Aircraft Landing Problem:
Efficient Algorithm for a Given Landing Sequence

Abhishek Awasthi[*], Oliver Kramer[†] and Jörg Lässig[*]

[*]Department of Computer Science
University of Applied Sciences Zittau/Görlitz
Görlitz, Germany
{abhishek.awasthi, joerg.laessig}@hszg.de

[†]Department of Computing Science
Carl von Ossietzky University of Oldenburg
Oldenburg, Germany
oliver.kramer@uni-oldenburg.de

October 2, 2018

Abstract

In this paper, we investigate a special case of the static aircraft landing problem (ALP) with the objective to optimize landing sequences and landing times for a set of air planes. The problem is to land the planes on one or multiple runways within a time window as close as possible to the preferable target landing time, maintaining a safety distance constraint. The objective of this well-known NP-hard optimization problem is to minimize the sum of the total penalty incurred by all the aircraft for arriving earlier or later than their preferred landing times. For a problem variant that optimizes a given feasible landing sequence for the single runway case, we present an exact polynomial algorithm and prove the run-time complexity to lie in $O(N^3)$, where N is the number of aircraft. The proposed algorithm returns the optimal solution for the ALP for a given feasible landing sequence on a single runway for a common practical case of the ALP described in the paper. Furthermore, we propose a strategy for the ALP with multiple runways and present our results for all the benchmark instances with single and multiple runways, while comparing them to previous results in the literature.

1 Introduction

Prior to landing, an aircraft must go through an approaching stage directed by the air traffic control (ATC) tower. The ATC gives instructions to the aircraft regarding to the runway, speed and altitude of the aircraft in order to align it with the allocated runway and maintain the safety distance with its preceding aircraft. During peak hours, controllers must handle safely and effectively landings of a continuous flow of aircraft entering the radar range onto

1

the assigned runway(s). The capacity of runways is highly constrained and this makes the scheduling of landings a difficult task to perform effectively. Increasing the capacity of an airport by building new runways is an expensive and difficult affair. Hence, the air traffic controllers face the problem of allocating a landing sequence and landing times to the aircraft in the radar range. Additionally, in case of airports with multiple runways, they have to take a decision on the runway allotment too, *i.e.* which aircraft lands on which runway. These decisions are made with the availability of certain information about the aircraft in the radar range [1, 2, 3]. A target landing time is defined as the time at which an aircraft can land if it flies straight to the runway at its cruise speed (most economical). This target landing time is bounded by an earliest landing time and a latest landing time known as the time window. The earliest landing time is determined as the time at which an aircraft can land if it flies straight to the runway at its fastest speed with no holding, and the latest landing time is determined as the time at which an aircraft can land after it is held for its maximal allowable time before landing. All the aircraft have to land within their time window and there are asymmetric penalties associated with each aircraft for landing earlier or later than its target landing time. Besides, there is another constraint of the safety distance that has to be maintained by any aircraft with its preceding aircraft. This separation is necessary as every aircraft creates a wake vortices at its rear and can cause a serious aerodynamic instability to a closely following aircraft. There are several types of planes which land on a runway hence the safety distance between any two aircraft depends on their types. This safety distance between any two aircraft can be easily converted to a safety time by considering the required separation and their relative speeds. If several runways are available for landing, the application of this constraint for aircraft landing on different runways usually depends upon the relative positions of the runways [1, 2, 3]. A formal definition of the ALP is given in Section 3.

The objective of the ALP is to minimize the total penalty incurred due to the deviation of the scheduled landing times of all the aircraft with their target landing times. Hence, the air traffic controllers not only have to find suitable landing times for all the aircraft but also a landing sequence so as to reduce the total penalty. This work considers the static case of the aircraft landing problem where the set of aircraft that are waiting to land is already known. For a special but practically very common case of the safety constraint, we present a polynomially bound exact algorithm for optimizing any given feasible landing sequence for the single runway case and an effective strategy for the multiple runway case. In the later part of the paper we present our results for all the benchmark instances provided by Beasley [4] and compare the results with previous work on this problem.

2   Related Work

The aircraft landing problem described in this paper was first introduced and studied by Beasley in the mid-nineties [5]. Since then, it has been studied by several researchers using different metaheuristics, hybrid metaheuristics, linear programming, variants of exact branch and bound algorithms *etc.,* for both the static and dynamic cases of the problem. In 1995, Beasley *et al.* presented a mixed-integer zero-one formulation of the problem for the single runway case and later extended it to the multiple runway case [5]. The ALP was studied for up to 50 aircraft with multiple runways using linear programming based tree search and an effective heuristic algorithm for the problem. Again in 1995, Abela *et al.* [6] proposed

a genetic algorithm and a branch and bound algorithm to solve the problem of scheduling aircraft landings. Ernst *et al.* presented a simplex algorithm which evaluated the landing times based on some partial ordering information. This method was used in a problem space search heuristic as well as a branch-and-bound method for both, the single and multiple runway case, for again up to 50 aircraft [2]. Beasley *et al.* adopted similar methodologies and presented extended results [1]. In 1998, Ciesielski *et al.* developed a real time algorithm for the aircraft landings using a genetic algorithm and performed experiments on landing data for the Sydney airport on the busiest day of the year [7]. In 2001, Beasley *et al.* developed a population heuristic and implemented it on actual operational data related to aircraft landings at the London Heathrow airport [8]. The dynamic case of the ALP was studied again by Beasley *et al.* by expressing it as a displacement problem and using heuristics and linear programming [9]. In 2006, Pinol and Beasley presented two heuristic techniques, Scatter Search and the Bionomic Algorithm and published results for the available test problems involving up to 500 aircraft and 5 runways [3]. The dynamic case of the problem for the single-runway case was again studied by Moser *et al.* in 2007 [10]. They used extremal optimization along with a deterministic algorithm to optimize a landing sequence. In 2008 Tang *et al.* implemented a multi-objective evolutionary approach to simultaneously minimize the total scheduled time of arrival and the total cost incurred [11]. In 2009, Bencheikh *et al.* approached the ALP using hybrid methods combining genetic algorithms and ant colony optimization by formulating the problem as a job shop scheduling problem [12]. The same authors presented an ant colony algorithm along with a new heuristic to adjust the landing times of the aircraft in a given landing sequence in order to reduce the total penalty cost, in 2011 [13]. In 2012, a hybrid meta-heuristic algorithm was suggested using simulated annealing with variable neighbourhood search and variable neighbourhood descent [14].

3   Problem Formulation

In this section we give the mathematical formulation of the static aircraft landing problem based on [3]. We also define some new parameters which are later used in the presented algorithm in the next sections.

Let,

N = the number of aircraft,

E_i = the earliest landing time for aircraft i, $i = 1, 2, \ldots, N$,

L_i = the latest landing time for aircraft i, $i = 1, 2, \ldots, N$,

T_i = the target landing time for aircraft i, $i = 1, 2, \ldots, N$,

ST_i = the scheduled landing time for aircraft i,

$S_{i,j}$ = the required separation time between planes i and j, where plane i lands before plane j on the same runway, $i \neq j$,

$s_{i,j}$ = the required separation time between planes i and j, where plane i lands before plane j on different runways, $i \neq j$,

g_i = the penalty cost per time unit associated with plane i for landing before T_i,

h_i = the penalty cost per time unit associated with plane i for landing after T_i,

α_i = earliness (time) of plane i from T_i, $\alpha_i = \max\{0, T_i - ST_i\}$, $i = 1, 2, \ldots, N$,

β_i = tardiness (time) of plane i from T_i, $\beta i = \max\{0, ST_i - T_i\}$, $i = 1, 2, \ldots, N$.

The total penalty corresponding to any aircraft i is then expressed as $\alpha_i g_i + \beta_i h_i$. If aircraft i lands at its target landing time then both α_i and β_i are equal to zero and the cost incurred by its landing is equal to zero. However, if aircraft i does not land at T_i, either α_i or β_i is non-zero and there is a strictly positive cost incurred. The objective function of the problem can now be defined as

$$\min \sum_{i=1}^{N} (\alpha_i g_i + \beta_i h_i) \, . \tag{1}$$

4   The Exact Algorithm

In this section we present our exact polynomial algorithm for the aircraft landing problem with a special case of the safety constraint for the single runway case. Before stating the algorithm we first present some new parameters, definitions and lemmas which are useful for its explanation. We first define D_i as the algebraic deviation of the scheduled landing time of plane i from its target landing time, $D_i = ST_i - T_i$, $i = 1, 2, \ldots, N$. We also define ES_i as the minimum of extra separation times maintained between plane i and all its preceding planes, and the deviation from its earliest landing time, for $i > 1$. For $i = 1$ we define ES_i as the deviation of its scheduled landing time with its earliest landing time, as there are no planes landing before the first plane. Mathematically, ES_i can be written as

$$ES_i = \begin{cases} ST_i - E_i, & \text{if } i = 1, \\ ST_i - \max\{SP_i, E_i\}, & \text{if } 2 \leq i \leq N, \end{cases} \tag{2}$$

where,

$$SP_i = \max_{j=1,2,\ldots,i-1} (ST_j + S_{j,i}) \, . \tag{3}$$

Here, SP_i is the time at which an aircraft i can land maintaining the safety constraint with all its preceding planes. Let P be a given landing sequence of the planes where the ith plane in this sequence is denoted by i, $i = 1, 2, \ldots, N$. Note that without loss of any generality we can assume this, since the planes can be ranked in any sequence as per their order of landing.

Lemma 1. *If the initial assignment of the landing times of all the aircraft in any landing sequence for a single runway is done according to ST_i, where*

$$ST_i = \begin{cases} L_i, & \text{if } i = N \\ \min\{PS_i, L_i\} & \text{if } 1 \leq i \leq N-1, \end{cases} \tag{4}$$

where,

$$PS_i = \min_{j=i+1,i+2,\ldots,N} (ST_j - S_{i,j}), \tag{5}$$

then the optimal solution can be obtained only by reducing the landing times of the aircraft while respecting the constraints or leaving the landing times unchanged.

Proof. Equation (4) schedules the landing times of the aircraft in the reverse order starting from the last plane to the first plane in the landing sequence. The last plane is assigned a landing at its latest landing time L_N and any of the preceding planes are assigned as late as possible from their target landing time, while maintaining the safety distance constraint. This is ensured by $\min\{PS_i, L_i\}$, where L_i is the latest landing time of aircraft i and PS_i is the closet landing time possible for aircraft i to all its following aircraft. We define PS_i as $PS_i = \min\limits_{j=i+1,i+2,\ldots,N}(ST_j - S_{i,j})$ where any plane i maintains the safety distance with all its following planes.

If any of the aircraft lands outside its time window with this assignment, then it shows that this landing sequence in infeasible. Since the landings times are assigned as close as possible to their latest landing times, increasing the landing time of any aircraft will cause infeasibility as the last aircraft is landing at its latest landing time and all the preceding planes are scheduled as close as possible. Hence, the optimal solution can be obtained only by decreasing the landing times or leaving them unchanged if there is no reduction possible. □

Given this initialization, it is possible to reduce the landing times straight away to improve the solution as is depicted in Algorithm 1. Let the initial landing times of the aircraft be assigned according to Equation (4) for any given feasible landing sequence. If any aircraft i with $i = 1, 2, \ldots, N$, has a positive deviation D_i from its target landing time and maintains a positive extra safety separation ES_i, then we can decrease the landing time ST_i by $\min\{D_i, ES_i\}$. The reason is, this reduction of the landing time is independent of other aircraft as we do not disturb the safety constraint and reduce the landing time of i only to bring it closer to its target landing time, which is the requirement of Equation (1). If $D_i > ES_i$ then we reduce the landing time by ES_i so as to maintain the safety constraint and if $D_i < ES_i$ then we reduce the landing time to its target landing time.

However, if the value of $D_i \leq 0$ for all the aircraft, then there is no improvement possible and Equation (4) is the optimal assignment for this landing sequence with respect to Equation (1). Note that $ES_i \geq 0 \ \forall \ i$, since the safety distance constraint is always maintained. We present this improvement of the initial landing times for the single runway case in Algorithm 1. We would like to point out that Algorithm 1 will not necessarily return the optimal solution but only fetch an improvement to the initial assignment of the landing times.

Algorithm 1 Improvement of Individual Landing Times

 1: Initialization: Equation (4)
 2: $i \leftarrow 1$
 3: **while** $i \neq N + 1$ **do**
 4:    **Compute** D_i, ES_i
 5:    **if** $(D_i > 0)$ **then**
 6:       $ST_i \leftarrow ST_i - \min\{D_i, ES_i\}$
 7:       **Update** D_i, ES_i
 8:    $i \leftarrow i + 1$
 9: **return** ST, ES, D

Lemma 2. *Implementation of Algorithm 1 will yield either one of the below mentioned cases for any aircraft $i, i = 1, 2, 3, \ldots, N$:*

$$a) \ D_i > 0, ES_i = 0, \quad b) \ D_i = 0, ES_i > 0,$$
$$c) \ D_i = 0, ES_i = 0, \quad d) \ D_i < 0, ES_i = 0, \qquad (6)$$
$$e) \ D_i < 0, ES_i > 0.$$

Proof. The initialization of the landing times using Equation (4) can assign the landing time to any aircraft i anywhere in its time window, if the landing sequence is feasible. Hence, we have the following five cases:

Case 1: $ST_i = E_i$.
If $i = 1$, then $D_i < 0$ and $ES_i = 0$ from Equation (2). If $i > 1$ then $D_i < 0$ but we need to check for the value of ES_i. According to Equation (2), we have $ES_i \leftarrow ST_i - \max\{SP_i, E_i\}$. Note that $ST_i \geq SP_i$, $i = 2, 3, \ldots, N$ since the safety separation is always maintained between any two aircraft landing consecutively. This implies that we can write $\max\{SP_i, E_i\} = ST_i$ due to the case constraint, *i.e.* $E_i = ST_i$. Hence, we have $ES_i = 0$ from its definition. Since a reduction in the landing time is possible only if $D_i > 0$, the values of D_i and ES_i will remain unchanged by the implementation of Algorithm 1, satisfying Case *d*.

Case 2: $E_i < ST_i < T_i$.
$D_i < 0$ for any i, which means that the landing time for aircraft i will remain unchanged. If $i = 1$, then $ES_i > 0$ from Equation (2). If $i > 1$ then again from Equation (2) we can deduce that $ES_i \geq 0$ because $ST_i \geq SP_i$ (safety constraint) and $ST_i > E_i$ (case constraint). This proves that if the initial landing time for any aircraft lies between E_i and T_i then Algorithm 1 will not fetch any reduction hence satisfying Case *d* or *e* of Lemma 2.

Case 3: $ST_i = T_i$.
$D_i = 0$ for any i since the landing occurs at the target landing time. And $ES_i \geq 0$ for any i, by the same reasons as in Case *2*. In this case as well there will be no reduction and Case *b* or *c* of Lemma 2 is satisfied.

Case 4: $T_i < ST_i < L_i$.
If the initial landing time for any aircraft i lies between T_i and L_i, then $D_i > 0$ by definition and $ES_i \geq 0$ because $ST_i > E_i$ and $ST_i \geq SP_i$. Hence, Algorithm 1 will reduce the landing time of plane i by $\min\{D_i, ES_i\}$. If $\min\{D_i, ES_i\} = D_i$ then the reduction in the landing time will fetch $D_i = 0$ and $ES_i > 0$, satisfying Case *b*. If $\min\{D_i, ES_i\} = ES_i$ then the reduction in the landing time will fetch $D_i > 0$ and $ES_i = 0$, satisfying Case *a*. However, if after the initialization the values of D_i and ES_i are equal then the implementation of Algorithm 1 will fetch $D_i = 0$ and $ES_i = 0$, satisfying Case *c*. Finally, if $ES_i = 0$ then there will be effectively no reduction because $\min\{D_i, ES_i\}$ will be equal to zero and Case *a* of Lemma 2 will be satisfied.

Case 5: $ST_i = L_i$.
$D_i > 0$ and $ES_i > 0$ after the initialization and yet again the Algorithm 1 will fetch either one of Case *a*, *b* or *c*, with the same arguments as in Case 4. $\qquad\square$

We now give some additional definitions necessary for the understanding of our main algorithm.

Definition 1. *PL is a vector of length N and any element of PL (PL_i) is the net penalty possessed by any aircraft i, $i = 1, 2, \ldots, N$. We define PL_i, $i = 1, 2, \ldots, N$, as*

$$PL_i = \begin{cases} -g_i, & \text{if } D_i \leq 0 \\ h_i, & \text{if } D_i > 0 \ . \end{cases} \tag{7}$$

With the above definition we can now express the objective function stated in Equation (1) in a compact form as

$$\min \sum_{i=1}^{N} (D_i \cdot PL_i) \ . \tag{8}$$

Definition 2. *Let i be any aircraft landing at ST_i then we define $\sigma(i)$ as the algebraic deviation of the landing time of aircraft i from its earliest landing time E_i. Mathematically, $\sigma(i) = ST_i - E_i$, $i = 1, 2, \ldots, N$.*

Definition 3. *Let aircraft $(i, i+1, \ldots, j)$ be the aircraft in any given sequence which land consecutively in this order on the same runway, we define μ such that μ is the last plane in $(i, i+1, \ldots, j)$ with $D_\mu \leq 0$.*

Definition 4. *We define $\Gamma = \{(i_1 : j_1), (i_2 : j_2), \ldots, (i_c : j_c)\}$ as the sets of aircraft which land consecutively, where c is the total number of sets in Γ and $1 \leq i_1 < j_1 < i_2 < j_2 < \cdots < i_c < j_c \leq N$. And for any set $\Gamma(k) = (i_k : j_k)$, where the aircraft $i_k, i_k + 1, \ldots, j_k$ land one after another consecutively on the same runway, the following properties hold:*

$$\begin{cases} ES_{i_k} > 0, \\ ES_m = 0, m = i_k + 1, \ldots, j_k \\ \sum_{\rho=i_k}^{j_k} PL_\rho > 0 \\ \sigma(\rho) > 0, \rho = i_k, \ldots, j_k \text{ and} \\ \sum_{\rho=\mu}^{j_k} PL_\rho > 0 \text{ if } \mu \text{ exists for } \Gamma(k) \ . \end{cases} \tag{9}$$

Definition 5. *We define $SNG(X_{i:j})$ as the smallest non-negative number in vector X from elements X_i to X_j, $(i < j)$.*

With the above concepts and definitions we present our main algorithm (Algorithm 2) for optimizing a given landing sequence P on a single runway for the special case of the ALP when the safety constraint for any aircraft is to be maintained only with its preceding plane. In other words, when $SP_i = ST_{i-1} + S_{i-1,i}$ and $PS_i = ST_{i+1} - S_{i,i+1}$. For the general case of the safety constraint the algorithm still returns a feasible solution but not necessarily optimal. Later we show with our results that this special case of the safety constraint holds for several instances and we obtain optimum results for many instances. Moreover we also obtain better results than the best known solutions for several instances.

Algorithm 2 Main Algorithm: Single Runway

1: Apply Algorithm 1
2: Calculate PL, Γ, c, σ
3: $Sol \leftarrow \sum\limits_{i=1}^{N}(D_i \cdot PL_i)$
4: **while** $\Gamma \neq \varnothing$ **do**
5:    **for** $k = 1$ **to** c **do**
6:       $(i_k, j_k) \leftarrow \Gamma(k)$
7:       $\gamma = \min\limits_{\rho = i_k,\ldots,j_k} \sigma(\rho)$
8:       $pos = \min(SNG(D_{\Gamma(k)}), ES_{i_k}, \gamma)$
9:       **for** $p = i_k$ **to** j_k **do**
10:          $ST_p \leftarrow ST_p - pos$
11:          $D_p \leftarrow D_p - pos$
12:       $ES_{i_k} \leftarrow ES_{i_k} - pos$
13:       **if** $j_k < N$ **then**
14:          $ES_{j_k+1} \leftarrow ES_{j_k+1} + pos$
15:       Update PL, σ
16:    $Sol \leftarrow \sum\limits_{i=1}^{N}(D_i \cdot PL_i)$
17:    Calculate Γ, c
18: **return** Sol

---

5 Proof of Optimality

Theorem 1. *Algorithm 2 returns the optimal value for Equation* (8) *for any given feasible landing sequence on a single runway when $SP_i = ST_{i-1} + S_{i-1,i}$ for $i = 2, 3, \ldots, N$ and $PS_i = ST_{i+1} - S_{i,i+1}$ for $i = 1, 2, \ldots, N - 1$.*

Proof. The initialization of the landing times for any sequence is done according to Lemma 1. It allocates the landing times as late as possible, hence the solution can be improved only by reducing the landing times. Thereafter we show that for any aircraft i we can reduce its landing time straight away, independent of other aircraft, if $D_i > 0$ and $ES_i > 0$. The reason is, if there is an extra safety separation between $i - 1$ and i as well as a positive deviation from the target landing time, then the reduction of ST_i by $\min\{D_i, ES_i\}$ will bring aircraft i closer to T_i and hence yield an overall reduction in the total weighted tardiness thereby improving the overall solution. Note that this reduction will neither cause any aircraft to land earlier than its target landing time nor will it disrupt the safety separation. The implementation of Algorithm 1 will fetch one of the five possibilities to all the aircraft, mentioned and proved in Lemma 2.

    The next step is to prove that a further improvement to the solution is possible iff $\Gamma \neq \varnothing$. If $\Gamma \neq \varnothing$, then we have $ES_{i_k} > 0$, $ES_m = 0, (m = i_k + 1, \ldots, j_k)$, $\sum_{\rho=i_k}^{j_k} PL_\rho > 0$, $\sigma(\rho) > 0$ where ρ are all the planes in the set $\Gamma(k)$ and $\sum_{\rho=\mu}^{j_k} PL_\rho > 0$, if μ exists. We have $ES_{i_k} > 0$ and $ES_m = 0, (m = i_k + 1, \ldots, j_k)$. Reducing the landing time of any aircraft in m will cause infeasibility as it will disrupt the safety constraint since $ES_m = 0$. But reducing the landing times of all the aircraft from i_k to j_k by $pos = \min(SNG(D_{\Gamma(k)}), ES_{i_k}, \gamma)$ will not

cause any infeasibility for two reasons. First, the definition of $\Gamma(k)$ ensures that all the planes have a positive deviation from their earliest landing times since $\sigma(\rho) > 0$ and the reduction of the landing times by *pos* will not cause any infeasibility since all the aircraft in set $\Gamma(k)$ will be allocated a landing time within their time window since $pos \leq \gamma$. Second, we would reduce all the landing times by the same amount and not more than ES_{i_k}. This will maintain the safety separation between all the aircraft in $\Gamma(k)$ and also the required separation between aircraft $i_k - 1$ and i_k.

Notice that PL_ρ is the net penalty of aircraft ρ as stated in Definition 1. Hence a positive value for the summation of the net penalties of aircraft i_k to j_k landing consecutively means, that the total tardiness penalty is higher than the total earliness penalty and an increase in the landing times of all the aircraft in $\Gamma(k)$ by the same amount is only going to worsen the solution. As for μ, let's say there exists a μ for the set $\Gamma(k)$ such that $\sum_{\rho=\mu}^{j_k} PL_\rho < 0$. This shows that aircraft μ to j_k already possess a net earliness penalty and further reducing their landing times will fetch an increase in the overall penalty. However, $\sum_{\rho=i_k}^{j_k} PL_\rho > 0$ means that $\sum_{\rho=i_k}^{\mu-1} PL_\rho > 0$ which implies that aircraft i_k to $\mu - 1$ possess a net positive tardiness penalty. Thus, a reduction in landing times by $\min(SNG(D_{i_k:\mu-1}), ES_{i_k}, \gamma)$ will reduce the total weighted tardiness as well as ensure that the increase in the earliness penalty (if any) of aircraft i_k to $\mu - 1$ does not exceed the reduction in the net tardiness penalty and thereby reducing the overall penalty. In such a case $\Gamma(k)$ will become $(i_k : \mu - 1)$.

Conversely, if $\Gamma = \varnothing$, then either one of the cases will not hold in Definition 4. We prove this by contradiction for all these cases:

Case 1: $ES_{i_k} > 0$.
If $ES_{i_k} = 0$ and all the other conditions hold then there is no scope of reduction and an increase in the landing times will only worsen the solution. Note that ES_{i_k} will never be negative, for any i_k, $i_k = 1, 2, \ldots, N - 1$.

Case 2: $ES_m = 0, m = i_k + 1, \ldots, j_k$.
If $ES_m \neq 0, (m = i_k + 1, \ldots, j_k)$ then we have two cases. One, if for some m, $ES_m < 0$, then the solution is infeasible. Second, if for some m, $ES_m > 0$ then it contradicts the definition of Γ.

Case 3: $\sigma(\rho) > 0, \rho = i_k, \ldots, j_k$.
If the value of $\sigma(\rho) = 0$, then a reduction of the landing times for all the planes in the set $\Gamma(k)$ by any positive value will make the solution infeasible since the aircraft ρ is already landing at its earliest landing time. Note that the value of $\sigma(\rho)$ cannot be negative for any aircraft ρ at any stage.

Case 4: $\sum_{\rho=i_k}^{j_k} PL_\rho > 0$.
If $\sum_{\rho=i_k}^{j_k} PL_\rho = 0$ for any plane ρ, then any change to the landing times of all the aircraft in $\Gamma(k)$ will only worsen the solution by increasing the total lateness penalty or the total earliness penalty. If $\sum_{\rho=i_k}^{j_k} PL_\rho < 0$, then the reduction of landing times is again going to worsen the solution as the total earliness penalty is already higher than the total lateness penalty. Moreover, an increase in the landing time is not good either, because it will only take us back to an earlier step where $\sum_{\rho=i_k}^{j_k} PL_\rho > 0$.

Lemma 3. *If $\Gamma(k) \neq \varnothing$ then pos exists and pos > 0.*

Proof. From Algorithm 2 we have, $pos = \min(SNG(D_{\Gamma(k)}), ES_{i_k}, \gamma)$. So *pos* will exist with a positive value only if $SNG(D_{\Gamma(k)}) > 0$, $ES_{i_k} > 0$ and $\gamma > 0$. Clearly, $ES_{i_k} > 0$ from the definition of $\Gamma(k)$. Besides, $ES_m = 0$ for $m = i_k + 1, \ldots, j_k$ and $\sum_{m=i_k}^{j_k} PL_m > 0$ again from Equation (4). Note that we proved in Lemma 2 that if $ES_i = 0$ for any $i = 1, 2, 3, \ldots, N$ then $D_i \leq 0$. Moreover, $\sum_{m=i_k}^{j_k} PL_m > 0$ shows that for at least one aircraft m in the $\Gamma(k)$ has $PL_m > 0$. Recall from Equation (1) that for any aircraft m, $PL_m > 0$ only if $D_m > 0$. Thus we have $ES_{i_k} > 0$ and $D_m > 0$ at least for one aircraft m, where $m = i_k, i_k + 1, \ldots, j_k$. Furthermore, if $\Gamma \neq \varnothing$ then obviously $\gamma > 0$ since the $\gamma = \min\limits_{\rho = i_k, \ldots, j_k} \sigma(\rho)$ and $\sigma(\rho) > 0$ for all the aircraft in the set $\Gamma(k)$ from Equation (9). Hence, this proves that *pos* will exist and will be greater than zero if $\Gamma(k) \neq \varnothing$. □

We reduce the landing times by $\min(SNG(D_{\Gamma(k)}), ES_{i_k}, \gamma)$ because this will neither disrupt the safety constraint nor cause infeasibility. Besides, this will not alter the number of planes arriving early ($D_m < 0$). If we reduce the landing times by a greater quantity we will certainly reduce the lateness penalty but we might as well end up increasing the earliness penalty by a greater amount. Hence we do not want to change the number of planes arriving early. Notice that a reduction in the landing time for aircraft j_k by *pos* means that it will increase the extra safety separation between j_k and $j_k + 1$, which is why we have line 14 in Algorithm 2. Hence, to summarize, Algorithm 2 allocates the latest possible initial landing times to all the aircraft and then makes improvements to the solution at every step until there is no improvement possible. □

6   Multiple Runways

In this section we propose an effective approach for allocating the runways to all the aircraft in a given landing sequence. We do not prove the optimality of this approach but our results show that it is an effective strategy and performs better than other approaches mentioned in the literature. In the multiple runway case the only difference is the initial assignment of the runways to all the aircraft in a given sequence. We propose an initialization algorithm for the multiple runway case which again takes the input as a landing sequence of planes waiting to land and the number of runways R at the airport. We make an assumption as in [3], that if aircraft i and j land on different runways then $S_{i,j} = 0$. Proposition (1) assigns the appropriate runway to all the aircraft and the landing sequence on each runway. Let $A_{1r}, A_{2r}, \ldots, A_{nr}$ be the sequence of planes on runway $r, r = 1, 2, \ldots, R$ and nr be the number of planes landing on runway r.

Proposition 1. *Assign the first R air planes $1, 2, \ldots, R$, one on each runway at their respective target landing times. For any following aircraft $i, i = R + 1, R + 2, \ldots, N$ assign the same runway as $i - 1$ at a landing time of T_i if T_i is greater than or equal to the allowed landing time for plane i by maintaining the safety distance constraint with all the preceding aircraft on the same runway. Otherwise, assign a runway r at T_i which offers zero deviation from T_i. If none of the above two conditions hold then select a runway which gives the least feasible positive deviation to plane i from its target landing time.*

Here we make an obvious assumption that the number of air planes waiting to land is more than the number of runways present at the airport. The landing sequence in this proposition is maintained in the sense that any aircraft i does not land before $i-1$ lands. Once we have this assignment of aircraft to runways, each runway has a fixed number of planes landing in a known sequence. Using this to our benefit, we can now apply Algorithm 2 to each runway separately. We would mention here that in this work we assume the safety separation time between aircraft landing on different runways to be equal to zero, *i.e.* $s_{A_{ir}, A_{jr'}} = 0$, where r and r' are two different runways. This assumption was also considered by Pinol *et al.* in [3].

7 Algorithm Run-Time Complexity

Lemma 4. *The run-time complexity of Algorithm 2 is $O(N^3)$ where N is the total number of aircraft.*

Proof. Calculation of Γ requires finding all the sets of planes landing consecutively, such that they hold certain properties as mentioned in Equation (9). The worst case scenario for the calculation of Γ will occur when every aircraft lies in one of the sets of Γ. Let any set $\Gamma(k)$ has x_k aircraft where $k = 1, 2, \ldots, c$, then we have, $\sum_{k=1}^{c} x_k = N$, since all the sets of Γ are disjoint. The run-time for the calculating and checking the first four properties of any set $\Gamma(k)$ is $O(x_k)$. However the computation of μ and checking $\sum_{\rho=\mu}^{j_k} PL_\rho > 0$ requires a computation of all the prior properties, if μ exists. In the worst case the value of j_k will drop down to $i_k + 1$ and this would require a total run-time of $O(x_k^2)$ where x_k is the number of aircraft in the set $\Gamma(k)$ obtained initially by the computation of the first four properties in Equation (9). Let T be the run-time of the computation of all the sets of Γ. Since all the properties are calculated in a sequential manner, we have, $T = \sum_{k=1}^{c} O(x_k^2)$. Besides, $x_k > 0$ for $k = 1, 2, \ldots, c$, we can write $\sum_{k=1}^{c} O(x_k^2) \leq \left(\sum_{k=1}^{c} O(x_k) \right)^2$. Now using $\sum_{k=1}^{c} x_k = N$ we get $T = O(N^2)$. It is straight forward to observe that the complexity of Algorithm 1 is $O(N^2)$ due to the calculations of ST and ES. The computation of PL and Sol in Algorithm 2 are both of $O(N)$ each. The *while* loop in line 4 involves several iterations so we first study the run-time of a single iteration of the *while* loop. The *for* loop in line 5 is run for the number of sets in Γ. Hence, the total run-time of the *for* loop is $\sum_{k=1}^{c} O(x_k)$, which is again equal to $O(N)$. The next steps inside the *while* loop involve the computation of Sol with a run-time of $O(N)$ and all the sets of Γ which requires $O(N^2)$ run-time each at every iteration. Since the computation of Sol and Γ is carried out sequentially, the total run-time complexity of the algorithm is basically equal to $O(\lambda N^2)$, where λ is the number of times the *while* loop is iterated. Clearly, the maximum value of λ can be equal to the maximum number of aircraft in any set $\Gamma(k)$, which is equal to the total number of aircraft N. Hence the run-time complexity of Algorithm 2 is $O(N^3)$. $\square$

8 Results

We now present our results for the aircraft landing problem with single and multiple runways for the benchmark instances provided by Beasley [4]. We implement the algorithm

Table 1: Results for small benchmark instances and comparison with Scatter Search and the Bionomic Algorithm [3].

N	R	$\mathbf{Z}_{opt}$	SCS			BA			PSA		
			$\mathbf{Z}_{SCS}$	$\mathbf{T}_{run}$	$\mathbf{G}_{best}$	$\mathbf{Z}_{BA}$	$\mathbf{T}_{run}$	$\mathbf{G}_{best}$	$\mathbf{Z}_{PSA}$	$\mathbf{T}_{run}$	$\mathbf{G}_{best}$
	1	700	700	0.4	0	700	6	0	700	0.006	0
10	2	90	90	2.4	0	90	4.5	0	90	0.211	0
	3	0	0	3.9	0	0	3.4	0	0	0.243	0
	1	1480	1480	0.6	0	1480	9	0	1480	0.095	0
15	2	210	210	4.5	0	210	4.9	0	210	0.312	0
	3	0	0	4.6	0	0	4.3	0	0	0.290	0
	1	820	820	0.8	0	820	9.9	0	820	0.300	0
20	2	60	60	4.8	0	60	5.8	0	60	0.363	0
	3	0	0	6.2	0	0	6.3	0	0	0.381	0
	1	2520	2520	0.8	0	2520	9.5	0	2520	0.014	0
20	2	640	640	5.2	0	640	5.5	0	640	0.352	0
	3	130	130	4.6	0	130	5.7	0	130	0.371	0
	4	0	0	5.6	0	0	5.2	0	0	0.377	0
	1	3100	3100	0.9	0	3100	10	0	3100	0.285	0
20	2	650	650	5	0	670.02	6.1	3.08	650	2.230	0
	3	170	170	5.4	0	170	4.3	0	170	0.456	0
	4	0	0	5.6	0	0	6.8	0	0	0.507	0
	1	24442	24442	15.8	0	24442	27.4	0	24442	0.002	0
30	2	554	554	7	0	573.99	10.1	3.61	554	2.629	0
	3	0	0	5.4	0	0	8.7	0	0	0.297	0
44	1	1550	1550	19.5	0	1550	7.9	0	1550	0.015	0
	2	0	0	11.8	0	0	12.4	0	0	0.345	0
	1	1950	2964.97	4.2	52.05	2654.92	28.7	36.15	1995	3.915	2.31
50	2	135	135	12.1	0	135	19.6	0	135	4.357	0
	3	0	0	13.9	0	0	18.1	0	0	0.421	0
	Average			6.0	2.1		9.6	1.7		0.75	0.092

as described above to find the optimal solution for the special case of the ALP in conjunction with Simulated Annealing (SA). We use a slightly modified Simulated Annealing algorithm to generate the landing sequences and Algorithm 2 to optimize each sequence to its minimum penalty. The ensemble size for SA is taken to be 20 for all the instances. The initial temperature is kept as twice the standard deviation of the energy at infinite temperature: $\sigma_{E_{T=\infty}} = \sqrt{\langle E^2 \rangle_{T=\infty} - \langle E \rangle^2_{T=\infty}}$. We estimate this quantity by randomly sampling the configuration space [15]. An exponential schedule for cooling is adopted with a cooling rate of 0.999. One of the modifications from the standard SA is in the acceptance criterion. We implement two acceptance criteria: the Metropolis acceptance probability, $\min\{1, \exp((-\triangle E)/T)\}$ [15] and a constant acceptance probability of 0.07. A solution is accepted with this constant probability if it is rejected by the Metropolis criterion. This concept of a constant probability is useful when the SA is run for many iterations and the metropolis acceptance probability is almost zero, since the temperature would become infinitesimally small.

Apart from this, we also incorporate elitism in our modified SA. Elitism has been successfully adopted in evolutionary algorithms for several complex optimization problems [16, 17]. As for the perturbation rule, we first randomly select a certain number of aircraft in any given landing sequence and permute them randomly to create a new sequence. The number of planes selected for this permutation is taken as $3 + \lfloor \sqrt{N/50} \rfloor$, where N is the number of aircraft. For large instances the size of this permutation is quite small but we have observed that it works well with our modified simulated annealing algorithm. We take the initial landing sequence for our algorithm as the sequence as per their target landing times.

Table 2: Results for large benchmark instances and comparison with the Scatter Search and the Bionomic Algorithm [3].

N	R	Z_{best}	SCS			BA			PSA		
			Z_{SCS}	T_{run}	G_{best}	Z_{BA}	T_{run}	G_{best}	Z_{PSA}	T_{run}	G_{best}
	1	5611.70	7298.57	11.9	30.06	6425.95	55.4	14.51	5703.54	14.294	1.637
100	2	452.92	478.6	34.2	5.67	700.80	48.7	54.73	**444.1**	10.78	*
	3	75.75	75.75	39	0	142.00	46.6	87.46	75.75	0.868	0
	4	0	0	33.6	0	NA	43.9	n/d	0	0.027	0
	1	12329.31	17872.56	22.7	44.96	16508.94	92.5	33.90	13515.68	31.411	9.62
	2	1288.73	1390.15	60.8	7.87	1623.15	84.5	25.95	**1203.76**	29.090	*
150	3	220.79	240.39	66.8	8.88	653.27	80.3	195.88	**205.21**	19.010	*
	4	34.22	39.94	64.7	16.74	134.27	78.8	292.40	34.22	3.532	0
	5	0	0	60.7	0	NA	76.2	n/d	0	0.0171	0
	1	12418.32	14647.40	25.6	17.95	14488.45	141.7	16.67	13401.57	27.782	7.92
	2	1540.84	1682.44	95.9	9.19	2134.67	128.7	38.54	**1400.64**	43.77	*
200	3	280.82	341.44	102.1	21.59	1095.45	120.3	290.09	**253.15**	11.125	*
	4	54.53	56.04	99.3	2.77	313.25	116.8	474.47	54.53	0.0245	0
	5	0	0	95.6	0	NA	115.8	n/d	0	0.0230	0
	1	16209.78	19800.24	38.1	22.15	20032.04	201.1	23.58	17346.45	34.93	7.01
	2	1961.39	2330.13	126.6	18.80	2945.61	183.5	50.18	**1753.67**	47.24	*
250	3	290.04	340.73	145.4	17.48	864.34	171	198.01	**233.49**	16.271	*
	4	3.49	12.96	144.5	271.63	464.76	168.8	13216.91	**2.44**	1.324	*
	5	0	0	138.6	0	NA	166.2	n/d	0	0.0308	0
	1	44832.28	46284.84	123.7	3.24	45294.15	585.2	1.03	**43052.04**	52.717	*
	2	5501.96	5706.63	383.6	3.72	7563.54	537.9	37.47	**4593.77**	48.223	*
500	3	1108.51	1130.45	456	1.98	3133.64	515.8	182.69	**712.81**	45.168	*
	4	188.46	231.76	441.3	22.98	2425.12	497.7	1186.81	**89.95**	48.6	*
	5	7.35	7.35	442.1	0	1647.02	488.7	22308.44	**0**	0.0554	*
	Average			135.5	22.0		197.8	1936.5		20.263	1.091

NA: Results not available.

All the computations were carried out in MATLAB on a 1.73 GHz machine with 2 GB RAM. To better explain and compare our results we first define some new parameters used in Table 1 and 2. Most of these parameters are derived from Pinol *et al.* [3] with slight changes as explained below.
Let,
Z_{opt} = the value of the optimal solution,
Z_{best} = the best known solutions for ALP provided in [3],
Z_{SCS} = the best solutions obtained in [3] using Scatter Search (SCS),
Z_{BA} = the best solutions obtained in [3] using the Bionomic Algorithm (BA),
Z_{PSA} = the best solutions obtained in this work,
T_{run} = the average run-time in seconds over 10 replications,
G_{best} = percentage gap between the best obtained results and Z_{opt} if the optimal solution known and Z_{best} if the optimal solution is not known.

G_{best} is defined as $G_{\text{best}} = 100 \cdot (Z_{\text{PSA}} - Z_{\text{best}})/Z_{\text{best}}$; if the optimal solution is known then $Z_{\text{best}} = Z_{\text{opt}}$. However, if $Z_{\text{best}} = 0$ we follow the same notation as assumed in [3]. If $Z_{\text{best}} = 0$, then the value of $G_{\text{best}} = 0$ if the best solution obtained is also zero and n/d (not defined) if the best solution obtained is greater than zero. This definition of G_{best} is the same as explained by Pinol *et al.* [3]. If for any instance the result obtained by us is better than the best known solution then we denote G_{best} by an asterisk (*). The values of Z_{best} are the best results obtained by Pinol *et al.* [3] during the course of their work. The results shown in Table 1 and 2 are obtained by using Algorithm 2, Proposition 1 and simulated annealing depending on single or multiple runways. For the single runway case we use simulated annealing to generate the landing sequences and Algorithm 2 to optimize

each sequence. For the multiple runway case we first generate a complete landing sequence of all the aircraft using simulated annealing, allocate the aircraft and their landing sequence to each runway using Proposition 1 and then apply Algorithm 2 to each runway separately for optimization. For brevity we call this approach PSA. It is clear from Table 1 that our approach is much faster and finds the optimal solution for all benchmark instances except for one. The reason that the optimum is found for all other instances is that the optimal sequences for all the remaining instances hold the special case of the safety constraint, *i.e.,* the safety constraint for any aircraft depends only on its preceding plane. However for the instance '*airland*8' with 50 aircraft and a single runway, our algorithm does not return the optimal solution as the optimal landing sequence does not satisfy the special case of the safety constraint.

Nevertheless, our approach still achieves a better result than Scatter Search and the Bionomic Algorithm, with a percentage gap of just 2.31 percent from the optimal value in less than 4 seconds. The average percentage gap for our approach on all the benchmark instances is 0.09 percent as opposed to 2.1 percent and 1.7 percent for Scatter Search and the Bionomic Algorithm, respectively. Moreover the average run-time for PSA is just 0.75 seconds, which is 8 times faster than Scatter Search with 6.0 seconds and more than 12 times faster than the Bionomic Algorithm with an average run-time of 9.6 seconds. Note that Pinol *et al.* [3] implemented their algorithms using C++ on a 2 GHz Pentium PC with 512 MB memory.

Table 2 presents our results for the large instances. The optimal solutions of these instances are unknown and hence we compare our results with the best known solutions. Not only do we obtain better results than the previous approaches, we also achieve better results than the best known values for 13 out of 24 instances. We are unable to reach the best known solutions for four instances but in general we perform much better than Scatter Search and the Bionomic Algorithm. The maximum percentage gap for any of these instances with the best known solutions is 9.62 percent as opposed to a percentage gap of 44.96 percent with Scatter Search and 33.90 percent with the Bionomic Algorithm, for the same set of instances. Again, the average percentage gap for our approach is 1.091 percent as opposed to 22.0 percent and 1936.5 percent for Scatter Search and the Bionomic Algorithm, respectively. Moreover, the average run-time for PSA is just 20.263 seconds which is again much faster than Scatter Search with 135.5 seconds and the Bionomic Algorithm with an average run-time of 197.8 seconds, for all the instances in Table 2. Hence, we show that the use of our polynomial algorithm fetches faster and better results than previous approaches. We would like to mention here that although we do not prove that Proposition (1) returns optimal results, nevertheless we obtain optimal solutions for all the small instances in much less time. For the large instances, the results again show that it is an effective approach and yields better and faster results for all the instances.

9   Conclusion

The Aircraft landing problem has mostly been approached using linear programming, meta-heuristic approaches or branch and bound algorithms in the last two decades [1, 2, 8, 14]. This paper is the first attempt to schedule the landings of the aircraft for a given feasible landing sequence using a polynomially bound algorithm. We have tested our approach over all the benchmark instances which have been applied in major previous research and our

results show that we are able to find better solutions than the best known solutions for many instances. In future we intend to optimize our algorithm for the general case of the safety constraint for all the benchmark instances accordingly. The authors are willing to provide the extended results for the results obtained in this work for any (or all) instance(s) via email.

Acknowledgement

References

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

[2] A. Ernst, M. Krishnamoorthy, and R. Storer, "Heuristic and exact algorithms for scheduling aircraft landings," *Networks*, vol. 34, no. 3, pp. 229–241, 1999.

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

[4] J. Beasley, "OR-library: Distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

[5] J. Beasley, M. Krishnamoorthy, Y. Sharaiha, and D. Abramson, "Scheduling aircraft landings - the static case," *The Management School, Imperial College, London SW7 2AZ, England*, 1995.

[6] J. Abela, D. Abramson, M. Krishnamoorthy, A. De Silva, and G. Mills, "Computing optimal schedules for landing aircraft," in *Proceedings of the 12th National Conference of the Australian Society for Operations Research, Adelaide*, pp. 71–90, 1993.

[7] V. Ciesielski and P. Scerri, "An anytime algorithm for scheduling of aircraft landing times using genetic algorithms," *Australian Journal of Intelligent Information Processing Systems*, vol. 4, pp. 206–213, 1997.

[8] J. Beasley, J. Sonander, and P. Havelock, "Scheduling aircraft landings at london heathrow using a population heuristic," *Journal of the Operational Research Society*, vol. 52, no. 5, pp. 483–493, 2001.

[9] J. Beasley, M. Krishnamoorthy, Y. Sharaiha, and D. Abramson, "Displacement problem and dynamically scheduling aircraft landings," *Journal of the operational research society*, vol. 55, no. 1, pp. 54–64, 2004.

[10] I. Moser and T. Hendtlass, "Solving dynamic single-runway aircraft landing problems with extremal optimisation," in *IEEE Symposium on Computational Intelligence in Scheduling, SCIS.*, pp. 206–211, 2007.

[11] K. Tang, Z. Wang, X. Cao, and J. Zhang, "A multi-objective evolutionary approach to aircraft landing scheduling problems," in *IEEE Conference on Evolutionary Computation, CEC.*, pp. 3650–3656, 2008.

[12] G. Bencheikh, J. Boukachour, A. Alaoui, and F. Khoukhi, "Hybrid method for aircraft landing scheduling based on a job shop formulation," *International Journal of Computer Science and Network Security*, vol. 9, no. 8, pp. 78–88, 2009.

[13] G. Bencheikh, J. Boukachour, and A. 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.

[14] A. Salehipour, M. Modarres, and L. Naeni, "An efficient hybrid meta-heuristic for aircraft landing problem," *Computers & Operations Research*, vol. 40, no. 1, pp. 207–213, 2012.

[15] P. Salamon, P. Sibani, and R. Frost, *Facts, Conjectures, and Improvements for Simulated Annealing.* Society for Industrial and Applied Mathematics, 2002.

[16] M. Gen, Y. Tsujimura, and E. Kubota, "Solving job-shop scheduling problems by genetic algorithm," in *IEEE International Conference on Systems, Man, and Cybernetics. Humans, Information and Technology.*, vol. 2, pp. 1577–1582, 1994.

[17] J. Kim, "Genetic algorithm stopping criteria for optimization of construction resource scheduling problems," *Construction Management and Economics*, vol. 31, no. 1, pp. 3–19, 2013.