft i or j must land first; separation constraints must be ensured by constraints (5), where the role of M is to ensure that the equation is redundant if j lands before i; the constraints (6) ensure that each aircraft should land on only one runway; when aircrafts i and j are assigned to land on the same runway, constraints (7) ensure that the runways assigned to aircrafts i and j are identical; additional constraints (8) are introduced in order to link $\vec{\tau}_i$ and $\overleftarrow{\tau}_i$ to the decision variable t_i; constraints (9) ensure that the decision variables y_{ij}, γ_{ir}, and δ_{ij} are binary and the decision variables t_i, $\vec{\tau}_i$, and $\overleftarrow{\tau}_i$ are nonnegative, respectively.

2.2. Bat Algorithm. The basic bat algorithm (BA) is a metaheuristic proposed by Yang in 2010 [13]. Under several ideal rules, the BA has the following steps.

Step 1. Initialize the bat population and other parameters, and these initial individuals are evaluated.

Step 2. Each bat individual randomly selects a certain frequency f_i of sonic pulse, and the position of bat individual is updated according to their selected frequency. The formulas are as follows:

$$\begin{aligned} f_i &= f_{\min} + \left(f_{\max} - f_{\min}\right)\beta, \\ v_i^t &= v_i^{t-1} + \left(x_i^t - x_*\right) f_i, \\ x_i^t &= x_i^{t-1} + v_i^t, \end{aligned} \tag{10}$$

where x_i^t and v_i^t represent the positions and velocities of individual i in a D-dimensional search space at generation t, $\beta \in [0,1]$ is a random vector drawn from a uniform distribution, and x_* is the current global best location (solution) which is located after comparing all the solutions among all the n bats. Meanwhile, these new individuals are evaluated.

Step 3. If a random number is greater than its pulse emission rate R, then a new position is generated around the current global best position for each individual, which is the equal to local search. The updating formula of local search adopts $x = x_* + \varepsilon \times L_t$, where $\varepsilon \in [-1,1]$ is a random number and $L_t = \langle L_i^t \rangle$ is the average loudness of all the bats at current generation t.

Step 4. If the local search is effective and its loudness L is greater than a random number, then the new position is accepted and its pulse emission rate R and loudness L are updated, where pulse emission rate is increased and loudness is decreased. L_i and R_i are updated by $L_i^{t+1} = \alpha \times L_i^t$, $R_i^{t+1} = R_i^0 \times [1 - \exp(-\gamma \times t)]$, where α, γ are constants.

Step 5. If the termination criterion is met, then the algorithm stops; otherwise repeat algorithm (go to Step 2).

In general, the bat algorithm has three procedures, position updating, local search, and decreasing the probability of local search. For the details of BA refer to [13].

3. Hybrid Metaheuristic Based on Bat Algorithm for ALP

Basic bat algorithm is a continuous optimization algorithm, which is successfully applied to solve real optimization problem [15, 17]. However, the standard continuous encoding scheme of BA cannot be used to solve ALP directly. Therefore, in order to solve aircraft landing problem effectively, HBA is proposed.

3.1. Solution Representation in HBA. In order to apply BA to ALP, the first step is to devise a suitable representation of the candidate solutions for this particular problem. Each individual is a sequence S ($S = (s_1, s_2, \ldots, s_n)$) with integer number s_i ($s_i \in \{1, \ldots, m\}$), where the integer number represents the runway and the length of this sequence S is the number of aircrafts. For example, if we have three runways and ten aircrafts, the coded individual with integer is $1 \to 3 \to 2 \to 2 \to 3 \to 2 \to 1 \to 2 \to 1 \to 3$ Figure 1 shows that the aircrafts 1, 7, and 9 land on runway number 1, the aircrafts 3, 4, 6 and 8 land on runway number 2, the aircrafts 2, 5, and 10 land on runway number 3.

Aircraft:	1	2	3	4	5	6	7	8	9	10
Runway:	1	3	2	2	3	2	1	2	1	3

Figure 1: The representation of the candidate solutions.

3.2. Landing Time Assignment. The assigned landing time (ALT) of each aircraft is the very important step in the ALP; the purpose is to reduce the total cost of penalty caused by all aircrafts. In this paper, the landing time is assigned based on the target landing time T_i assigned to each aircraft. There are four types of assignment strategies carried out in the proposed algorithm: forward assignment strategy (FAS), backward assignment strategy (BAS), random forward assignment strategy (RFAS), and random backward assignment strategy (RBAS). For each kind of assignment strategy, firstly, all aircrafts are found out on each runway, and the target landing time T of these aircrafts is sorted in ascending order, namely, $T_{i_1} \le T_{i_2} \le \cdots \le T_{i_{p-1}} \le T_{i_p}$, where p is the number of aircrafts on this runway.

(i) FAS: for each two sorted aircrafts i_k and i_{k+1} on a runway, the ALT is $\vec{t}_{i_k}$ and $\vec{t}_{i_{k+1}}$, separately. If $T_{i_{k+1}} < \vec{t}_{i_k} + S_{i_k i_{k+1}}$; then $\vec{t}_{i_{k+1}} = \vec{t}_{i_k} + S_{i_k i_{k+1}}$, otherwise $\vec{t}_{i_{k+1}} = T_{i_{k+1}}$.

(ii) BAS: for each two sorted aircrafts i_k and i_{k+1} on a runway, the ALT is $\overleftarrow{t}_{i_k}$ and $\overleftarrow{t}_{i_{k+1}}$, separately. If $T_{i_k} > \overleftarrow{t}_{i_{k+1}} - S_{i_k i_{k+1}}$; then $\overleftarrow{t}_{i_k} = \overleftarrow{t}_{i_{k+1}} - S_{i_k i_{k+1}}$, otherwise $\overleftarrow{t}_{i_k} = T_{i_k}$.

(iii) RFAS: for each two sorted aircrafts i_k and i_{k+1} on a runway, the ALT is $\overset{\frown}{t}_{i_k}$ and $\overset{\frown}{t}_{i_{k+1}}$, separately. $\overset{\frown}{t}_{i_k}$ is rounded down to the nearest integer between $\overleftarrow{t}_{i_k}$ and $\vec{t}_{i_k}$. If $\overset{\frown}{t}_{i_{k+1}} < \overset{\frown}{t}_{i_k} + S_{i_k i_{k+1}}$, then $\overset{\frown}{t}_{i_{k+1}} = \overset{\frown}{t}_{i_k} + S_{i_k i_{k+1}}$.

(iv) BFAS: for each two sorted aircrafts i_k and i_{k+1} on a runway, the ALT is $\breve{t}_{i_k}$ and $\breve{t}_{i_{k+1}}$, separately. $\breve{t}_{i_k}$ is rounded down to the nearest integer between $\overleftarrow{t}_{i_k}$ and $\vec{t}_{i_k}$. If $\breve{t}_{i_k} < \breve{t}_{i_{k+1}} - S_{i_k i_{k+1}}$, then $\breve{t}_{i_k} = \breve{t}_{i_{k+1}} - S_{i_k i_{k+1}}$.

The assigned landing time is effective just while the all constraints (3)–(9) are satisfied, and the four kinds of total cost are compared; the best total cost is used as the objective function value.

3.3. Initialization Construction. The initialization of a population is usually performed by randomly selecting a runway from the available runways for each aircraft. However, the target landing time of aircraft is ordered, and the time window must ensure considering the safety. So, a preprocessing is performed to improve the performance of HBA in the initialization construction.

In initialization, *ps* individuals are generated, where *ps* is population size. For each individual, all target landing time T_i $(i = 1, 2, \ldots, n)$ is sorted in an ascending order; namely, $T_{i_1} \leq T_{i_2} \leq \cdots \leq T_{i_{n-1}} \leq T_{i_n}$. The first aircraft i_1 lands on a runway randomly, for each two sorted aircrafts i_k and i_{k+1}, if $T_{i_{k+1}} < T_{i_1} + S_{i_k i_{k+1}}$, and then aircraft i_{k+1} is allocated to other runway, namely, aircrafts i_k and i_{k+1} on different runways. Otherwise, the aircraft i_{k+1} is allocated to the runway that aircraft i_k lands on. The initialization repeats until the *ps* individuals are generated.

According to the experiments performed, this initialization construction can develop an initial schedule of landing aircrafts which has a very good quality. However, to derive near-optimal solutions, this initial schedule should be improved using the bat algorithm provided in Section 3.4.

3.4. Hybrid Bat Algorithm. In original bat algorithm framework, the idea is that, firstly, the bat individual randomly selects a certain frequency of sonic pulse, and the position of bat individual is updated according to its selected frequency; secondly, if a random number is greater than its pulse emission rate *R*, then a new position is generated around the current global best position for each individual, which is equal to local search; at last, if the local search is effective and its loudness *L* is greater than a random number, then the new position is accepted, and its pulse emission rate *R* and loudness *L* are updated, where pulse emission rate is increased and loudness is decreased. In general, the bat algorithm has three procedures: position updating, local search, and decreasing the probability of local search.

In this paper, the frequency f is a runway, $f \in \{1, \ldots, m\}$. The position updating is different from continuous bat algorithm. The position updating is used to assign the landing sequence, which is performed as follows:

(i) select a frequency f randomly, namely; select a runway randomly;

(ii) select an aircraft randomly on selected runway; then assign this aircraft to other runway.

There is an example used to illustrate the procedure in Figure 2. If the frequency $f = 2$ and the second aircraft is selected, then, this aircraft is assigned to runway number 3.

For the local search part, this procedure is controlled by pulse emission rate *R*. The *R* is equivalent to the probability of performing local search, and the *R* is updated by

$$R(t) = \left(1 + \exp\left(-\frac{5}{t_{\max}} \times \left(t - \frac{t_{\max}}{2}\right)\right)\right)^{-1}, \quad (11)$$

where t denotes the tth generation $t_{\max}$ is the maximal generation. The rate *R* is similar to sigmoid function. The

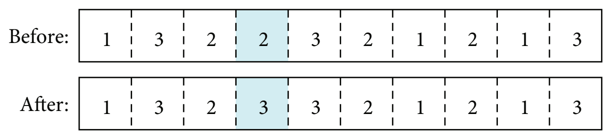

Figure 2: An individual with ten aircrafts before and after position updating.

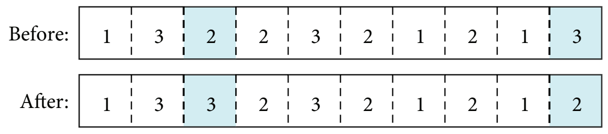

Figure 3: An individual with ten aircrafts before and after swap local search.

purpose of the local search is to enhance the solution generated, and the operation is performed on the current global best individual in bat algorithm. In this paper, two types of local search are presented: the swap and loop subsequence inserting (LSI).

The purpose of swap is mildly mutating the current global best individual, so that an improved solution can be found out around the current optimal solution. The swap is illustrated in Figure 3, where the third and tenth components are randomly chosen to be exchanged; note that the two selected aircrafts land on different runways. Thus, the value of the third component is changed from 2 to 3, while the value of the tenth component is switched from 3 to 2.

The LSI is a variant of inserting operation; the purpose is to mutate the current global best individual in a large extent, so that the diversity of population can be ensured. It can effectively avoid prematurity and greatly improve efficiency of global search. The LSI is illustrated in Figure 4; a start point is randomly chosen (ninth component) and a random length of subsequence is determined (5), so the sub-sequence can be determined $(1 \rightarrow 3 \rightarrow 1 \rightarrow 3)$. A random insert point is chosen in remainder sub-sequence (third, fifth components); then the selected sub-sequence is inserted into remainder sub-sequence before insert point.

The local search systematically explores different neighborhood structures. The swap and LSI are preformed according to the pulse emission rate *R*. In other words, if a random number is greater than the *R*, the swap is performed; otherwise, the LSI is performed.

The loudness L_i of bat individual i determines the accepted probability a solution generated by local search in original bat algorithm. Meanwhile, it also dominates the updating of pulse emission rate *R* and loudness *L* in continuous bat algorithm. However, in this paper, a runway balance (RB) operation is performed according to the value of loudness *L* for each individual.

The intention of RB operation is to balance the load of each runway. Firstly, the aircrafts are counted on each runway; an aircraft selected randomly on runway with maximum aircrafts is assigned to a runway with minimum aircrafts. If the amount of aircrafts on runway (maximum aircrafts or minimum aircrafts) is equal, then a runway is selected

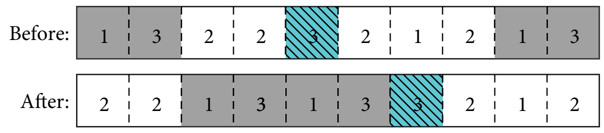

Figure 4: An individual with ten aircrafts before and after LSI local search.

randomly. This process is stopped until the difference of aircrafts amount on each runway is not more than one.

The loudness L_i is updated by (12)

$$L_i^{t+1} = \alpha \times L_i^t, \tag{12}$$

where α is a constant and initial value of $L_i^1 \in (1,2)$. If a random number is less than its loudness L_i^t, then the RB operation is performed; otherwise, each aircraft is assigned to a random runway.

The iterative process is repeated until the termination criterion is met; Algorithm 1 shows the pseudo-code of HBA based on the framework of bat algorithm for ALP. The lines 1–3, the bat population and other parameters are initialized (the initialization of population uses a method described in Section 3.3), and these initial individuals are evaluated (Section 3.2 detailedly described this procedure). In lines 5–9, the individuals are updated by selecting frequency (this part corresponds to the position updating of BA; this method is described in Section 3.4), and these new individuals are evaluated. Lines 10–15 show the local search and these solutions generated by local search are evaluated. In lines 16–24, the runway balance operation is carried out. Lines 25-26 are the judgment of termination criterion and the output of results.

4. Simulation Results and Comparisons

The simulation experiment is extensively investigated by a large number of benchmark instances; these well-studied problems are taken from the web OR-Library (last update: June 2012, http://people.brunel.ac.uk/~mastjjb/jeb/info.html), a reference site which contains detailed information regarding a large number of benchmark instances. In this paper, the whole 13 instances from OR-Library are selected; these instances have been widely used as benchmarks to certify the performance of algorithms by many researchers [2, 9].

All computational experiments are conducted with MATLAB 2012a on a 3.0 GHz Athlon PC with 2.0 GB memory. There are two kinds of termination criterion for different instances. For instances where the value of optimal (exact) solution (V_{opt}) is known, the algorithm is repeated until the objective function value is equal to the V_{opt}; if the objective function value is greater than the V_{opt} when the maximum generation $t_{\max}(= 200)$ is met, then the algorithm also is terminated. For instances where the value of V_{opt} is not known, the termination criterion is set as maximum generation $t_{\max} = 1000$.

4.1. Sensitivity Analysis. Since performance is affected by the settings of the parameter values used in metaheuristics, a sensitivity analysis is conducted to examine the effect of different parameter values on the proposed HBA. Two parameters, that is, population size ps and parameter α, are used to investigate the performance with respect to different values, which are based on the average results obtained from instances involving from 10 to 50 aircrafts, and each instance run ten times. The principal figures of merit for comparison of different parameter values are the average percentage gap G_{avg} (%) associated with the best solution found and the average computational time CT_{avg} in seconds for exact solution is reached. The percentage gap $G(\%)$ is measurement criteria referenced [2]. Figures 5 and 6 show the statistical result, where G_{avg} is shown on the left y-axis and CT_{avg} is shown on the right y-axis and different parameter values (ps and α) are listed in on the horizontal x-axis.

The performance of the HBA in terms of different population numbers is shown in Figure 5. It shows that better results can be obtained with a larger population. However, it does not improve significantly when the value of ps is equal to or greater than 10. On the other hand, the computational time increases steadily when the population rises.

Figure 6 illustrates the performance effect of the HBA with a decreasing α value. It indicates that the average solution quality deteriorates when the value of α decreases gradually. In some cases, increasing the value above 0.9 may actually worsen the average objective value. This is probably because the search is extremely random and RB operation is carried out rarely.

4.2. Comparisons of Results. Based on the results of the sensitivity analysis, the parameter values are set in the proposed HBA (i.e., ps= 10, α= 0.9) for comparison with other algorithms for solving the ALP with multiple runways that were proposed and tested to be valid in previous studies. The computational results obtained are shown in Tables 1 and 2. In Tables 1 and 2, for each problem, the instance (Ins); the number of aircrafts (n); the number of runways (m); the value of optimal solution (V_{opt}); the value of the best-known solution (V_{best}) if the optimal solution is not known; the objective function value (O) obtained and the percentage gap (G%) associated with the best solution found over the 15 replications and the average execution time (T) in seconds for 15 replications. Note that in order to compare the results, the s_{ij} is set zero in accordance with the previous literature, where G is calculated on the basis of V_{opt} or V_{best}; $G = 100 * (O - V_{\text{opt}})/V_{\text{opt}}$. One complication is that for some problem where V_{opt} or V_{best} is zero, the percentage gap is defined as zero if and only if the best solution found as zero is also zero, otherwise it is undefined (nd).

Table 1 presents results for a set of instances involving from 10 to 50 aircrafts. SS is scatter search, the bionomic algorithm is marked as BA1 [2], and IACA is improved ant colony algorithm [8]; heuristic is an effective heuristic algorithm in [1], FCFS is first-come first-served, and the result is reference [2]. According to Table 1, the percentage gap G of SS and BA1 is better than HBA; however, the average

```
(1) Initialize the ps, $t = 1$, bat population and other parameters;
(2) Construct initial bat population;    // (3.3 Initialization construction)
(3) Assign landing time and evaluate each individual;    // (3.2 Landing time assignment)
(4) repeat
(5)     for $i$ = 1:ps do
(6)         Determine frequency $f$;
(7)         Update each bat individual;
(8)     end
(9)     Assign landing time and evaluate each individual;    // (3.2 Landing time assignment)
(10)    if rand > $R(t)$ then
(11)        Carry out swap local search operation;
(12)    else
(13)        Carry out LSI local search operation;
(14)    end
(15)    Assign landing time and evaluate each individual;    // (3.2 Landing time assignment)
(16)    Compute loudness of each individual by (12);
(17)    for $i$ = 1:ps do
(18)        if rand < $L_i$ then
(19)            Carry out RB operation;    // Runway balance operation
(20)        else
(21)            Assign each aircraft to a random runway;
(22)        end
(23)    end
(24)    Assign landing time and evaluate each individual;    // (3.2 Landing time assignment)
(25) until $t = t_{\max}$
(26) Output result and plot
```

Algorithm 1: The pseudo-code of HBA for ALP.

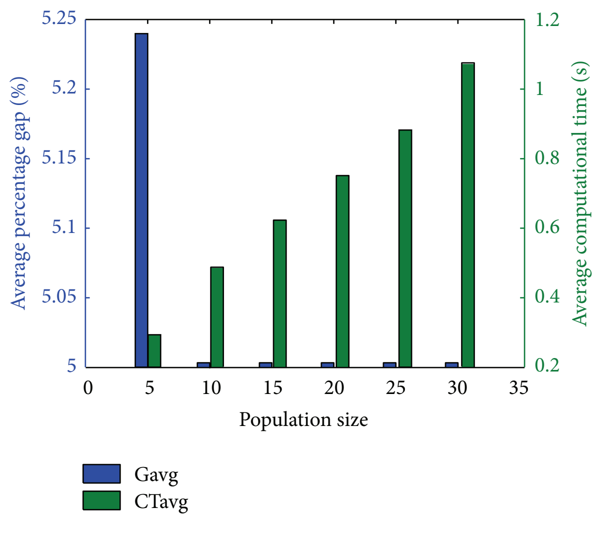

Figure 5: The effect of the value of population size *ps* with respect to relative computational time and average percentage gap.

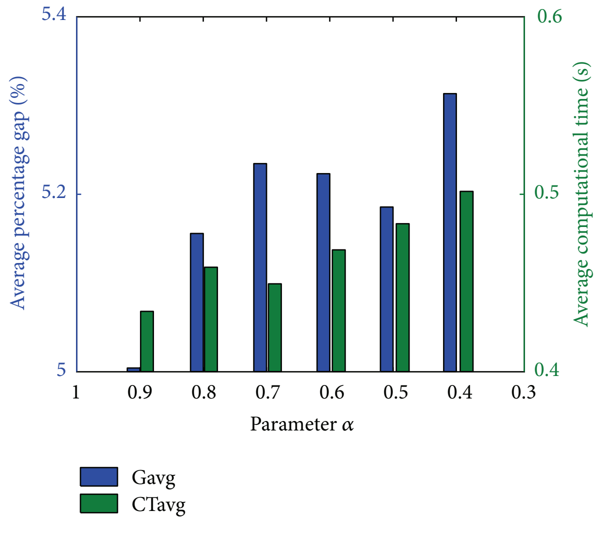

Figure 6: The effect of the value of α with respect to relative computational time and average percentage gap.

execution time T of SS and BA1 is longer than the T of HBA and the average execution time of HBA only expends 0.43 seconds; by contrast, the SS and BA1 expend 6.4 and 7.7 seconds, respectively. The comparison of results between HBA and heuristic shows the solutions obtained by HBA are superior to the solutions obtained by heuristic. Meanwhile, the average execution time is approximate between HBA and heuristic. Comparing with IACA and FCFS, it is evidently shown that the HBA is effective.

Table 2 presents results for a set of instances involving from 100 to 500 aircrafts. The termination criterion is set as maximum generation $t_{\max} = 1000$. The IACA and heuristic did not select these instances for testing. From Table 2, even for the larger instances, the HBA found that optimal solutions

Table 1: The comparisons of computational results for first group instances.

Ins	n	m	V_{opt}	HBA			SS		BA1		IACA		Heuristic		FCFS
				O	G	T	G	T	G	T	G	T	G	T	G
Airland1	10	2	90	90	0	0.08	0	2.4	0	4.5	0	0.1	0	0.1	0
		3	0	0	0	0.11	0	3.9	0	3.4	0	0.0	0	0.1	0
Airland2	15	2	210	210	0	0.09	0	4.5	0	4.9	0	0.6	0	0.1	9.52
		3	0	0	0	0.10	0	4.6	0	4.3	0	0.4	0	0.1	0
Airland3	20	2	60	60	0	0.09	0	4.8	0	5.8	0	11.3	0	0.1	116.67
		3	0	0	0	0.10	0	6.2	0	6.3	0	9.3	0	0.2	nd
Airland4	20	2	640	640	0	0.55	0	5.2	0	5.5	0	7.9	0	0.1	0
		3	130	130	0	0.14	0	4.6	0	5.7	0	6.4	0	0.1	0
		4	0	0	0	0.14	0	5.6	0	5.2	0	5.8	0	0.2	0
Airland5	20	2	650	890	36.92	1.44	0	5.0	3.08	6.1	12.31	2.4	64.62	0.1	16.92
		3	170	170	0	0.16	0	5.4	0	4.3	0	7.0	41.18	0.1	5.88
		4	0	0	0	0.21	0	5.6	0	6.8	0	3.2	0	0.2	nd
Airland6	30	2	554	636	14.80	1.61	0	7.0	3.61	10.1	51.08	3.8	59.21	0.1	59.21
		3	0	0	0	0.30	0	5.4	0	8.7	0	12.2	0	0.2	0
Airland7	44	2	0	0	0	0.09	0	11.8	0	12.4	0	55.2	0	0.2	0
Airland8	50	2	135	180	33.33	2.01	0	12.1	0	19.6	22.22	168.1	88.89	0.2	425.93
		3	0	0	0	0.16	0	13.9	0	18.1	nd	108.2	0	0.6	nd
Average					5.00	0.43	0	6.4	0.39	7.7	5.35	23.6	14.93	0.2	45.30

Table 2: The comparisons of computational results for second group instances.

Ins	n	m	V_{best}	HBA			SS		BA1		FCFS
				O	G	T	G	T	G	T	G
Airland9	100	2	452.92	499.49	10.28	16.0	5.67	24.3	54.73	48.7	172.60
		3	75.75	77.03	1.69	16.5	0	39.0	87.46	46.6	342.81
		4	0	0	0	16.6	0	33.6	nd	43.9	nd
Airland10	150	2	1288.73	1407.45	9.21	18.6	7.87	60.8	25.95	84.5	103.47
		3	220.79	224.13	1.51	21.0	8.88	66.8	195.88	80.3	552.16
		4	34.22	34.22	0	20.6	16.74	64.7	292.40	78.8	3473.49
		5	0	0	0	23.1	0	60.7	nd	76.2	nd
Airland11	200	2	1540.84	1673.95	8.64	23.2	9.19	95.9	38.54	128.7	129.80
		3	280.82	280.64	−0.06	26.1	21.59	102.1	290.09	120.3	764.25
		4	54.53	54.53	0	27.4	2.77	99.3	474.47	116.8	3947.88
		5	0	0	0	27.2	0	95.6	nd	115.8	nd
Airland12	250	2	1961.39	2482.26	26.56	28.3	18.80	126.6	50.18	183.5	137.42
		3	290.04	243.79	−15.95	31.5	17.48	145.4	198.01	171.0	903.97
		4	3.49	2.44	−30.09	33.3	271.63	144.5	13216.91	168.8	70752.44
		5	0	0	0	34.6	0	138.6	nd	166.2	nd
Airland13	500	2	5501.96	5184.06	−5.78	58.0	3.72	383.6	37.47	537.9	56.91
		3	1108.51	755.15	−31.88	60.7	1.98	456.0	182.69	515.8	462.60
		4	188.46	90.03	−52.23	63.7	22.98	441.3	1186.81	497.7	2027.94
		5	7.35	0	−100	65.9	0	442.1	22308.44	488.7	52628.71
Average					−9.37	34.0	21.54	159.0	2576.00	193.2	9097.10

in several cases are prominent. We can clearly find that the percentage deviation G from the best-known solutions is negative, which indicates the solutions found by HBA are better than the best-known solutions. The average percentage gap G of HBA is −9.37% for this group of instances; however, the average percentage gap G of SS, BA1, and FCFS is positive and is much greater than the G of HBA. On the other hand, the average execution time T expended by HBA is much lesser than the average execution time T of SS and BA1 expended.

Figure 7 illustrates the performance of the HBA, the SS, the BA1, the IACA, and the heuristic with respect to the computational time. HBA and heuristic perform almost

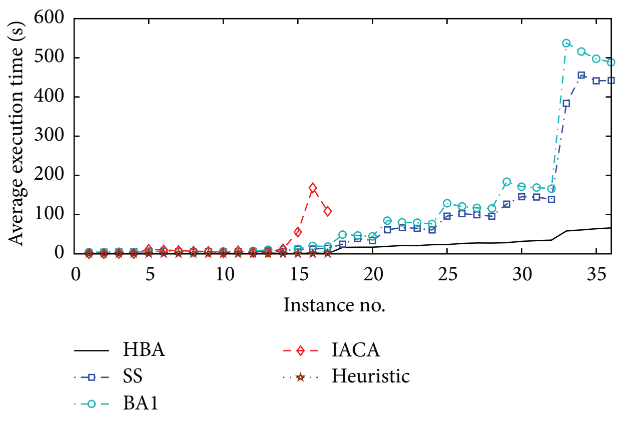

FIGURE 7: The computational time of each test case.

the same on instances up to 50 aircrafts. For the larger instances up to 500 aircrafts, the computational time of HBA is shortened observably, which also demonstrates that the HBA has a faster convergence rate.

5. Conclusions

In this paper, we considered the multiple runways aircraft landing problem with the objective of minimizing the total deviation of landing time from the target time. In order to solve the larger instances involving up to 500 aircraft and multiple runways, a hybrid metaheuristic based on bat algorithm (HBA, for short) has been implemented. The HBA includes a problem-dependent initialization construction, and several local search operations are integrated into the framework of bat algorithm. The computational results of the HBA show that the proposed algorithm is very effective and competitive and can obtain solutions with high quality for instances up to 500 aircrafts in a short time. Moreover, the landing time assignment of each aircraft is a key for solving ALP; several excellent assignment strategies need to be presented in our further work; meanwhile, the aircraft take-off problem (ATP) in airport runway scheduling problem also is our future work.

Acknowledgment

This work is supported by National Science Foundation of China under Grant no. 61165015, Key Project of Guangxi Science Foundation under Grant no. 2012GXNSFDA053028, Key Project of Guangxi High School Science Foundation under Grant no. 20121ZD008, the Funded by Open Research Fund Program of Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China under Grant no. IPIU01201100, and the Innovation Project of Guangxi Graduate Education under Grant no. YCSZ2012063.

References

[1] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson, "Scheduling aircraft landings—the static case," *Transportation Science*, vol. 34, no. 2, pp. 180–197, 2000.

[2] H. Pinol and J. E. Beasley, "Scatter search and bionomic algorithms for the aircraft landing problem," *European Journal of Operational Research*, vol. 171, no. 2, pp. 439–462, 2006.

[3] J. A. Bennell, M. Mesgarpour, and C. N. Potts, "Airport runway scheduling," *Annals of Operations Research*, vol. 204, pp. 249–270, 2013.

[4] A. R. Brentnall and R. C. H. Cheng, "Some effects of aircraft arrival sequence algorithms," *Journal of the Operational Research Society*, vol. 60, no. 7, pp. 962–972, 2009.

[5] M. Wen, *Algorithms of Scheduling Aircraft Landing Problem*, Technical University of Denmark, DTU, Lyngby, Denmark, 2005.

[6] N. Bäuerle, O. Engelhardt-Funke, and M. Kolonko, "On the waiting time of arriving aircrafts and the capacity of airports with one or two runways," *European Journal of Operational Research*, vol. 177, no. 2, pp. 1180–1196, 2006.

[7] X.-B. Hu and E. Di Paolo, "An efficient genetic algorithm with uniform crossover for air traffic control," *Computers and Operations Research*, vol. 36, no. 1, pp. 245–259, 2009.

[8] G. Bencheikh, J. Boukachour, and A. E. H. Alaoui, "Improved ant colony algorithm to solve the aircraft landing problem," *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 224–233, 2011.

[9] A. Salehipour, M. Modarres, and N. L. Moslemi, "An efficient hybrid meta-heuristic for aircraft landing problem," *Computers & Operations Research*, vol. 40, no. 1, pp. 207–213, 2013.

[10] S.-P. Yu, X.-B. Cao, and J. Zhang, "A real-time schedule method for aircraft landing scheduling problem based on cellular automation," *Applied Soft Computing Journal*, vol. 11, no. 4, pp. 3485–3493, 2011.

[11] G. Hancerliogullari, G. Rabadi, A. H. Al-Salem, and M. Kharbeche, "Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem," *Journal of Air Transport Management*, vol. 32, pp. 39–48, 2013.

[12] D. Briskorn and R. Stolletz, "Aircraft landing problems with aircraft classes," *Journal of Scheduling*, pp. 1–15, 2013.

[13] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO '10)*, pp. 65–74, Springer, Berlin, Germany, 2010.

[14] A. H. Gandomi, X. S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, no. 6, pp. 1239–1255, 2013.

[15] X. S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[16] S. Mishra, K. Shaw, and D. Mishra, "A new meta-heuristic bat inspired classification approach for microarray data," *Procedia Technology*, vol. 4, pp. 802–806, 2012.

[17] J. Xie, Y. Zhou, and H. Chen, "A novel bat algorithm based on differential operator and Lévy-flights trajectory," *Computational Intelligence and Neuroscience*, vol. 2013, Article ID 453812, 13 pages, 2013.

[18] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang, "A bat algorithm with mutation for UCAV path planning," *The Scientific World Journal*, vol. 2012, Article ID 418946, 15 pages, 2012.