Omega 56 (2015) 88–98

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem[☆]

Nasser R. Sabar[a,*], Graham Kendall[a,b]

[a] ASAP Research Group, The University of Nottingham Malaysia Campus, Jalan Broga, Semenyih 43500, Selangor, Malaysia.
[b] ASAP Research Group, The University of Nottingham, Nottingham, UK

ARTICLE INFO

Article history:
Received 5 March 2014
Accepted 18 March 2015
Available online 28 March 2015

Keywords:
Aircraft landing problem
Iterated local search
Meta-heuristics
Airport operation

ABSTRACT

Landing aircraft safely is an important operation that air traffic controllers have to deal with on a daily basis. For each arriving aircraft a runway and a landing time must be allocated. If these allocations can be done in an efficient way, it could give the airport a competitive advantage. The Aircraft Landing Problem (ALP) aims to minimize the deviation from a preferred target time of each aircraft. It is an NP-hard problem, meaning that we may have to resort to heuristic methods as exact methods may not be suitable, especially as the problem size increases. This paper proposes an iterated local search (ILS) algorithm for the ALP. ILS is a single solution based search methodology that successively invokes a local search procedure to find a local optimum solution. A perturbation operator is used to modify the current solution in order to escape from the local optimum and to provide a new solution for the local search procedure. As different problems and/or instances have different characteristics, the success of the ILS is highly dependent on the local search, the perturbation operator(s) and the perturbation strength. To address these issues, we utilize four perturbation operators and a time varying perturbation strength which changes as the algorithm progresses. A variable neighborhood descent algorithm is used as our local search. The proposed ILS generates high quality solutions for the ALP benchmark instances taken from the scientific literature, demonstrating its efficiency in terms of both solution quality and computational time. Moreover, the proposed ILS produces new best results for some instances.

1. Introduction

The last decade has seen a growing demand for air transportation [1,2]. This continual increase stretches airport capacity, ultimately leading to airports being unable to cope with future demand. Given the lead time in building a new airport, the controversy that often surrounds new airports (or even new runways), the costs involved and, perhaps, the lack of space to extend means that it may not be possible to simply build our way out of trouble [1,2]. Therefore, the industry has to find other ways to meet the ever increasing demand for air travel. The Aircraft Landing Problem (ALP) is an important element in airport operations [1–4]. Efficient solution methodologies for tackling the ALP is important, both from an economical and environmental perspective [1–4].

The ALP was introduced by Beasley et al. [3]. It assigns each arriving aircraft a runway and a landing time. Each plane is required to land within a given time window. The goal is to minimize the sum of the penalties incurred when an aircraft lands before or after its preferred target time. ALP can be considered as a combination of two sub-problems [5]: a sequencing problem (which decides the sequence of aircraft landings) and a scheduling problem (which decides the landing times for each aircraft in the generated sequence). In addition, Beasley et al. [3] also highlighted that the practical complexities of ALP are mainly due to the inclusion of additional constraints and considerations such as the control, separation times, latest times, runway allocation and additional terms in the objective function. The ALP can be formulated as a machine job scheduling problem with sequence-dependent processing times, where penalties are added for earliness and tardiness violations [5], representing the earliest and latest landing times. ALP is an NP-hard optimization problem, making it more and more difficult to solve to optimality as the problem size increases [2,3]. It may therefore be necessary to utilize heuristic or meta-heuristic approaches [6].

Beasley et al. [3] presented eight small-sized ALP instances that include single and multiple runways and proposed a mixed-integer program to solve them. They also applied a first-come-first-serve (FCFS) rule and obtained the optimal solutions for instances containing up to 50 aircraft. Pinol et al. [6] presented two meta-heuristic algorithms for ALP, scatter search and a bionomic algorithm, as well as large-scaled ALP instances that involve up to 500 aircraft and five runways. Both algorithms were tested on small and large-sized instances and they

[☆] This manuscript was processed by Associate Editor Pesch.
[*] Corresponding author.
E-mail addresses: Nasser.Sabar@nottingham.edu.my (N.R. Sabar), Graham.Kendall@nottingham.edu.my (G. Kendall).

http://dx.doi.org/10.1016/j.omega.2015.03.007
0305-0483/© 2015 Elsevier Ltd. All rights reserved.

managed to find the optimal solutions for instances with up to 50 aircraft. However, the performance markedly decreased as the problem size increased. Salehipour et al. [7] presented a mixed-integer goal programming approach and a hybrid meta-heuristic that combined simulated annealing with variable neighborhood descent and variable neighborhood search for the ALP. The proposed mixed-integer goal programming approach was able to provide optimal solutions for instances with up to 100 aircraft. For larger instances, with up to 500 aircraft, the proposed hybrid meta-heuristic produced good results when compared to existing methodologies. Comprehensive reviews on recent methods in airport runway optimization can be found in [1,2].

Mixed-integer programming formulations and meta-heuristic approaches have been able to find good quality solutions for the ALP, even obtaining optimal solutions for small instances. However, performance deteriorates on larger instances. This suggests that further research on ALP might be useful and may improve on the best known results. In this work, we propose an Iterated Local Search (ILS) algorithm for the ALP. ILS is an iterative, single solution based meta-heuristic [8,9]. ILS extends classical local search algorithms by including a diversification mechanism, the idea being to perform a randomized walk in the neighborhood of the current local optimum to generate a new starting solution instead of generating a new solution from scratch [8]. ILS iteratively invokes a local search procedure (to reach a local optimum solution) and a perturbation operator (to modify the current solution in order to escape from the local optimum) for a predefined number of iterations. Like many other meta-heuristic methodologies [19-22] ILS is based on a simple framework, yet it has been shown to be an effective and efficient solution methodology for many real world problems [8,9].

We were also motivated by the fact that meta-heuristic frameworks are very adaptable, enabling other meta-heuristic algorithms such as variable neighborhood search to be easily hybridized with ILS [10]. In addition, the ILS designer has several degrees of freedom to select the appropriate ILS components such as the local search procedure and the perturbation operator. Blum and Roli [10] pointed out that the perturbation operator has a large influence on the performance of ILS and controlling the perturbation strength is quite important. A small perturbation strength may lead the local search to return to previously visited solutions. If the perturbation strength is too large, this may lead the algorithm to behave as a random restart method, which typically leads to low quality solutions. Lourenço et al. [8] suggested that “A good perturbation transforms one excellent solution into an excellent starting point for a local search”. Consequently, various ILS variants that use different perturbation operators have been proposed. For example, Katayama and Narihisa [11] used 4-opt with a greedy algorithm as a perturbation mechanism, Thierens [12] proposed a perturbation mechanism that utilizes a population of solutions, Katayama and Narihisa [13] used a crossover operator as a perturbation mechanism for ILS and Zhang et al. [14] used the guided mutation operator as perturbation operator in ILS. More details about ILS variants can be found in [8,9,23,24].

Despite ILS producing very good results for various optimization problems, most existing ILS's use a single perturbation operator and the perturbation strength remains the same during the optimization process. In addition, the success of ILS is highly dependent on the employed local search and the type of perturbation operator. This is because different problems and/or instances possess different characteristics, and therefore require different ILS parameters/configurations. To address these issues, in this work, we propose an ILS with the following components:

i) *Local search phase*: the proposed ILS uses variable neighborhood decent (VND) as a local search. VND escapes from the current local optimum by using a set of neighborhood structures that are applied in a systematic way. The idea is that different neighborhood structures generate different search trajectories.
ii) *Perturbation phase*: in this work, we employ multiple perturbation operators and a time varying perturbation strength. We utilize four different perturbation operators, where each one modifies the current local optimum solution. The time varying perturbation strength changes as the algorithm progresses. The idea is to assign a larger perturbation strength in the early stages of the search, in order to focus on exploring the search space. The perturbation strength is gradually decreased so that we gradually focus more on exploitation.

The 13 small and large ALP benchmark instances introduced in [3] are used to demonstrate the effectiveness of our proposed algorithm. An experimental comparison is conducted to evaluate ILS with, and without, the additional components. Our results demonstrate that ILS, with the additional components, produces very good results across all ALP instances. In addition, the proposed ILS finds new best solutions for some ALP instances when compared to the best known results in the scientific literature.

2. Problem description

ALP is a combinatorial optimization problem and can be defined as follows: given a set of arrival aircraft, each one associated with a target landing time, a predefined time window for landing, and a set of runways, the goal is to assign a runway and a landing time for each aircraft with a minimum total cost deviation from the target landing times, while respecting the following constraints:

- Each aircraft is assigned to only one runway.
- A maximum of one aircraft is assigned to a runway at a specific landing time.
- The landing time of each aircraft should be within the aircraft's landing time window.
- The separation time between two aircraft landing on the same runway should be respected.

A penalty is incurred if the aircraft is scheduled to land before or after its target time. The objective is to minimize the overall penalty by generating the best landing sequence and landing time for the given set of aircraft. The following formulation presents the model more formally (adopted from [3]).

Notation:

- n: the number of the arrival aircraft.
- m: the number of runways.
- s_{ij}: the separation time ($s_{ij} > 0$) between aircraft i and j when they are assigned to the same runway.
- t_{ij}: the separation time between aircraft i and j when they are assigned to 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 for late lading of aircraft i.
- $C2_i$: the incurred penalty for early landing of aircraft i.

Decision variables

- x_i: the assigned landing time of aircraft $i (i = 1, 2, \ldots, n)$.
- y_{ij}: equal to 1 if aircraft i is assigned to land before aircraft j. Otherwise it takes 0.
- y_{ir}: equal to 1 if aircraft i is scheduled to land on a runway $r (r = 1, 2, \ldots, m)$. Otherwise, it takes 0.
- δ_{ij}: equal to 1 if aircraft i and j are scheduled to land on the same runway. Otherwise, it takes 0.

- a_i: the tardiness of landing when an aircraft i scheduled to land after the target time, $a_i = \max(0, x_i - T_i)$.
- b_i: the earliness of landing when an aircraft i is scheduled to land before the target time, $b_i = \max(0, T_i - x_i)$.

$$Min\, f = \sum_{i=1}^{n} (a_i C1_i + b_i C2_i) \tag{1}$$

S.t.

$$E_i \le x_i \le L_i, \qquad i = 1, 2, \ldots, n \tag{2}$$

$$(x_j - x_i) \ge s_{ij}\delta_{ij} + t_{ij}(1 - \delta_{ij}) - My_{ji}, \qquad i,j = 1, 2, \ldots, n, \qquad i \ne j \tag{3}$$

$$y_{ij} + y_{ji} = 1, \qquad i,j = 1, 2, \ldots, n, \qquad i \ne j \tag{4}$$

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

$$\sum_{r=1}^{m} y_{ir} = 1, \qquad i = 1, 2, \ldots, n \tag{6}$$

$$y_{ij}, y_{ir}, \delta_{ij} \in \{0, 1\}, \qquad i,j = 1, 2, \ldots, n, \qquad r = 1, 2, \ldots, m \tag{7}$$

$$x_i, a_i, b_i \ge 0, \qquad i = 1, 2, \ldots, n \tag{8}$$

The objective function (1) minimizes the total penalty of landing deviations from the desired landing time. Eq. (2) ensures that each aircraft is scheduled to land within its time window. Eq. (3) ensures that if aircraft i and j are allocated to the same runway, there should be at least s_{ij} time between the landing times of both aircraft, while if they are allocated to different runways, there should be at least t_{ij} time. M in Eq. (3) is used to ensure that the equation is redundant in the case when j lands before i ($y_{ij} = 1$). Eq. (4) considers the scheduling order where either aircraft i or aircraft j must land first. Eq. (5) ensures that when both aircraft i and j are allocated to the same runway, the runways must be identical. Eq. (6) ensures that each aircraft is allocated to only one runway. Eqs. (7) and (8) ensure that the decision variables y_{ij}, y_{ir}, and δ_{ij} are binary, while x_i, a_i, b_i are nonnegative.

3. The proposed method

In this section, we first discuss the basic Iterated Local Search algorithm followed by the proposed method, together with its components.

3.1. Iterated local search (ILS)

Iterated Local Search (ILS) is a single solution iterative improvement based meta-heuristic [9]. ILS starts with an initial solution and successively alternates between three phases: local search, perturbation and an acceptance criterion. The idea of ILS is to use a local search algorithm to find a local optimum solution and a perturbation operator to perturb (modify) the current local optimum in order to escape from the local optimum and to move to another point in the search space. That is, the main principle of ILS is that the local search algorithm can escape from a local optimum by restarting the search using a new perturbed solution instead of generating a completely new solution. ILS has been applied to various optimization problems and has produced very good results when compared to other methods [9]. Algorithm 1 shows the pseudo code of a basic ILS [9].

Algorithm 1. Basic Iterated Local Search (ILS)

```
1    Set MaxIter; Iter ← 0;
2    $S_0$ ← Generate Initial Solution ();
3    $S_1$ ← Local search ($S_0$);
4    while (Iter < MaxIter) do
5        $S_2$ ← Perturbation phase ($S_1$);
6        $S_3$ ← Local search phase ($S_2$);
7        $S_1$ ← Acceptance criterion phase ($S_1$, $S_3$, history);
8        Iter = Iter + 1;
9    end while
10   Return the best solution
```

After setting the maximum number of iterations parameter (*MaxIter*) (line 1) and generating an initial solution (S_0) (line 2), ILS calls the local search algorithm to find a local optimum solution (S_1) (line 3). Next, ILS executes the while loop (lines 4–9) for a fixed number of iterations. Within the while loop, the following are executed: call the perturbation phase (line 5) to perturb the current local optimum solution (S_1) in order to generate a new starting solution (S_2) for the local search phase. Next, call the local search phase to iteratively improve a given solution (S_2) and to find a new local optimum solution (S_3) (line 6). Then, the generated solution by the local search phase (S_3) is either accepted or rejected based on the utilized the acceptance criterion (line 7). Next, update the iteration counter (*Iter*) (line 8) and check the stopping condition. If the stopping condition is satisfied, stop and return the best solution (line 10). Otherwise, start a new iteration.

3.2. The proposed ILS

The proposed ILS follows the general framework of the basic ILS (Section 3.1, Algorithm 1). It starts with an initial feasible solution (Section 3.2.1) and iteratively invokes the local search phase (Section 3.2.2), perturbation phase (Section 3.2.3) and the acceptance criterion phase (Section 3.2.4) for a predefined number of iterations. The main difference between the basic ILS and the proposed ILS is that the proposed ILS employs new elements in each of the three main phases (local search, perturbation and acceptance criterion). The following subsections discuss the initial solution generation and the main components (local search, perturbation and the acceptance criterion) of the proposed ILS.

3.2.1. Initial solution generation method

We propose a Randomized Greedy (GR) heuristic to generate an initial solution for the ILS. It starts by sorting the set of arrival aircraft in ascending order based on their preferred landing time. Then GR randomly selects 10% (this value is determined by preliminary testing) of the sorted aircraft and allocates them runways and landing times. The remaining (90%) aircraft are scheduled in a greedy manner based on their preferred landing time. We use a random allocation to ensure that each call of GR will generate different initial solutions. This helps the proposed ILS by starting with a variety of solutions that are scattered over the search space. It can also help in generating a feasible solution, as we can simply start again if an infeasible solution is generated. The proposed GR algorithm is presented in Algorithm 2.

GR first sets the number of arrival aircraft (n), runways (m) and other indices (lines 1–4). The aircraft are then sorted in ascending order based on their preferred landing time (line 5). Line 6 randomly selects 10% of aircraft, storing them in *RS*. Each aircraft in *RS* is assigned to a random runway (line 8). The safety constraint (separation time) between each pair of aircraft is checked and if it is violated, they are assigned to different runways (lines 9 and 10). Otherwise, they are assigned to the same runway (line 11). Next allocate the landing times for the pair of aircraft that have already been assigned to a runway (line 10) and remove them from *RS* (line 12–13). Once all the aircraft in *RS* have been processed, the aircraft in *SS* are allocated to a runway (lines 16–29) using a greedy procedure. GR will sequentially assign them one by one based on their order in *SS*. At each assignment, the safety constraint between each pair of aircraft is checked and the aircraft are assigned to different runways if there is a violation in the safety constraint (line 20). Otherwise, they are assigned to the same runway (21). Next, assign landing times to the allocated pair of aircraft (line 27) and remove them from *SS* (line 28).

Algorithm 2. The randomized greedy heuristic (GR)

```
1  Let $n$ be the number of aircraft, $m$ the number of runways;
2  Let $s_{ij}$ represents the safety constraint between aircraft $i$
   and aircraft $j$;
3  Let $T_i$ represents the preferred landing time of aircraft $i$,
   $i=1,\ldots,n$;
4  Set $i \leftarrow 0$; It$\leftarrow 0$;
5  Sort the aircraft based on their preferred landing times ($T_i$),
   $i=1,\ldots,n$; and add them to SS$=\{T_1, T_2,\ldots,T_n\}$ where
   $T_1 \le T_2 \le T_3,\ldots, \le T_n$
6  Randomly select 10% from SS and add them to RS
7  while (RS not empty) do
8      Select a runway ($r$) at random
9      For each two aircraft $a$ and $b$ belong to RS
10     If ($T_b \le T_a + s_{ab}$) then assign $a$ to $r$ and $b$ to $r+1$
11     Else assign both aircraft $a$ and $b$ to the same runway, $r$
12     Allocate landing times to $a$ and $b$
13     Remove $a$ and $b$ from RS
14 end while
15 Update SS
16 while (SS not empty) do
17     Let $a$ be the first aircraft and $b$ the second aircraft in SS
18     for $It=1$ to $m$ do
19     If the current runway is empty then
20         If ($T_b \le T_a + s_{ab}$) then assign $a$ to $It$ and $b$ to $It+1$
21         Else assign both $a$ and $b$ to the same runway, $It$
22         Break;
23     else
24         Check the separation time of $a$ and $b$ with those
   already assigned in current runway;
25         If the separation time constraint is not violated
   then assign $a$ and $b$ and break;
26     end for
27     Allocate landing times to $a$ and $b$;
28     Remove the $a$ and $b$ from SS;
29 end while
30 Return the generated solution
```

3.2.2. Local search phase

The algorithm starts with an initial solution generated by the perturbation phase. Then, it iteratively generates a neighborhood solution by modifying the current solution using a predefined neighborhood structure, accepting only improving solutions, for a fixed number of iterations. The steepest decent algorithm has the advantage of being simple to understand, fast and is easy to implement. However, steepest descent algorithms will become trapped in a local optimum. Furthermore, the type of neighborhood structures that are used within the local search algorithm has an impact on the algorithm's performance.

Therefore, we utilize the Variable Neighborhood Descent (VND) algorithm as our local search [15]. VND is a single solution based meta-heuristic that begins with an initial solution and iteratively improves it using a set of neighborhood structures, executed in a predefined sequence. We utilize VND due to its ability to escape from a local optimum and it also uses a set of neighborhood structures to navigate through the search space. Given an initial solution generated by the perturbation phase, VND attempts to improve it by iteratively exploring its neighborhood solutions, for a predefined number of iterations. The sequence of the neighborhood structures is randomly generated and the neighborhood structures are applied in a systematic way. Once the sequence of the neighborhood structures has been generated, VND starts with the first neighborhood structure in the sequence and keeps applying it as long as it yields an improvement of the objective function. If no improvement is obtained, VND will stop using the current neighborhood structure and restart the search using the next neighborhood structure in the sequence. VND will terminate if the last neighborhood structure in the sequence has been applied and no further improvement can be obtained. That is, VND can escape from a local optimum solution by changing the neighborhood structure, as different neighborhoods generate different landscapes, with the local optimum being different for each [15]. We use the following four neighborhoods in our VND algorithm:

- NS_1: randomly select a runway and examine all possible swaps between each pair of aircraft. Only swaps that lead to an improvement are accepted.
- NS_2: randomly select two different runways and examine all possible swaps of aircraft between the selected runways, accepting only improving swaps.
- NS_3: randomly select a runway and move all aircraft that are currently assigned to this runway to a different position on the same runway. Only improving moves are accepted.
- NS_4: randomly select a runway and examine the possibility of moving all aircraft currently assigned to this runway to a different runway. Only improving moves are accepted.

It should be noted that NS_1 and NS_3 attempt to change aircraft landing times, but using the same runway, while NS_2 and NS_4 attempt to change the runways of the scheduled aircraft. Algorithm 3 shows the pseudo code of VND [15].

Algorithm 3. Variable neighborhood decent algorithm (VND)

```
1  Let $K$ be the number of the neighborhood structures (NS)
2  Set $i \leftarrow 1$;
3  $S_0 \leftarrow$ Generate Initial Solution ();
4  while ($i < K$) do
5      $S_1 \leftarrow$ Generates a neighborhood of $S_0$ using $NS_i$;
6      If $f(S_1) < f(S_0)$ then
7          $S_0 = S_1$;
8          $i=1$;
9      else
10         $i=i+1$;
11     end if
12 end while
13 Return the best solution
```

After setting the maximum number of the available neighborhood structures (K) (line 1), the index of neighborhood structures (i) (line 2) and generating an initial solution (S_0) (line 3), the main loop of VND is executed (lines 4–12). At each iteration, generate a neighborhood solution using NS_i (line 5). If the quality of generated neighborhood solution (S_1) by NS_i is better than S_0 (line 6), then replace S_0 with S_1 (line 7) and set $i=1$ (line 8). Otherwise, increase i by one ($i=i+1$) to call the next neighborhood structure in the sequence (line 10). VND will stop the search if the local optimum of the K neighborhood structure cannot be improved any further.

3.2.3. Perturbation phase

The perturbation phase is essential in the ILS as it controls the diversification aspect of the algorithm. It is responsible for providing, for each call of the local search phase, a new starting solution by perturbing (modifying) the current local optimum solution in order to escape from the local optimum and to move the search to a new point in the search space. The success of the local search algorithm depends on the diversification strategy of the overall algorithm. Consequently, the amount of perturbation has a large influence on the performance of the ILS. If it is too small the local search algorithm may return to previously visited solutions, wasting computational resources, and the search may end up cycling between already visited local optimum solutions. If the perturbations are too large, it may lead the algorithm to behave as a random restart method, which typically leads to low quality solutions. Generally, the perturbation size is controlled by two factors: the type of perturbation operator and the perturbation strength (or how many times the perturbation operator should be applied to a given local optimum solution). In this work, we address these issues by proposing multiple perturbation operators and a time varying perturbation strength for the ILS, as follows:

a) *Multiple perturbation operators*: we utilize four different perturbation operators to modify the current local optimum solution. At each call to the perturbation phase we randomly select one of the perturbation operators, and apply it for a number of iterations determined by the perturbation strength. The generated solutions are accepted regardless of their quality as long as there is no violation of the problem constraints. The perturbation operators are:
   i. Move_1: randomly select one aircraft and move it to different position within the same runway.
   ii. Move_2: randomly select one aircraft and move it to a different runway.
   iii. Swap_1: randomly select two different aircraft from the same runway and swap their positions.
   iv. Swap_2: randomly select two different aircraft from different runways and swap their positions.
b) *Time varying perturbation strength*: the perturbation strength controls how many times the perturbation operator should be applied. A large perturbation strength will enable new areas of the search space to be explored. A smaller value will focus on exploiting the current neighborhood of the search space. We utilize a time varying perturbation strength which starts with a high value, and which is gradually reduced as the ILS algorithm progresses. Let *pt* represent how many times the selected perturbation operator should be applied to a given solution. *pt* is calculated as follows:

$$pt = \begin{cases} tv * n & f\ tv \geq 1 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

and

$$tv = (maxtv - mintv)\frac{IterMax - Iter}{IterMax} + mintv \tag{10}$$

where *n* represents the number of aircraft in the problem instance. *tv* is the time varying variable, *mintv* and *maxtv* are constants that represent the minimum and the maximum time variation. *Iter* represents the current iteration of the search and *IterMax* represents the maximum number of iterations. Eq. (10) will be linearly varying based on the current iteration which starts with a large value, and which decreases as the search progress. In Eq. (9), we set $pt = 1$ if *tv* returns a negative value. The minimum number of applications of the selected perturbation is one, whilst the maximum is determined by the value of *tv*. That is, the diversification degree of the proposed ILS will linearly decrease as the search progress.

3.2.4. Acceptance criterion phase

The acceptance criterion phase compares the quality of the final solution generated by the local search phase with the one that has been used as an initial solution and then decides whether to accept or reject the new solution to be used in the next iteration. In this work,

Table 1
The ALP benchmark instances.

Instance name	N	m	Instance nos.
Airland1	10	1	1
		2	2
		3	3
Airland2	15	1	4
		2	5
		3	6
Airland3	20	1	7
		2	8
		3	9
Airland4	20	1	10
		2	11
		3	12
		4	13
Airland5	20	1	14
		2	15
		3	16
		4	17
Airland6	30	1	18
		2	19
		3	20
Airland7	44	1	21
		2	22
Airland8	50	1	23
		2	24
		3	25
Airland9	100	1	26
		2	27
		3	28
		4	29
Airland10	150	1	30
		2	31
		3	32
		4	33
		5	34
Airland11	200	1	35
		2	36
		3	37
		4	38
		5	39
Airland12	250	1	40
		2	41
		3	42
		4	43
		5	44
Airland13	500	1	45
		2	46
		3	47
		4	48
		5	49

Table 2
The parameter settings of the ILS.

Nos.	Parameter	ILS1		ILS
		Tested range	Suggested value	Suggested value
1	*MaxIter* of ILS	10–500	100	150
2	Minimum time varying, *mintv*	0.1–0.3	0.1	0.1
3	Maximum time varying, *maxtv*	0.8–0.9	0.9	0.9
4	The size of *RS* in GR	5–50% of aircraft	10% of aircraft	10% of aircraft

we utilize the exponential Monte Carlo acceptance criterion (EMC) [16]. EMC always accept improving solutions. Worse solutions are also accepted with a probability of $R < \exp(-\delta)$, where R is a random number between [0, 1] and δ is the difference between the quality of the final and initial solutions. The probability of accepting worse solutions will decrease as δ increases [16,25-27].

4. Experimental setup

This section discusses the benchmark instances that have been used to evaluate the proposed ILS and the parameter settings.

4.1. Benchmark instances

The 13 ALP benchmark instances that have been studied in the scientific literature are used to evaluate the performance of the proposed ILS. These instances were introduced by [3] and can be downloaded from the OR-library (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/airlandinfo.html). The main characteristics of these instances are presented in Table 1. The first column represents instance names, the second column shows the number of aircraft (n) in each instance, the third column shows the number of runways (m) and the fourth column represents the instance number. In these instances, the number of aircraft ranges from 10 to 500. Each instance has a different number of runways that range from 1 to 5. Instances Airland1 to Airland8 (instances 1 to 25) are considered small instances. Instances Airland9 to Airland13 (instances 26 to 49) are considered large instances [3].

4.2. ILS parameter settings

The proposed ILS has four parameters that need to be determined in advance. In this work, we have performed two different procedures to set the parameters. In the first procedure, the values of all parameters are determined one by one through manually changing the value of one parameter, while fixing the others (denoted as ILS1).

Table 3
The comparative results between the seven ILS variants.

Instance name	**Instance nos.**	***m***	**BKV**	**ILS**	**ILS1**	**ILS2**	**ILS3**	**ILS4**	**ILS5**	**ILS6**
				Δ(%)	Δ(%)	Δ(%)	Δ(%)	Δ(%)	Δ(%)	Δ(%)
Airland1	1	1	700	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	2	2	90	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	3	3	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Airland2	4	1	1,480	**0**	**0**	0.3	0.8	0.2	1.2	0.7
	5	2	210	**0**	**0**	1.2	**0**	0.7	0.5	0.3
	6	3	0	**0**	**0**	**0**	**0**	0.4	0.7	**0**
Airland3	7	1	820	**0**	**0**	**0**	0.7	0.6	0.3	**0**
	8	2	60	**0**	**0**	**0**	**0**	1.2	0.1	**0**
	9	3	0	**0**	**0**	**0**	0.4	1.04	0.2	0.7
Airland4	10	1	2,520	**0**	**0**	0	0.5	0.9	0.8	**0**
	11	2	640	**0**	**0**	1.6	0.8	**0**	0.6	0.1
	12	3	130	**0**	**0**	**0**	0.2	0.3	0.4	0.3
	13	4	0	**0**	**0**	**0**	**0**	**0**	0.2	0.6
Airland5	14	1	3,100	**0**	**0**	**0**	**0**	0.4	0.6	0.8
	15	2	650	**0**	**0**	**0**	0.1	1. 4	0.2	0.3
	16	3	170	**0**	**0**	1.7	**0**	1.1	1.7	1.2
	17	4	0	**0**	**0**	**0**	0.3	1.6	1.6	**0**
Airland6	18	1	24,442	**0**	**0**	0.6	0	1.2	0.5	1.5
	19	2	554	**0**	**0**	**0**	0	**0**	0.3	1.7
	20	3	0	**0**	**0**	0.7	0.1	0.7	0.4	1.7
Airland7	21	1	1,550	**0**	**0**	**0**	0	0.9	0.6	1.6
	22	2	0	**0**	**0**	1.4	2.01	0.2	0.4	1.3
Airland8	23	1	1,950	**0**	**0**	**0**	0	0.6	1.8	1.6
	24	2	135	**0**	**0**	**0**	1.7	0.7	1.7	0.8
	25	3	0	**0**	**0**	0.5	1.4	0.4	1.2	0.4
Airland9	26	1	5,611.70	**0**	**0**	33.14	41.12	51.27	53.47	55.14
	27	2	452.92	**−1.74**	**−0.83**	18.46	22.73	28.41	31.41	34.12
	28	3	75.75	**−2.31**	**0**	66.12	71.56	84.14	72.48	78.92
	29	4	0	**0**	**0**	27.42	31.75	30.83	28.14	31.21
Airland10	30	1	12,329.31	**−0.06**	**0**	51.74	41.12	37.14	38.56	36.75
	31	2	1,288.73	**−1.37**	**−0.93**	13.73	14.82	19.65	22.45	20.73
	32	3	220.79	**−9.41**	**−7.33**	12.75	18.91	20.97	19.43	23.47
	33	4	34.22	**−6.16**	**0**	18.68	21.52	31.16	34.27	22.17
	34	5	0	**0**	**0**	8.17	11.72	12.14	10.46	8.12
Airland11	35	1	12,418.32	**−0.05**	**0**	21.76	19.56	16.72	18.11	15.24
	36	2	1,540.84	**−8.49**	**−8.36**	16.26	21.47	13.17	11.49	16.26
	37	3	280.82	**−3.46**	**−2.83**	22.31	26.22	22.76	26.15	23.45
	38	4	54.53	**−6.47**	**0**	18.77	21.10	19.21	24.17	25.34
	39	5	0	**0**	**0**	8.24	17.24	9.24	7.42	7.67
Airland12	40	1	16,209.78	**0**	**0**	20.02	20.16	16.73	16.87	18.43
	41	2	1,961.39	**0**	**0**	13.75	21.01	16.72	12.16	11.47
	42	3	290.04	**−6.21**	**−3.94**	12.94	11.86	6.14	25.81	8.64
	43	4	3.49	**−2.57**	**0**	53.41	66.72	72.81	85.41	87.42
	44	5	0	**0**	**0**	6.75	5.62	5.27	5.11	8.71
Airland13	45	1	44,832.38	**−7.70**	**−7.68**	14.11	16.12	13.16	12.19	15.16
	46	2	5,501.96	**−0.79**	**−0.60**	16.43	14.19	18.81	16.74	16.52
	47	3	1,108.51	**0**	**0**	12.73	12.87	14.95	16.78	15.22
	48	4	188.46	**−50.71**	**−47.78**	28.84	29.11	32.49	34.77	38.45
	49	5	7.35	**−59.18**	**−50.06**	16.42	13.73	17.49	21.86	12.97

Then, the best values for all parameters are recorded. In the second procedure, we utilize ParamILS (a state-of-the-art method for parameter tuning and algorithm configuration) to find the appropriate parameter values as well as the components of the proposed ILS [17] (denoted as ILS). In both procedures, we randomly selected six different instances for the purpose of parameter calibration. The instances used were: Airland3, Airland5, Airland6, Airland8, Airland9 and Airland12. The values of the parameters of ILS and ILS1 are presented in Table 2. Please note that ParamILS searches for the appropriate combination of algorithm components (neighborhoods and perturbation operators) and the parameter values for ILS, whilst ILS1 only calibrates the parameter values.

5. Experimental results and comparisons

We have carried out two experimental tests. The first compares the performance of the proposed ILS with and without the multiple perturbation operators as well as the time varying perturbation strength (Section 5.1). The aim of this test is to evaluate the performance of the ILS components. The second test compares the results of ILS with the best known solutions reported by the state of the art methods for ALP instances (Section 5.2).

5.1. Effectiveness evaluation of the proposed ILS components

This experiment assesses the impact of using the multiple perturbation operators and time varying perturbation strength within the ILS framework. The proposed multiple perturbation operators employ four different operators and the time varying perturbation strength changes as the algorithm progresses. We have tested the proposed ILS using each perturbation operator separately and without the time varying perturbation strength. The outcome is seven different ILS variants as follows:

- ILS: the proposed ILS that uses both multiple perturbation operators and the time varying perturbation strength (used ParamILS for the parameter tuning).
- ILS1: same as above but manually tuned.
- ILS2: uses the first perturbation operator only.
- ILS3: uses the second perturbation operator only.
- ILS4: uses the third perturbation operator only.
- ILS5: uses the fourth perturbation operator only.
- ILS6: uses the four perturbation operators but without the time varying perturbation strength.

To ensure a fair comparison between the seven ILS variants (ILS, ILS1–ILS6), the initial solution, random seeds, number of runs, stopping condition and computer resources are the same for all experiments. For all experimental tests, we ran the seven variants 31 times, with different random seeds across the thirteen ALP benchmark instances. Then, the best result obtained by the seven ILS variants (ILS, ILS1–ILS6), over the 31 runs is reported.

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: $\Delta(\%)=((B-BKV)/BKV))*100$, where B is the best result returned by the tested algorithm over 31 runs and the BKV results are taken from [6]. In the table, the best results obtained by the compared seven ILS variants (ILS, ILS1–ILS6) are indicated in bold. From the results reported in Table 3 we can make the following observations:

– Comparing the results of ILS and ILS1, with and without the multiple perturbation operators (ILS2–ILS5), on the small-sized instances (Airland1–Airland8), both ILS and ILS1 matches the best known results on all small-sized instances (instances 1–25). Whereas ILS2, ILS3, ILS4 and ILS5 were able match the best known results on 17, 13, 6 and 3 of the 25 small-sized instances, respectively. On the large-scaled instances (Airland9–Airland13) the result of ILS and ILS1 is much better than ILS1, ILS2, ILS3 and ILS4 across all instances (instances 26–49). Indeed, ILS matched the best known results on 8 instances, and produced new best results on 16 of the 24 large-scaled instances. The results also demonstrate that the performance of the ILS2, ILS3, ILS4 and ILS5 deteriorates as the instance size increases, while both ILS and ILS1 exhibited very good performance over these instances. The results show that using multiple perturbation operators does help ILS and ILS1 in attaining very good results over both small and large instances.
– Comparing the results of ILS and ILS1, with and without the time varying perturbation strength (ILS6), the results reported in Table 3 demonstrate that both ILS and ILS1 outperformed ILS6 on all instances. ILS was able to reach the best known results on 33 instances, while ILS6 managed only 8 instances out of 49. Furthermore, ILS obtained new best results for 16 instances. Table 8 also reveals that the performance of ILS6 decreases on the larger instances (Airland9–Airland13). From the results, one can conclude that the use of a time varying perturbation strength with ILS and ILS1 helps the search in finding very good solutions and has robust performance over all tested instances.

We now compare the performance of the seven ILS (ILS, ILS1–ILS6) variants from the average and standard deviation (Std) perspective (see Tables A1 and A2, Appendix A). The average and standard deviation results obtained by ILS are much better than other ILS variants (ILS1–ILS6) on all instances (see Tables A1 and A2, Appendix A). The results show that ILS is better than other variants (ILS1–ILS6) in term of consistency, effectiveness and generality.

The results presented in Table 3 establish that both the multiple perturbation operators and time varying perturbation strength contribute to ILS performance. To verify this on a more formal basis, a multiple comparison using a Friedman statistical test with a significance level of 0.05 was conducted [18]. The Friedman test is conducted first (to properly control the FWER [18]), and, if significant differences are detected (the Friedman or Iman–Davemport statistic is below the critical level, 0.05), we use post-hoc methods (Holm and Hochberg) to obtain the adjusted p-values for each comparison between the control algorithm (the best-performing one in the comparison) and the rest

Table 4
The average ranking of Friedman test for ILS and compared variants.

Nos.	Algorithm	**Ranking**
1	ILS	**1.0408**
2	ILS1	2
3	ILS4	4.3673
4	ILS2	4.7653
5	ILS5	4.8673
6	ILS3	4.898
7	ILS6	6.0612

Table 5
The adjusted p-value.

Nos.	**Algorithm**	**Unadjusted p**	**p Holm**	**p Hochberg**
1	ILS6	0	0	0
2	ILS3	0	0	0
3	ILS5	0	0	0
4	ILS2	0	0	0
5	ILS4	0	0	0
6	ILS1	0.027966	0.027966	0.027966

Table 6
The computational results of ILS compared to the state of the art methods.

Instance name	Instance nos.	*m*	BKV	ILS	SS	BA	SA1	SA2
				Δ(%)	Δ(%)	Δ(%)	Δ(%)	Δ(%)
Airland1	1	1	700	**0**	**0**	**0**	**0**	**0**
	2	2	90	**0**	**0**	**0**	**0**	**0**
	3	3	0	**0**	**0**	**0**	**0**	**0**
Airland2	4	1	1,480	**0**	**0**	**0**	**0**	**0**
	5	2	210	**0**	**0**	**0**	**0**	**0**
	6	3	0	**0**	**0**	**0**	100	100
Airland3	7	1	820	**0**	**0**	**0**	**0**	0
	8	2	60	**0**	**0**	**0**	16.66	16.66
	9	3	0	**0**	**0**	**0**	100	100
Airland4	10	1	2,520	**0**	**0**	**0**	**0**	0
	11	2	640	**0**	**0**	**0**	3.12	3.12
	12	3	130	**0**	**0**	**0**	23.07	27.07
	13	4	0	**0**	**0**	**0**	100	100
Airland5	14	1	3,100	**0**	**0**	**0**	**0**	**0**
	15	2	650	**0**	**0**	3.08	**0**	**0**
	16	3	170	**0**	**0**	**0**	**0**	**0**
	17	4	0	**0**	**0**	**0**	100	100
Airland6	18	1	24,442	**0**	**0**	**0**	**0**	**0**
	19	2	554	**0**	**0**	3.61	**0**	**0**
	20	3	0	**0**	**0**	**0**	**0**	**0**
Airland7	21	1	1,550	**0**	**0**	**0**	**0**	**0**
	22	2	0	**0**	**0**	**0**	**0**	**0**
Airland8	23	1	1,950	0	52.05	36.15	**0**	**0**
	24	2	135	0	0	**0**	**0**	**0**
	25	3	0	0	0	**0**	100	100
Airland9	26	1	5,611.70	0	30.06	14.51	8.55	8.55
	27	2	452.92	−**1.74**	5.67	54.73	−0.58	**0**
	28	3	75.75	−**2.31**	0	87.46	**0**	**0**
	29	4	0	**0**	0	**–**	**0**	**0**
Airland10	30	1	12,329.31	−**0.06**	44.96	33.90	**0**	**0**
	31	2	1,288.73	−**1.37**	7.87	25.95	−5.39	**0**
	32	3	220.79	−**9.41**	8.88	195.88	−6.49	**0**
	33	4	34.22	−**6.16**	16.74	292.40	3.09	3.09
	34	5	0	**0**	0	–	100	100
Airland11	35	1	12,418.32	−**0.05**	17.95	16.67	**0**	**0**
	36	2	1,540.84	−**8.49**	9.19	38.54	−8.04	**0**
	37	3	280.82	−**3.46**	21.59	290.09	−2.81	**0**
	38	4	54.53	−**6.47**	2.77	474.47	**0**	**0**
	39	5	0	**0**	0	**–**	**0**	**0**
Airland12	40	1	16,209.78	**0**	22.15	23.58	**0**	**0**
	41	2	1,961.39	**0**	18.80	50.18	**0**	**0**
	42	3	290.04	−**6.21**	17.48	198.01	−3.56	**0**
	43	4	3.49	−**2.57**	271.63	13,216.91	**0**	**0**
	44	5	0	**0**	0	–	**0**	**0**
Airland13	45	1	44,832.38	−**7.70**	3.24	1.03	−7.54	**0**
	46	2	5,501.96	−**0.79**	3.72	37.47	−0.47	**0**
	47	3	1,108.51	**0**	1.98	182.69	−32.79	**0**
	48	4	188.46	−**50.71**	22.98	1,186.81	−46.62	**0**
	49	5	7.35	−**59.18**	**0**	22,308.44	−48.16	**0**

Note: Best results are shown with grey background. Δ(%) represents the percentage gap from the best result. "–" Indicates no feasible solution has been reported.

(see [18] for more details). The *p*-value of Friedman (*p*-value=0.000) and Iman–Davemport (*p*-value=0.000) are less than the critical level 0.05. This implies that there is a significant difference between the compared methods (ILS, and ILS1–ILS6). As a result, a post-hoc statistical test is used to detect the correct difference between the methods [18]. First, the Friedman test is conducted to calculate the average ranking of each method and to identify the best performing method to be used as the control method. Table 4 summarizes the average ranking (the lower the better) produced by the Friedman test for each method. ILS is ranked first with ILS1, ILS4, ILS2, ILS5, ILS3 and ILS6 following, in the order given.

Next, the Holm and Hochberg statistical test [18] is performed to obtain the adjusted *p*-values for each comparison between ILS (the control method) and ILS1, ILS2, ILS3, ILS4, ILS5 and ILS6. The adjusted *p*-values of Holm and Hochberg statistical tests in Table 5

Table 7
The programming language and computer hardware of ILS and the compared algorithms.

Nos.	Algo-rithms	Programming language	Operating system	Hardware
1	ILS	Java	Microsoft windows 7	Intel 2.66 GHz (2 Quad), 2 GB RAM
2	SS	C++	–	cIntel 2 GHz Pentium, 512 MB RAM
3	BA	C++	–	Intel 2 GHz Pentium, 512 MB RAM
4	SA1	C++	–	Intel 2.4 GHz, 512 MB RAM
5	SA2	C++	–	Intel 2.4 GHz, 512 MB RAM

Note "–" Indicates authors did not mentioned the used operating system.

demonstrate that ILS outperforms all other variants (ILS1–ILS6) with a critical level of 0.05 (adjusted p-value <0.05). This positive result supports the fact that the multiple perturbation operators and time varying perturbation strength contribute to the performance of ILS. To summarize, the favorable results achieved by ILS demonstrate the benefit of using multiple perturbation operators as well as a time varying perturbation strength. This is mainly because each problem instance has different complexity and landscape structures and consequently multiple perturbation operators help ILS work well across all different instances.

5.2. ILS compared to state of the art methodologies

In this section, we compare the results of ILS with the results reported by state of the art methodologies. The following meta-heuristic methods report the best known results in the scientific literature:

- SS: Scatter search algorithm proposed by [6].
- BA: Bionomic algorithm proposed by [6].
- SA1: A hybrid simulated annealing and variable neighborhood descent proposed by [7].
- SA2: A hybrid simulated annealing and variable neighborhood search proposed by [7].

The results of ILS as well as SS, BA, SA1 and SA2 are compared in Table 6 in terms of the percentage gap from the best known values in the literature (BKV). A bold font indicates that the obtained result is the same as the best known value. New best results are highlighted with a gray background. Table 6 shows that ILS obtained new best results for 16 instances and matched the best known results for 33 instances, of the 49 tested. Generally, it can be seen that, even though ILS does not obtain new best solutions for all instances, it does obtain many new best results. If we consider an individual comparison, ILS is able to obtain better solutions on 19, 25, 26 and 26 out of 49 instances compared to SS, BA, SA1 and SA2, respectively. These results show that ILS is better than SS, BA, SA1 and SA2. Considering the generality, ILS is able to produce good results across all large instances, while SS, BA, SA1 and SA2 produces low quality solutions for these instances.

We now compare ILS against SS, BA, SA1 and SA2 from a computational resources perspective. The programming language and the computer hardware of our ILS, as well as the compared algorithms are presented in Table 7. Please note that comparison of the computational resources can only be indicative because different researchers will inevitably use different computer resources and do not always report factors such as the operating systems, language used, compilers and the skill level of the programmer. It should also be noted that none of the compared algorithms report the number of fitness function evaluations and thus it is difficult, if not impossible, to ensure a fair comparison between the compared algorithms from a computational resource perspective, but we thought it was worth reporting in case it is useful to future researchers.

Table 8 shows the computational times (in seconds) of ILS compared to SS, BA, SA1 and SA2. It is clear from Table 8 that the computational time of ILS is lower. The reported results demonstrate that the proposed ILS produces very good quality solutions with lower computational times when compared to other state of the art methods. This indicates that ILS is an effective and efficient solution method for the ALP. Furthermore, the proposed ILS has fewer parameters that need to be set in advance.

Table 8
The computation time of ILS compared to other methods (SS, BA, SA1 and SA2).

Instance name	Instance nos.	*m*	ILS	SS	BA	SA1	SA2
Airland1	1	1	**0**	4	60	0	0
	2	2	**0**	24	45	0	0
	3	3	**0**	39	34	0	0
Airland2	4	1	**0**	6	90	1.59	1.38
	5	2	**0**	45	49	1.66	1.65
	6	3	**0**	46	43	1.98	1.91
Airland3	7	1	**0**	8	99	1.78	1.73
	8	2	**0.8**	48	58	3.12	4.22
	9	3	**0.10**	62	63	3.29	5.11
Airland4	10	1	**1.7**	8	95	1.98	2.85
	11	2	**1.9**	52	55	3.56	3.94
	12	3	**2**	46	57	3.74	5.05
	13	4	**2.3**	56	52	4.06	7.15
Airland5	14	1	**1.3**	9	100	1.85	1.89
	15	2	**2.4**	50	61	3.04	4.84
	16	3	**3.7**	54	43	4.11	4.92
	17	4	**3.1**	56	68	4.35	3.04
Airland6	18	1	**1.7**	158	274	2.12	2.14
	19	2	**2.6**	70	101	3.98	4.01
	20	3	**2.5**	54	87	4.41	5.91
Airland7	21	1	**1.8**	195	79	2.68	2.65
	22	2	**1.6**	118	124	2.83	2.37
Airland8	23	1	**4.8**	42	287	7.1	7.31
	24	2	**6.2**	121	196	10.73	9.85
	25	3	**9.5**	139	181	14.11	17.39
Airland9	26	1	**7.6**	119	554	11.59	10.12
	27	2	**11.4**	342	487	13.78	13.64
	28	3	**10.9**	390	466	17.95	18.46
	29	4	**13.7**	336	439	19.69	21.18
Airland10	30	1	**14.33**	227	925	20.12	20.75
	31	2	**15.6**	608	845	21.33	22.04
	32	3	**17.3**	668	803	27.62	25.19
	33	4	**22.7**	647	788	30.12	41.28
	34	5	**34.3**	607	762	39.85	40.15
Airland11	35	1	**18.4**	256	1417	24.17	33.84
	36	2	**21.7**	959	1287	29.09	33.99
	37	3	**34.2**	1021	1203	41.22	37.19
	38	4	**37.1**	993	1168	42.4	45.96
	39	5	**54.8**	956	1158	66.23	61.05
Airland12	40	1	**197.7**	381	2011	219.03	198.85
	41	2	**310.4**	1266	1835	362.6	313.46
	42	3	**401.5**	1454	1710	412.73	379.91
	43	4	**398.1**	1445	1688	410.33	401.04
	44	5	**357.6**	1386	1662	394.6	386.16
Airland13	45	1	**486.4**	1237	5852	566.82	528.84
	46	2	**1011.2**	3836	5379	1047.93	1294.23
	47	3	**1123.4**	4560	5158	1241	1334.33
	48	4	**1181.2**	4413	4977	1201.8	1197.48
	49	5	**1152.4**	4421	4887	1203.93	1185.46

Note: The presented time is in second. Bold indicate the best computation time.

6. Conclusion

We have proposed an Iterated Local Search algorithm for the aircraft landing problem. ILS starts with an initial solution and iteratively improves it by alternating between three different phases; local search, perturbation and acceptance. We propose a new perturbation phase for ILS that utilizes multiple perturbation operators and a time varying perturbation strength. The performance of the proposed ILS has been demonstrated across thirteen small and large instances. The experimental results show that the proposed ILS produces very good results when compared to ILS without using the perturbation operators and a time varying perturbation strength. It also outperforms the state of the art methods on many instances and produces new best results for some instances using smaller computational times. In future work, we intend to test the proposed ILS on other combinatorial optimization problems.

Appendix A

The average and standard deviation results of the proposed ILS and other variants see Table A1 and A2.

Table A1
The average results of ILS compared to other variants (bondfont indicates the best results).

Instance name	Instance nos.	ILS Average	ILS1 Average	ILS2 Average	ILS3 Average	ILS4 Average	ILS5 Average	ILS6 Average
Airland1	1	**701.43**	704.29	740.57	740.14	745.00	740.00	747.14
	2	**90.86**	91.57	117.71	109.14	111.00	107.29	113.29
	3	**0.29**	1.57	12.00	11.57	7.14	12.29	14.29
Airland2	4	**1,485.86**	1,492.86	1,498.71	1,506.57	1,506.57	1,511.57	1,533.00
	5	**211.29**	213.43	229.29	230.86	229.14	229.71	232.29
	6	**1.29**	1.57	9.29	11.57	6.43	13.86	15.43
Airland3	7	**825.00**	829.29	865.86	870.00	867.14	861.14	874.71
	8	**64.00**	64.57	76.86	78.00	78.29	81.00	84.00
	9	**1.29**	1.71	20.86	26.14	18.29	16.00	33.00
Airland4	10	**2,523.29**	2,534.57	2,547.71	2,557.57	2,563.00	2,561.71	2,586.00
	11	**643.71**	645.29	671.57	674.71	670.29	674.71	674.29
	12	**136.29**	137.57	151.71	148.00	148.00	154.71	150.43
	13	**0.71**	0.87	6.43	5.57	6.43	7.00	12.29
Airland5	14	**3,102.00**	3,117.71	3,144.29	3,128.43	3,141.14	3,141.00	3,144.57
	15	**658.86**	663.14	698.43	697.71	691.57	697.86	699.29
	16	**176.71**	179.57	200.14	198.57	198.43	199.71	198.14
	17	**1.29**	1.71	7.14	8.43	5.15	12.00	14.71
Airland6	18	**24,458.86**	24,466.00	24,673.57	24,676.00	24,850.29	24,694.43	24,713.14
	19	**564.57**	567.43	578.57	578.43	579.57	583.43	585.43
	20	**0.66**	0.71	9.29	10.00	4.57	12.71	14.15
Airland7	21	**1,558.57**	1,564.29	1,573.86	1,572.43	1,584.14	1,580.86	1,583.29
	22	**1.71**	1.86	9.86	11.57	9.71	14.00	19.00
Airland8	23	**1,959.00**	1,960.43	2,000.71	2,001.86	2,005.43	2,007.29	2,018.71
	24	**143.00**	149.00	160.43	161.29	161.86	164.14	163.14
	25	**2.71**	4.71	12.00	14.71	6.00	7.43	20.29
Airland9	26	**5,614.00**	5,621.72	7,527.80	7,953.09	8,547.23	8,649.54	8,067.80
	27	**455.27**	461.58	580.24	583.31	597.46	602.46	592.73
	28	**82.78**	84.46	151.12	154.00	155.14	149.43	155.87
	29	**2.29**	4.57	12.75	14.76	5.19	13.61	22.62
Airland10	30	**12,344.34**	12,348.34	18,831.01	18,480.58	18,393.58	18,308.16	18,579.57
	31	**1,281.55**	1,284.22	1,496.53	1,494.39	1,563.68	1,607.79	1,514.00
	32	**211.59**	214.81	273.71	280.07	278.93	279.60	279.08
	33	**41.24**	42.26	58.94	62.94	54.84	68.85	68.13
	34	**1.14**	3.57	39.01	32.73	16.73	11.44	29.01
Airland11	35	**12,419.59**	12,429.73	15,356.57	15,346.00	15,272.71	15,213.71	15,202.57
	36	**1,416.04**	1,423.46	1,824.36	1,894.64	1,857.64	1,856.50	1,818.64
	37	**276.45**	278.13	366.93	368.21	363.64	366.59	370.02
	38	**65.52**	68.88	77.97	74.59	71.73	75.87	78.71
	39	**1.29**	1.14	17.87	16.17	12.73	12.58	21.73
Airland12	40	**16,216.90**	16,226.90	19,500.14	19,495.71	19.287.57	19,394.71	19,403.29
	41	**1,969.66**	1,974.66	2,319.60	2,414.16	2,302.45	2,301.30	2,312.30
	42	**290.90**	294.70	339.66	351.66	323.41	383.59	337.59
	43	**7.15**	9.69	16.79	21.32	15.65	6.26	18.37
	44	**1.29**	3.86	14.15	7.29	8.15	6.72	9.73
Airland13	45	**41,391.41**	41,401.12	51,269.80	52,879.80	5,1145.52	51,194.29	52,102.79
	46	**5,470.18**	5,478.45	6,441.57	6,443.42	6,542.57	6,446.57	6,448.71
	47	**1,114.43**	1,121.86	1,294.01	1,289.40	1,289.40	1,291.34	1,295.14
	48	**104.86**	117.20	283.54	286.56	280.84	280.56	287.43
	49	**9.10**	11.55	24.21	19.60	20.35	15.83	15.90

Table A2
The standard deviation results of ILS compared to other variants (bondfont indicates the best results).

Instance name	Instance nos.	ILS STD	ILS1 STD	ILS2 STD	ILS3 STD	ILS4 STD	ILS5 STD	ILS6 STD
Airland1	1	**3.78**	7.87	41.85	41.25	32.79	42.52	40.61
	2	**1.07**	1.90	16.40	10.51	11.39	12.58	10.53
	3	**0.76**	2.15	13.75	13.93	8.09	11.70	13.36
Airland2	4	**4.81**	13.42	31.81	11.18	14.35	12.35	38.55
	5	**1.60**	4.47	13.43	16.05	13.45	15.28	15.79
	6	**1.38**	1.51	8.42	11.77	5.35	9.10	8.87
Airland3	7	**6.40**	10.63	41.10	42.53	40.19	36.02	43.06
	8	**5.29**	6.27	13.41	14.89	12.68	15.32	20.11
	9	**1.70**	2.14	22.50	22.91	17.96	15.94	20.58
Airland4	10	**4.64**	21.15	23.46	23.19	16.82	19.80	75.21
	11	**5.19**	8.40	13.99	16.50	18.00	18.39	16.44
	12	**7.70**	7.86	21.46	14.93	14.29	17.61	14.00
	13	**0.76**	0.83	4.58	6.97	6.50	5.69	9.57
Airland5	14	**3.21**	13.50	38.66	26.33	23.10	23.67	36.35
	15	**8.99**	13.20	27.23	27.99	14.57	25.24	28.55
	16	**8.58**	8.90	21.06	18.71	18.15	19.37	16.75
	17	**1.50**	2.43	6.89	6.68	6.01	7.83	12.93
Airland6	18	**13.80**	18.26	43.80	47.08	34.11	78.67	62.00
	19	**7.44**	8.08	22.11	21.43	22.89	19.08	27.82
	20	**0.69**	0.79	7.97	8.66	3.05	7.48	9.72
Airland7	21	**7.72**	10.36	23.05	21.12	13.93	15.95	29.97
	22	**1.38**	1.57	8.23	12.22	11.61	11.94	14.53
Airland8	23	**12.00**	13.23	60.23	60.28	57.14	54.25	69.64
	24	**9.31**	14.47	21.18	21.17	21.99	21.39	22.00
	25	**3.99**	4.42	10.57	11.01	7.72	8.32	13.63
Airland9	26	**3.23**	11.45	62.96	30.76	36.91	29.12	286.39
	27	**7.00**	8.36	34.83	25.22	10.28	9.57	26.60
	28	**7.71**	8.56	20.34	20.14	18.23	19.58	20.24
	29	**2.98**	4.16	11.64	16.74	4.29	11.29	14.27
Airland10	30	**15.78**	15.85	121.32	602.63	682.91	625.76	791.27
	31	**8.70**	9.14	23.14	15.27	20.27	27.14	32.94
	32	**7.63**	8.73	22.06	10.38	9.19	16.66	20.23
	33	**6.72**	7.60	15.00	14.75	14.07	21.53	13.65
	34	**1.77**	5.09	41.25	33.37	34.56	18.82	31.72
Airland11	35	**7.19**	13.20	120.97	229.24	338.53	300.14	393.70
	36	**5.74**	8.19	24.08	11.35	53.33	63.19	24.30
	37	**5.96**	8.51	16.39	16.70	16.33	11.35	19.20
	38	**12.64**	14.79	13.65	10.47	6.19	9.64	11.61
	39	**1.25**	1.37	16.15	17.13	16.01	12.12	15.40
Airland12	40	**6.81**	21.99	30.95	17.65	238.44	205.20	136.14
	41	**11.78**	11.86	60.06	36.45	37.58	52.71	74.78
	42	**12.32**	12.67	22.08	17.71	12.94	17.96	20.75
	43	**4.06**	4.78	13.32	14.63	10.86	0.50	11.26
	44	**1.80**	6.04	12.19	6.20	8.37	5.27	8.29
Airland13	45	**3.91**	4.93	82.04	596.67	132.00	295.32	350.47
	46	**10.34**	10.94	34.08	75.97	32.07	32.96	38.14
	47	**5.17**	7.37	35.45	31.80	12.74	20.18	12.80
	48	**6.34**	6.73	27.98	32.06	21.91	17.81	19.55
	49	**5.83**	6.56	20.24	8.81	18.53	7.64	6.46

References

[1] Bennell JA, Mesgarpour M, Potts CN. Airport runway scheduling. 4OR 2011;9:115–38.

[2] Bennell J, Mesgarpour M, Potts C. Airport runway scheduling. Annals of Operations Research. 2013;204:249–70.

[3] Beasley JE, Krishnamoorthy M, Sharaiha YM, Abramson D. Scheduling aircraft landings—the static case. Transportation Science 2000;34:180–97.

[4] Beasley J, Krishnamoorthy M, Sharaiha Y, Abramson D. Displacement problem and dynamically scheduling aircraft landings. Journal of the Operational Research Society 2004;55:54–64.

[5] Ernst AT, Krishnamoorthy M, Storer RH. Heuristic and exact algorithms for scheduling aircraft landings. Networks 1999;34:229–41.

[6] Pinol H, Beasley JE. Scatter search and bionomic algorithms for the aircraft landing problem. European Journal of Operational Research 2006;171:439–62.

[7] Salehipour A, Modarres M, Moslemi Naeni L. An efficient hybrid meta-heuristic for aircraft landing problem. Computers & Operations Research 2013;40:207–13.

[8] Lourenço HR, Martin OC, Stützle T. Iterated local search. In: Glover Fred, Kochenberger GA, editors. Handbook of metaheuristics. Springer; 2003. p. 320–53.

[9] Lourenço H, Martin O, Stützle T. Iterated local search: framework and applications. In: Gendreau M, Potvin J-Y, editors. Handbook of metaheuristics. US: Springer; 2010. p. 363–97.

[10] Blum C, Roli A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computing Surveys (CSUR) 2003;35:268–308.

[11] Katayama K, Narihisa H. Iterated local search approach using genetic transformation to the traveling salesman problem. In: Proceedings of GECCO'99; 1999. p. 321–28.

[12] Thierens D. Population-based iterated local search: restricting neighborhood search by crossover. Genetic and Evolutionary Computation—GECCO 2004. Springer; 2004. p. 234–45.

[13] Katayama K, Narihisa H. A new iterated local search algorithm using genetic crossover for the traveling salesman problem. In: Proceedings of the 1999 ACM symposium on applied computing: ACM; 1999. p. 302–36.

[14] Zhang Q, Sun J. Iterated local search with guided mutation. IEEE Congress on Evolutionary Computation: IEEE 2006:924–9.

[15] Hansen P, Mladenović N, Pérez JAM. Variable neighbourhood search: methods and applications. Annals of Operations Research. 2010;175:367–407.

[16] Ayob M, Kendall G. A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. Proceedings of the International Conference on Intelligent Technologies 2003:132–41 InTech.

[17] Hutter F, Hoos HH, Leyton-Brown K, Stützle T. ParamILS: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research 2009;36:267–306.

[18] García S, Fernández A, Luengo J, Herrera F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Information Sciences 2010;180:2044–64.

[19] Zhang Z, Liu M, Lim A. A memetic algorithm for the patient transportation problem. Omega 2015;54:60–71.

[20] Li G, Jiang H, He T. A genetic algorithm-based decomposition approach to solve an integrated equipment-workforce-service planning problem. Omega 2015;50:1–17.

[21] Liu R, Xie X, Garaix T. Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics. Omega 2014;47:17–32.

[22] Pan QK, Wang L, Li JQ, Duan JH. A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. Omega 2014;45:42–56.

[23] Ribas I, Companys R, Tort-Martorell X. An iterated greedy algorithm for the flowshop scheduling problem with blocking. Omega 2011;39(3):293–301.

[24] Pan QK, Ruiz R. An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. Omega 2014;44:41–50.

[25] Sabar NR, Ayob M, Kendall G, Qu R. A honey-bee mating optimization algorithm for educational timetabling problems. European Journal of Operational Research 2012;216(3):533–43.

[26] Sabar NR, Ayob M, Kendall G. Tabu exponential Monte-Carlo with counter heuristic for examination timetabling. In: Proceedings of the computational intelligence in scheduling, 2009. CI-Sched'09. (pp. 90-94). IEEE Symposium on, IEEE, (2009, April).

[27] Sabar NR, Ayob M, Kendall G, Qu R. Grammatical evolution hyper-heuristic for combinatorial optimization problems. Evolutionary Computation, IEEE Transactions on 2013;17(6):840–61.