Algorithms for single- and multiple-runway Aircraft Landing Problem

Amir Salehipour*

May 24, 2018

Abstract

The Aircraft Landing Problem is the problem of allocating an airport's runways to arriving aircraft as well as scheduling the landing time of aircraft, with the objective of minimizing total deviations from the target landing times. This work proposes new approaches to solve the Aircraft Landing Problem. The distinguishing factors of the proposed approaches include separating sequencing and scheduling decisions, decomposing the problem into smaller sub-problems, and restricting aircraft allocations to runways. We show that the combination of these factors into the proposed algorithms are very effective in solving the problem in a short time. Our work is motivated by the dynamic nature of the problem, i.e. due to the continuous changes in the number of arriving flights, and a short window for determining the landing schedules, the air traffic controllers will need to re-solve the problem on a regular basis and update the landing schedules, and fast and effective algorithms are therefore paramount. By solving a set of 49 benchmark instances we demonstrate that we fulfill this aim, and that the proposed methods obtain satisfactory solutions in a short amount of time.

Key words: Aircraft Landing Problem, Sequence relaxation, Decomposition, Runway allocation

1 Introduction

The Aircraft Landing Problem (ALP) is an important class of scheduling problems, and has broad applications in the area of transportation, providing significant benefits to systems experiencing long delays. The ALP is the problem of allocating an airport's runways to arriving aircraft as well as scheduling the landing time of aircraft. The amount of deviation from the target landing times between the earliest and latest landing times is the basis for determining the penalty. The objective of the ALP is to minimize total penalized deviations from the target landing times. Hence, the objective function includes two components: (1) penalizing early landings and (2) penalizing late landings. In practice, penalty per unit of early or late landing is a function of aircraft type, passenger capacity, and fuel economy, among others (Furini et al., 2015).

A safe landing must ensure the minimum separation between every ordered pair of aircraft. Two most applied separation requirements are "radar separation", which is a lengthwise spacing of five nautical miles (some busy airports may consider less than this) and a vertical spacing of 1,000 feet, and "wake turbulence separation (also known as wake vortex)", which is a time spacing and depends on the type of aircraft. The International Civil Aviation Organization (ICAO) considers four categories

*ARC DECRA Fellow, University of Technology Sydney, Australia. Email: amir.salehipour@gmail.com

of Light (L), Medium (M), Heavy (H) and Super (S) for wake turbulence (category S only includes Airbus A380). Other factors such as aircraft routes, and weather conditions may impact separation between aircraft.

Other operational constraints for the ALP may include earliest and latest time windows for landings, runway suitability for specific landings, and constrained position shifting. The latter is used to model the "fairness policy" among airlines (Balakrishnan and Chandran, 2010), and implies each aircraft in the sequence may only deviate by a certain number of positions from its position in the first-come-first-serve (FCFS) order.

As a routine administered by the air traffic controllers, the aircraft are sequenced for landings by using the FCFS rule, i.e. according to their arrival times to the airport. However, significant improvements are possible with the use of optimization techniques. This study contributes into this aim by presenting optimization algorithms for the ALP.

Available methods for solving the ALP can be grouped into exact and heuristics. Ernst et al. (1999) proposed an exact simplex-based algorithm for the ALP, and obtained optimal solutions for instances with up to 44 aircraft. Bianco et al. (1999) modeled the problem on one runway as a single machine scheduling problem with ready times, sequence-dependent setup times and objective function of minimizing the completion times. A mixed-integer program was developed by Beasley et al. (2000); the authors managed to solve instances with up to 50 aircraft to optimality. Later, several algorithms including branch-and-price and column generation were proposed for the ALP (Wen et al., 2005; Ghoniem and Farhadi, 2015; Ghoniem et al., 2015). Those exact methods solved larger instances, particularly on multiple runways.

Among heuristic and meta-heuristic algorithms, the first efforts are due to Abela et al. (1993), who proposed a Genetic Algorithm (GA) to solve small instances. In the domain of heuristic and meta-heuristic algorithms for the ALP, notable studies include the work of Salehipour et al. (2013), Vadlamani and Hosseini (2014), Sabar and Kendall (2015) and Girish (2016). Although, those studies investigated advanced meta-heuristics for the ALP, the algorithms do not have robust performance, due to inclusion of randomized components. In addition, they lose their performance for larger instances. Recently, Salehipour and Ahmadian (2017) and Salehipour et al. (2018) developed several algorithms including novel relaxation neighborhoods, and overcame some of the limitations of the previous heuristic algorithms, though they did not consider multiple runways. Salehipour et al. (2009) studied special cases of the ALP.

This study contributes into solving the ALP by developing simple and efficient algorithms, through changing the traditional views towards modeling and solution methods. Our solution methods, although relying on exact solvers, significantly enhance the capabilities of those solvers in delivering very high quality solutions for the ALP in a short time. Without applying the proposed solution methods, exact solvers are unable to solve the problem in a reasonable amount of time, which is extremely important for real-time and online applications. Indeed, our work is motivated by the dynamic nature of the ALP, i.e. due to the continuous changes in the number of arriving flights, and a short window for determining the landing schedules, the air traffic controller will be required to re-solve the problem on a regular basis and update the landing schedules, and therefore, fast and effective algorithms are crucial. Available exact solution methods are not able to deliver good quality solutions fast enough (Beasley et al., 2000; Wen et al., 2005; Ghoniem and Farhadi, 2015; Ghoniem et al., 2015). Furthermore, they are not efficient in solving large instances of the ALP. On the other hand, existing heuristic methods

do not guarantee solution's quality, nor do they have robust performance, because they include randomized elements (see for example Pinol and Beasley (2006); Salehipour et al. (2013); Girish (2016)). Not to mention that because they include a combination of different neighborhood structures, as well as hybrid algorithms, their implementation requires advanced algorithmic techniques and parameters tunning.

We believe that the most important advantages of the algorithms proposed in this study are (1) their conceptual simplicity, (2) their ability to deliver high quality solutions for large instances in a short time, (3) their robust performance, (4) their utilization of exact solvers, and (5) their usability for practical and online applications. Our algorithms have only a few parameters, allowing them to be easily tuned and implemented. In addition, we utilize available optimization solvers such as CPLEX and Gurobi because they have greatly improved in recent years, making solver based algorithms significantly faster and more effective.

The remaining of the paper is organized as follows. Section 2 defines the problem, notations, and formulates the ALP as a mixed-integer program. Section 3 proposes two solution methods for the ALP, one for the case of one runway, and one for the case of multiple runways. The section is followed by Section 4, which discusses the computational outcomes of the algorithms. Finally, Section 5 concludes the paper.

2 Problem statement

Given a set of aircraft $I = \{1, \ldots, n\}$, and runways $R = \{1, \ldots, m\}$, the aim of the Aircraft Landing Problem (ALP) is to obtain a sequence for aircraft landings and the schedule for those landings such that penalties associated with early and late landings are minimized.

The early landing of aircraft i is given by $\alpha_i = \max(0, T_i - x_i)$ when aircraft i lands earlier than time T_i, that is, if $x_i \leq T_i$, where x_i is the scheduled (planned) landing time of aircraft i, and T_i is the target landing time of aircraft i. The late landing of aircraft i is given by $\beta_i = \max(0, x_i - T_i)$ when aircraft i lands later than time T_i, that is, if $x_i \geq T_i$. Also, aircraft i is required to land in the time window $[E_i, L_i]$, i.e. $E_i \leq x_i \leq L_i$, where E_i is the earliest landing time of aircraft i, and L_i is the latest landing time of aircraft i. Note that $\alpha_i > 0$ if the decision variable x_i lies within the range $[E_i, T_i]$, while $\beta_i > 0$ if it lies within the range $[T_i, L_i]$. The reader may realize that capturing two penalties results in a piecewise objective function.

In addition, there is a safety separation time $s_{ij} > 0$ between every pair of aircraft $i, j \in I, i \neq j$ if they land one after another on the same runway. The separation time can be different for different pairs of aircraft. Hence, a change in the landing sequence would change the landing schedule. We assume that the separation time between a pair of aircraft landing on different runways is 0. This assumption has been taken in previous studies as well, for two examples see Pinol and Beasley (2006) and Girish (2016). Table 1 summarizes all mathematical notations used in this paper.

2.1 Mathematical formulation

The ALP can be formulated as a mixed-integer program. Model ALP discusses this, which is due to Salehipour et al. (2013).

Model ALP

Table 1: Mathematical notations used in the paper.

Sets	
I	The set of aircraft, $I = \{1, \ldots, n\}, \lvert I\rvert = n$.
R	The set of runways, $R = \{1, \ldots, m\}, \lvert R\rvert = m$.
Parameters	
s_{ij}	Separation time units between two ordered aircraft i and j when landing on the same runway; $s_{ij} > 0, i, j \in I, i \neq j$.
T_i	Target landing time of aircraft i; $T_i \geq 0, i \in I$.
E_i	Earliest landing time of aircraft i, $E_i \geq 0, i \in I$.
L_i	Latest landing time of aircraft i, $L_i \geq 0, L_i > E_i, i \in I$.
c_i^-	Cost of early landing of aircraft i, $c_i^- > 0, i \in I$.
c_i^+	Cost of late landing of aircraft i, $c_i^+ > 0, i \in I$.
Variables	
x_i	Scheduled landing time of aircraft i; $x_i \geq 0, i \in I$.
α_i	Earliness of landing aircraft i (landing before target landing time); $\alpha_i = \max(0, T_i - x_i)$, $\alpha_i \geq 0, i \in I$.
β_i	Lateness of landing aircraft i (landing after target landing time); $\beta_i = \max(0, x_i - T_i)$, $\beta_i \geq 0, i \in I$.
y_{ij}	Whether aircraft i lands before aircraft j; $y_{ij} \in \{0, 1\}, i, j \in I, i \neq j$.
δ_{ij}	Whether aircraft i and j land on the same runway; $\delta_{ij} \in \{0, 1\}, i, j \in I, i \neq j$.
γ_{ir}	Whether aircraft i is allocated to runway r; $\gamma_{ir} \in \{0, 1\}, i \in I, r \in R$.

$$\min z = \sum_{i=1}^{n} (\alpha_i c_i^- + \beta_i c_i^+) \tag{1}$$

Subject to

$$E_i \leq x_i \leq L_i \qquad \forall i \in I \tag{2}$$

$$x_i - T_i = \alpha_i - \beta_i \qquad \forall i \in I \tag{3}$$

$$x_j - x_i \geq s_{ij}\delta_{ij} - M y_{ji} \qquad \forall i, j \in I, i \neq j \tag{4}$$

$$y_{ij} + y_{ji} = 1 \qquad \forall i, j \in I, i \neq j \tag{5}$$

$$\delta_{ij} \geq \gamma_{ir} + \gamma_{jr} - 1 \qquad \forall i, j \in I, i \neq j, \forall r \in R \tag{6}$$

$$\sum_{r=1}^{m} \gamma_{ir} = 1 \qquad \forall i \in I \tag{7}$$

$$y_{ij}, \delta_{ij}, \gamma_{ir} \in \{0, 1\} \qquad \forall i, j \in I, i \neq j, \forall r \in R \tag{8}$$

$$x_i, \alpha_i, \beta_i \geq 0 \qquad \forall i \in I \tag{9}$$

The objective function (Equation (1)) minimizes the total cost of landing deviation from the target landing times. Constraints 2 ensure every aircraft lands in its time window. Constraints 3 link the decision variables x_i and parameters T_i to decision variables α_i and β_i. Constraints 4 impose if two aircraft i and j land on the same runway, at least s_{ij} time units should be elapsed before aircraft j could be land on that runway. Given a pair of aircraft, Constraints 5 ensure one lands before the other. Constraints 6 link the decision variables δ_{ij} and γ_{ir}. Constraints 7 ensure every aircraft lands on only one runway. Constraints 8 and 9 ensure decision variables y_{ij}, δ_{ij}, and γ_{ir} only take binary values, and decision variables x_i, α_i and β_i only take non-negative values. In the model, M is a constant taking a large value, which can be numerically calculated from the data.

2.2 Illustrative example 1

Now consider the following example, which includes 10 aircraft and two runways ($n = 10$ and $m = 2$), and represents instance "Airland1" from OR Library at http://people.brunel.ac.uk/~mastjjb/jeb/info.html. Table 2 details the landing data for the example, and Table 3 shows the separation times between every pair of aircraft landing on the same runway. Throughout this study we assumed that separation time between a pair of aircraft landing on two different runways is 0. Also, for the sake of this example, assume that the earliness and lateness costs per time unit of deviation from T_i are 1. An intuitive landing sequence for this example is then $\pi = (3, 4, 5, 6, 7, 8, 9, 1, 10, 2)$, which is obtained by ordering the aircraft landings on a non-decreasing order of their target landing times.

Table 2: Aircraft landing data for example 1.

Aircraft	E_i	T_i	L_i
1	129	155	559
2	195	258	744
3	89	98	510
4	96	106	521
5	110	123	555
6	120	135	576
7	124	138	577
8	126	140	573
9	135	150	591
10	160	180	657

Table 3: The separation times between every pair of aircraft for problem in example 1.

i	1	2	3	4	5	6	7	8	9	10
1	-	3	15	15	15	15	15	15	15	15
2	3	-	15	15	15	15	15	15	15	15
3	15	15	-	8	8	8	8	8	8	8
4	15	15	8	-	8	8	8	8	8	8
5	15	15	8	8	-	8	8	8	8	8
6	15	15	8	8	8	-	8	8	8	8
7	15	15	8	8	8	8	-	8	8	8
8	15	15	8	8	8	8	8	-	8	8
9	15	15	8	8	8	8	8	8	-	8
10	15	15	8	8	8	8	8	8	8	-

In sequence π, aircraft 3 is the first to land on runway 1 followed by aircraft 4. Therefore, $s_{34} = 8$, implying aircraft 4 cannot land earlier than time 106 on runway 1, which turns to be the target landing time of aircraft 4. Similarly, aircraft 5 and 6 can land on runway 1 at their target landing times. Thus, no deviation from their target landing times is occurred. Aircraft 7, however, cannot land earlier than time 143 on runway 1. This would impose 5 time units of delay for this aircraft. However, if aircraft 7 lands on runway 2 (different runway from that of aircraft 6) it can land at its target landing time, i.e. at time 138, and no delay therefore is experienced for aircraft 7. In this particular example, it would be indeed beneficial to force aircraft 6 and 7 to land on two different runways. We later discuss in Section 3.3 that how forcing two aircraft not to land on the same runway may lead to an algorithm for solving the ALP on multiple runways.

3 The proposed solution methods

The solution methods proposed in this study for solving the Aircraft Landing Problem (ALP) operate by decomposing the ALP into two problems of sequencing aircraft landings, and scheduling those

landings. It is not difficult to show that if a landing sequence is known, obtaining an optimal schedule for the sequence can be performed in polynomial time. Therefore, it only suffices to obtain a sequence for aircraft landings because we can then deliver an optimal landing schedule for this sequence by utilizing available solvers, e.g. CPLEX (ILOG, 2012) and Gurobi (Gurobi Optimization, 2016). In summary, when a new landing sequence is generated, e.g. through changing position of two aircraft, we can obtain the optimal schedule for this sequence. Therefore, we can compare the sequence with the available ones and choose the one with the smallest objective function value.

We propose two algorithms for obtaining high quality schedules for the ALP. The first algorithm is designed for the ALP when only one runway exists. The second algorithm can only be applied when multiple runways are available. For both algorithms the initial solution, i.e. a landing sequence, is generated by applying the construction method of Salehipour et al. (2013). From here, the algorithms implement a combination of several approaches and procedures, including separation of sequencing and scheduling decisions, decomposition of the problem into smaller sub-problems, and aircraft allocations to runways. These procedures aim to deliver new sequences by performing systematic changes to the current sequence. We utilize the available exact solvers to deliver the optimal schedule for the new sequences. In the next sections we discuss the procedure of generating an initial sequence, followed by algorithms for single- and multiple-runway cases.

3.1 Generating an initial sequence

In order to generate an initial sequence for the ALP we apply the construction method of Salehipour et al. (2013). This algorithm obtains an initial sequence for landings by applying the Earliest Target Landing Time (ETLT) dispatching rule, which positions aircraft landings on a non-decreasing order of their target landing times. Therefore, the aircraft with the earliest target landing time lands first, followed by the one with the second earliest target landing time. The algorithm stops when all aircraft are positioned to land, implying a landing sequence is developed. Note that because the ETLT ensures aircraft are sequenced according to their target landing times, which lie between the earliest and latest landing times, feasible sequences are always delivered. The ETLT is summarized in Algorithm 1.

Algorithm 1: The Earliest Target Landing Time (ETLT) dispatching rule for generating an initial sequence for the Aircraft Landing Problem.

Input: An instance of the ALP, where $I = \{1, \ldots, n\}$ and $R = \{1, \ldots, m\}$ are sets for aircraft and runways.

Output: A feasible landing sequence π for the ALP.

Let $\pi = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, where $T_{\sigma(1)} \leq T_{\sigma(2)} \leq ... \leq T_{\sigma(n)}$, be the sorted sequence of aircraft landings (sorted on a non-decreasing order of target landing times);

return π;

3.2 The Relax algorithm for the single runway case

This section proposes an algorithm for solving the ALP on a single runway. The algorithm starts from an initial sequence, which is generated by the ETLT dispatching rule, and proceeds by fixing a majority of aircraft in their current landing positions, and allowing only a small number of them to be re-positioned. Therefore, we call it the "Relax" algorithm because the method relaxes a small subset of aircraft to freely move around (thus re-ordering is possible), while keeps the current positions for the remaining aircraft. More precisely, $[k\%]$ of aircraft, where $k \in \mathbb{Z}^+, k \ll n$ is a parameter of the

method, are relaxed from their current positions in the sequence. Hence, they can be moved both forward and backward in the sequence.

In fact, those aircraft that are relaxed from their positions may be considered as a sub-problem of the original ALP. We are particularly interested in keeping the size of the sub-problem small, i.e. only a small number of aircraft are in the sub-problem. This is because the exact solvers are then able to optimally solve the sub-problem, and deliver the optimal sequence and schedule for the sub-problem. By starting from the beginning of the sequence and iterating to the end, we will solve a set of $\lceil \frac{n}{[k\%]} \rceil$ sub-problems. Indeed, the solution of each sub-problem positively contributes into a very good quality solution for the original ALP because we apply exact solvers to solve the sub-problems to optimality.

The Relax algorithm keeps generating and solving the sub-problems until the stopping criterion is met. Algorithm 2 summarizes the Relax solution method.

Algorithm 2: The Relax algorithm for the Aircraft Landing Problem (ALP) on a single runway.

Input: The mixed-integer program for the ALP (Model ALP); parameters $k \in \mathbb{Z}^+, k \ll n$ and $p := 0$; an initial sequence π and its schedule S.
Output: An improved sequence π and its schedule S for aircraft landings.

```
while the stopping condition is not met do
    while p < n − [k%] do
        relax(p) (relaxing the position of [k%] aircraft in π starting from position p);
        π′, S′ ← optimize(π) (optimizing the relaxed sequence by using an exact solver);
        Δ ← objective(S′) − objective(S);
        if Δ < 0 then
            S ← S′;
            π ← π′;
        end
        p := p + [k%];
    end
end
Report π, S;
```

3.3 The Restricted Runway Allocation algorithm for the multiple runway case

The example of Section 2.2 carries a fundamental observation regarding designing a solution method for the ALP on multiple runways. We first build the theoretical foundations for this algorithm, and then explain the algorithm.

Proposition 1. *Given $r > 1$ and a permutation of aircraft sorted on a non-decreasing order of $T_i, \forall i \in I$, i.e. $T_i \leq T_j, \Leftrightarrow i < j$, a new (improved) schedule may be obtained by letting a pair of ordered aircraft i and j to land on different runways provided that $T_j < T_i + s_{ij}$.*

Proof. The proof is followed by observing that a separation time of $s_{ij} > 0$ must be hold between aircraft i and j landing on the same runway. Since i lands prior to j, $T_j < T_i + s_{ij}$ implies that aircraft j cannot land at its target landing time. Because $s_{ij} = 0$ if aircraft i and j land on different runways, therefore, by allocating aircraft j to another available runway we may be able to land aircraft j at its target landing time. □

Note that Proposition 1 aims to allocate aircraft to different runways. Because runway allocation is performed from a non-optimal sequence, the proposition does not therefore guarantee optimal solutions. Also, the proposition only relies on the values of separation time to allocate aircraft to

runways. Nevertheless, the proposition proposes a method to generate new (improved) solutions for the ALP.

Proposition 2. *Following Proposition 1, if two aircraft i and j are allocated to land on two different runways, the inequality*

$$\gamma_{ir} + \gamma_{jr} \leq 1, \forall r \in R \tag{10}$$

can be generated.

Proof. Note that the binary decision variables $\gamma_{ir} \in \{0, 1\}, \forall i \in I, \forall r \in R$ takes a value of 1 if aircraft i lands on runway r and 0 otherwise. Constraints 10 state that only one of those aircraft can be allocated to runway r. □

Proposition 3. *Given parameter $\lambda > 0$, generating constraints 10 by following*

$$T_j + \lambda < T_i + s_{ij}, \forall i, j \in I \tag{11}$$

may result in an improved schedule (with an improved objective function value) if and only if $\lambda = 0$ does not lead to the optimal schedule.

Proof. Following Proposition 1, note that adding a positive constant λ to the left hand side of inequality $T_j < T_i + s_{ij}$ impacts the runway allocation because the greater the value of λ, the less likely that aircraft i and j are allocated to different runways. This results in more room for possible improvements because the optimization process may allocate the aircraft to the same runway. Therefore, new improved schedule may be obtained, of course, at the cost of higher computation time. □

According to the experiments we performed, which are detailed in Section 4, generating and adding constraints 10 to Model ALP can develop a landing schedule, which has a very competitive quality. Also, note that the so obtained schedule may easily be improved by an exact solver depending on the chosen values for λ. Generally speaking, larger values of λ aim to obtain better quality schedules (with smaller values for the objective function). Also, larger values of λ lead to longer solving time because less number of aircraft are pre-allocated to runways. The special case of $\lambda = 0$ results in a very restricted schedule, where a large number of aircraft are pre-assigned to runways, while $\lambda \gg M$, where M is a very big constant, results in the original mixed-integer program for the ALP. Thus, the user can investigate a trade-off between the quality of solution and the computation time.

Proposition 3 can be utilized to develop an algorithm for the ALP on multiple runways. The algorithm starts from an initial landing sequence, and generates a set of inequalities 10, and adds them to Model ALP. Then, it applies an exact solver to deliver an optimal schedule. Inspired by the role of inequalities 10, we call this solution method the Restricted Runway Allocation (RRA) algorithm. Algorithm 3 outlines the RRA.

3.4 Illustrative example 2

Recall from our earlier example in Section 2.2 that we observed that it would be beneficial to land aircraft 6 and 7 on two different runways. This can indeed be observed by applying Proposition 2 and Proposition 3 with $\lambda = 0$, which leads to

Algorithm 3: The Restricted Runway Allocation (RRA) algorithm for the Aircraft Landing Problem (ALP) on multiple runways.

Input: The mixed-integer program for the ALP (Model ALP); parameter $\lambda > 0$, $\lambda_{max} > \lambda$, and change factor τ; an initial sequence π and its schedule S.
Output: An improved sequence π and its schedule S for aircraft landings.

while *the stopping condition is not met* **do**
  while $\lambda < \lambda_{max}$ **do**
    Generate inequalities of the form $\gamma_{ir} + \gamma_{jr} \leq 1, \forall r \in R$ by utilizing Propositions 2 and 3; add the inequalities to Model ALP;
    $\pi, S \leftarrow$ `optimize`(π) (solve the model by using an exact solver);
    $\Delta \leftarrow$ `objective`(S') $-$ `objective`(S);
    if $\Delta < 0$ **then**
      $S \leftarrow S'$;
      $\pi \leftarrow \pi'$;
    end
    $\lambda := \lambda + \tau$;
  end
end
Report π, S;

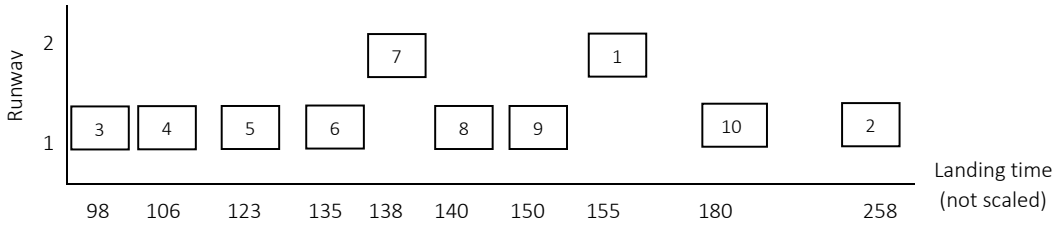

Figure 1: The Gantt chart illustrating the aircraft landing sequence and schedule for example 2.

$$\gamma_{6r} + \gamma_{7r} \leq 1, r = 1, 2$$

In addition, the following inequalities can be generated by applying Proposition 2:

$$\gamma_{7r} + \gamma_{8r} \leq 1, r = 1, 2$$

$$\gamma_{9r} + \gamma_{1r} \leq 1, r = 1, 2$$

which imply that it may be beneficial not to land aircraft 7 and 8, and 9 and 1 on the same runway. The Gantt chart for this example reveals this fact (see Figure 1). Note that if we use more than two runways here, we yield a total objective function of 0, because aircraft 8 could be scheduled to land on its target time on runway 3. We can easily generate the complete set of these constraints and add them to Model ALP, and perform the re-optimization.

4 Computational results

We tested the proposed solution methods on 49 standard benchmark instances of the Aircraft Landing Problem (ALP), which are available from OR Library at http://people.brunel.ac.uk/~mastjjb/jeb/info.html. The instances range from small (10 aircraft) to large (500 aircraft). Among those, 13 instances utilize one runway, and the remaining 36 instances utilize between two and five runways (multiple runways).

Because the scope of this study is to develop effective modeling and solution methods for the ALP

Table 4: The value of parameters used in the Relax and RRA algorithms.

Parameter	Relax algorithm	RRA algorithm
k	5	N/A
λ	N/A	15
λ_{max}	N/A	110
τ	N/A	5
Time (sec)	300	300

by utilizing exact solvers we therefore compare the performance of the proposed solution methods with two best performing exact methods in the literature. Those methods are the optimization solver CPLEX (ILOG, 2012), and the column generation (CG) algorithm of Ghoniem and Farhadi (2015). In addition, because the ALP with single runway is generally more difficult to solve than with multiple runways, and this is well-documented in the previous studies and observed in our results as well, we discuss the computational experiments in two parts; first for the single runway instances, and then for the multiple runway instances.

Table 4 summarizes the values for parameters, which were set after extensive testing. The reason for choosing those values is because they result in very good quality solutions in a short time, which is in line with the aim of the study. For example, in the Relax algorithm we set $k = 5$, which implies that we relax [5%] of aircraft to be optimized in each iteration of the algorithm. The reason for choosing $k = 5$ is because we realized that this value of k results in a very good quality solution in a short time. In particular, we observed that larger values ($k > 5$) increase the computation time without significant impact on the solution quality, and smaller values ($k < 5$) result in increased number of optimization iterations with considerable amount of computation time. Furthermore, due to the penalty for both earliness and tardiness we observed that aircraft may not be re-sequenced far from the current position both forward and backward. The Relax algorithm stops when there is no further improvement or when the time limit reaches, which is less than five minutes.

The Relax and RRA algorithms were coded in the programming language C++ and Python 2.7, and we implemented the solver CPLEX 12.5.0 (ILOG, 2012) as the optimization solver. We ran the algorithms on a personal computer with an Intel®Core i5-4570S CPU at 2.90GHz, and 8GB of memory. We used only one thread.

Section 4.1 and Section 4.2 discuss the outcomes of four methods of CPLEX, CG, Relax, and RRA for solving 49 instances of ALP. We considered four criteria to evaluate each method. Those criteria are "number of best solutions obtained", "percent of best solutions obtained", "average computation time in seconds" and "average gap in percent". For a single instance, the gap is calculated as

$$\frac{z - z^*}{z^*} \times 100, z^* > 0$$

where z is the objective function value of a solution method, and z^* is the best value for the objective function, which is obtained across those four methods. It is important to mention that the stopping criterion for both solver CPLEX and CG is 60 minutes of running, which is 12 times longer than the computation time requirement of our methods.

4.1 Results for instances with one runway

Table 5 summarizes the outcomes of three methods of CPLEX, CG and Relax over four criteria discussed earlier. As the table shows the most superior performance in terms of obtaining the best

Table 5: Summary of the solution methods for solving benchmark instances with one runway.

Method	Number of best solutions obtained	Percent of best solutions obtained	Average time (sec)	Average gap (%)
CPLEX	12	92.31	1384.96	0.00
CG	10	76.92	1385.14	0.08
Relax	10	76.92	96.67	0.74

known solutions is observed for CPLEX, followed by Relax and CG. However, CPLEX spends more than 20 minutes, on average, to obtain those solutions. The CG has a weak computation time performance as well, which is very similar to that of CPLEX. It is also less capable than CPLEX because it obtains less number of best known solutions. On top of this, both CPLEX and CG methods suffer from long computation times for medium and large instances as they spend 60 minutes to deliver solutions when $n \geq 100$. This is further shown in Table 6. On the contrary, our proposed Relax method is very quick. Its average computation time is about 1.5 minutes. Also, it reports the best known solution for 10 instances and that within 5 minutes. More importantly, while the Relax obtains the same number of best solutions as the CG, it is more than 12 times faster. Note that no single method is shown to be able to deliver the best known solution for all instances.

According to Table 6, which details the computational results of solution methods of CPLEX, CG and Relax over 13 benchmark instances with one runway, all methods are able to deliver the optimal solution for the first eight instances, i.e. for those with up to 50 aircraft. Within these 13 instances, the average gap of the Relax method is 0.74%. Although the gap is small, the reason that the Relax's gap is slightly larger than that of CPLEX and CG is mainly related to the two last instances (with 250 and 500 aircraft). For these two instances we do not have the outcomes of the CG method over a short computation time of five minutes (thus, within the same time of the Relax). We highly suspect that if the CG would run for 5 minutes, instead of 60 minutes, its gap would be larger than the current one. Similar behavior is highly expected for the CPLEX, therefore, strengthening the role of the Relax method in delivering high quality solutions for the ALP, and that within a reasonable computation time.

The discussion concludes that the Relax algorithm is a high performing method for solving the ALP on one runway, particularly, it is able to deliver the best known solutions for 10 instances, and within five minutes, and that about 12 times faster than CPLEX and CG. This conclusion is further supported by the overall performance of the methods, which is illustrated in Figure 2.

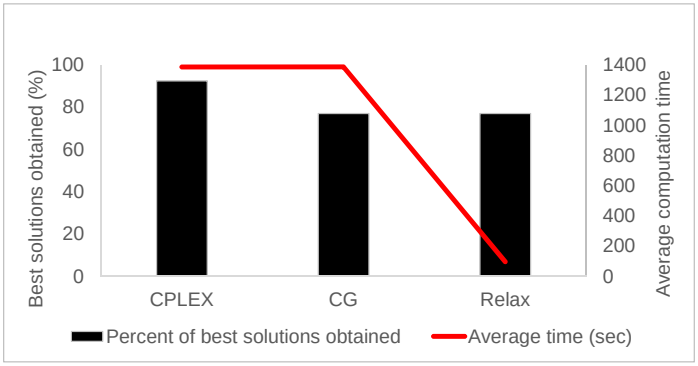

Figure 2: Effectiveness of three solution methods for solving instances of ALP with one runway.

Table 6: Detailed outcomes of three solution methods for solving benchmark instances of the ALP with one runway. The best known solutions obtained by the Relax algorithm were highlighted.

				CPLEX			CG			Relax		
Instance	n	z^*	FCFS	z	Time(s)	Gap(%)	z	Time(s)	Gap(%)	z	Time(s)	Gap(%)
Airland1	10	700.00	1280	700.00	0.11	0.00	700.00	0.28	0.00	**700.00**	0.22	0.00
Airland2	15	1480.00	1790	1480.00	0.14	0.00	1480.00	0.12	0.00	**1480.00**	1.71	0.00
Airland3	20	820.00	1790	820.00	0.13	0.00	820.00	0.12	0.00	**820.00**	2.69	0.00
Airland4	20	2520.00	4890	2520.00	1.10	0.00	2520.00	1.45	0.00	**2520.00**	2.78	0.00
Airland5	20	3100.00	6470	3100.00	2.45	0.00	3100.00	4.14	0.00	**3100.00**	3.07	0.00
Airland6	30	24442.00	24442	24442.00	0.08	0.00	24442.00	0.08	0.00	**24442.00**	0.21	0.00
Airland7	44	1550.00	1550	1550.00	0.17	0.00	1550.00	0.14	0.00	**1550.00**	0.61	0.00
Airland8	50	1950.00	18915	1950.00	0.35	0.00	1950.00	0.50	0.00	**1950.00**	22.58	0.00
Airland9	100	5611.70	17602.63	5611.70	3600.00	0.00	5611.99	3600.00	0.01	**5611.70**	49.84	0.00
Airland10	150	12310.70	27201.83	12313.55	3600.00	0.02	12310.70	3600.00	0.00	12359.00	298.25	0.39
Airland11	200	12418.30	33405.36	12418.32	3600.00	0.00	12418.32	3600.00	0.00	**12418.32**	286.69	0.00
Airland12	250	16152.73	43351.63	16152.73	3600.00	0.00	16302.28	3600.00	0.93	16618.20	282.65	2.88
Airland13	500	37268.12	91991.72	37268.12	3600.00	0.00	37294.76	3600.00	0.07	39620.80	305.45	6.31
Minimum					0.08	0.00		0.08	0.00		0.21	0.00
Average					1384.96	0.00		1385.14	0.08		96.67	0.74
Maximum					3600.00	0.02		3600.00	0.93		305.45	6.31

Table 7: Summary of the solution methods for solving benchmark instances with multiple runways.

Method	Number of best solutions obtained	Percent of best solutions obtained	Average time (sec)	Average gap (%)
CPLEX	34	94.44	726.40	0.09
CG	36	100.00	556.03	0.00
RRA	32	88.89	22.15	0.17

4.2 Results for instances with multiple runways

Figure 3 illustrates effectiveness of our RRA method compared to the CPLEX and CG. As the figure shows, while there is not a dramatic change in the RRA's solution quality, its running time is significantly less than CPLEX and CG. Note that due to practical and online applications, it is very important that the RRA algorithm is capable of delivering high quality solutions in a short amount of time.

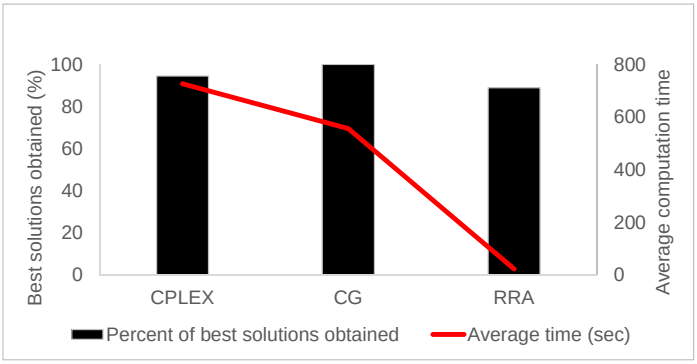

Figure 3: Effectiveness of three solution methods for solving instances of ALP with multiple runways.

Table 7 and Table 8 report summarized and detailed outcomes of CPLEX, CG and the proposed RRA on solving 36 benchmark instances of the ALP with multiple runways. As Table 7 shows, the large number of best known solutions obtained by the CG and CPLEX methods is due to their longer computation times. In particular, our proposed RRA method has an average computation time of less than half a minute, whereas on average, CG and CPLEX require more than 9 and 14 minutes. This distinguishes the RRA as a very efficient method for solving the ALP on multiple runways because (1) it is very fast, more precisely, about 20 times faster than CG and 28 times faster than CPLEX, (2) it delivers the best known solutions for 32 instances, this is for more than 88% of instances, and (3) it has a very small average gap of 0.17%. Notice that the CPLEX, even though it spends about 12 minutes to obtain the solutions, has an average gap of 0.09%. It is important to note that not a single method is able to deliver all best known solutions.

According to Table 8, the RRA is quite effective when the number of runways increases, even for very large instances, implying it is a superior method for scheduling aircraft landings at busy airports benefiting from several runways.

Table 8: Detailed outcomes of three solution methods for solving benchmark instances of the ALP with multiple runways. The best known solutions obtained by the RRA algorithm were highlighted.

					CPLEX			CG			RRA		
Instance	n	m	z^*	FCFS	z	Time (s)	Gap (%)	z	Time (s)	Gap (%)	z	Time (s)	Gap (%)
Airland1	10	2	90	200	90	0.11	0.00	90	0.13	0.00	**90**	0.14	0.00
		3	0	50	0	0.10	0.00	0	0.1	0.00	**0**	0.11	0.00
Airland2	15	2	210	310	210	0.27	0.00	210	0.25	0.00	**210**	0.14	0.00
		3	0	70	0	0.11	0.00	0	0.11	0.00	**0**	0.20	0.00
Airland3	20	2	60	150	60	0.13	0.00	60	0.28	0.00	**60**	0.16	0.00
		3	0	90	0	0.13	0.00	0	0.13	0.00	**0**	0.26	0.00
Airland4	20	2	640	1330	640	5.16	0.00	640	1.21	0.00	**640**	0.41	0.00
		3	130	550	130	0.35	0.00	130	0.67	0.00	**130**	0.48	0.00
		4	0	340	0	0.14	0.00	0	0.14	0.00	**0**	0.28	0.00
Airland5	20	2	650	860	650	2.40	0.00	650	1.02	0.00	**650**	0.27	0.00
		3	170	320	170	0.46	0.00	170	0.35	0.00	**170**	1.01	0.00
		4	0	190	0	0.14	0.00	0	0.15	0.00	**0**	0.28	0.00
Airland6	30	2	554	728	554	0.21	0.00	554	0.28	0.00	**554**	0.26	0.00
		3	0	0	0	0.12	0.00	0	0.11	0.00	**0**	0.29	0.00
Airland7	44	2	0	0	0	0.12	0.00	0	0.1	0.00	**0**	0.26	0.00
Airland8	50	2	135	15115	135	0.64	0.00	135	1.69	0.00	**135**	0.84	0.00
		3	0	14515	0	0.41	0.00	0	0.35	0.00	**0**	0.92	0.00
Airland9	100	2	444.1	10325.96	444.1	1340.53	0.00	444.1	6.32	0.00	**444.1**	1.75	0.00
		3	75.75	8718.4	75.75	4.38	0.00	75.75	4.55	0.00	**75.75**	5.36	0.00
		4	0	8197.53	0	0.64	0.00	0	0.47	0.00	**0**	3.63	0.00
Airland10	150	2	1143.7	13526.62	1143.7	3600.00	0.00	1143.7	3600	0.00	1151.56	5.41	0.69
		3	205.21	11475.79	205.21	46.52	0.00	205.21	32.16	0.00	206.27	19.92	0.52
		4	34.22	10720.51	34.22	7.04	0.00	34.22	6.85	0.00	**34.22**	5.34	0.00
		5	0	10521.95	0	1.46	0.00	0	0.82	0.00	**0**	3.54	0.00
Airland11	200	2	1330.91	18075.68	1330.91	3600.00	0.00	1330.91	3600	0.00	1385.69	8.57	4.12
		3	253.07	15745.94	253.07	373.11	0.00	253.07	1827.94	0.00	**253.07**	41.62	0.00
		4	54.53	14645.84	54.53	53.07	0.00	54.53	53.66	0.00	**54.53**	41.79	0.00
		5	0	14445.33	0	1.61	0.00	0	1.45	0.00	**0**	8.21	0.00
Airland12	250	2	1695.62	24522.92	1698.9	3600.00	0.19	1695.62	3600	0.00	**1695.62**	25.87	0.00
		3	221.97	21468.14	221.97	3600.00	0.00	221.97	14.28	0.00	**221.97**	38.68	0.00
		4	2.44	20293.54	2.44	24.92	0.00	2.44	12.09	0.00	**2.44**	10.27	0.00
		5	0	20040.19	0	2.85	0.00	0	1.95	0.00	**0**	12.94	0.00
Airland13	500	2	3920.39	49890.14	4037.97	3600.00	3.00	3920.39	3600	0.00	**3920.39**	164.65	0.00
		3	673.85	41744.78	673.85	3600.00	0.00	673.85	3600	0.00	679.31	292.46	0.81
		4	89.95	39767.02	89.95	2672.64	0.00	89.95	41.47	0.00	**89.95**	62.18	0.00
		5	0	38330.88	0	10.57	0.00	0	5.96	0.00	**0**	38.99	0.00
Minimum						0.10	0.00		0.10	0.00		0.11	0.00
Average						726.40	0.09		556.03	0.00		22.15	0.17
Maximum						3600.00	3.00		3600.00	0.00		292.46	4.12

4.3 Improvement over FCFS

A routine administered by the air traffic controllers is to sequence aircraft by using the so called first-come-first-serve (FCFS) dispatching rule, and then schedule the landings as soon as the runways are free for a safe landing. According to Figure 4 and Figure 5, which show the cumulative percentage of improvement obtained by the proposed Relax and RRA algorithms over the FCFS dispatching rule, huge improvements in the landing schedules can be obtained by using the methods proposed in this study. The percent of improvement was calculated as

$$\frac{z_{fcfs} - z}{z_{fcfs}} \times 100$$

where z_{fcfs} is the objective function value obtained by the FCFS and z is that of delivered by the either Relax (for the single runway case) or by the RRA (for the multiple runway case).

For example, the total accumulated improvement obtained by using the Relax method instead of the FCFS is more than 600% (see Figure 4). Also, the figure shows that for 38% of instances the amount of improvement is more than 100%, and for 70% of instances it is about 300%. Huge improvements are also shown to be obtained for the case of multiple runways by using the RRA algorithm instead of the FCFS dispatching rule. For instance, as Figure 5 illustrates for 31% of instances the RRA leads to a cumulated improvement in the order of 500%. These numbers further simulate the practical effectiveness of the proposed solutions methods.

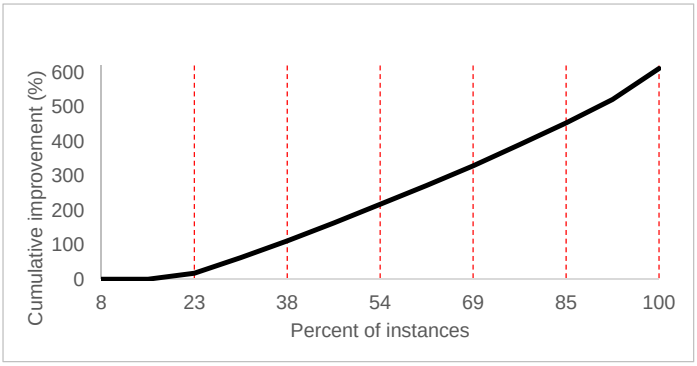

Figure 4: Improvements obtained by the proposed Relax method compared to the FCFS dispatching rule when one runway is available.

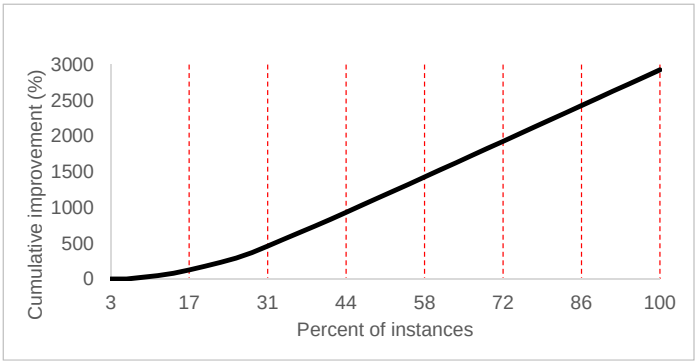

Figure 5: Improvements obtained by the proposed RRA method compared to the FCFS dispatching rule when multiple runways are available.

5 Conclusion

This study investigated a new approach to solve the Aircraft Landing Problem (ALP). We developed two algorithms, which are capable of delivering high quality solutions in a short time. We compared our algorithms with exact methods available in the literature and concluded that while they are quick enough to be implemented in practice, they are very competitive because they are able to obtain the best known solutions for 77% of instances with one runway and 89% of instances with multiple runways. While the available exact methods has an average running time of more than 20 minutes for instances with one runway and more than nine minutes for instances with multiple runways, our proposed methods' average computation time is around 1.5 and 0.5 minutes, respectively. We believe the most important advantages of our algorithms include their simplicity, solutions quality, and fast and robust performance. These characteristics contribute into usability of these algorithms for practical and online applications. Particularly, we showed that our algorithms, while deliver satisfactory solutions very quickly, can significantly improve the first-come-first serve routine, which is practiced by majority of airports.

Acknowledgments

Dr Amir Salehipour is the recipient of an Australian Research Council Discovery Early Career Researcher Award (DECRA) funded by the Australian Government.

References

Abela, J, Abramson, D, Krishnamoorthy, M, De Silva, A, and Mills, G. (1993). "Computing optimal schedules for landing aircraft". In: *proceedings of the 12th national conference of the Australian Society for Operations Research, Adelaide*, pp. 71–90.

Balakrishnan, H. and Chandran, B. G. (2010). "Algorithms for Scheduling Runway Operations Under Constrained Position Shifting". In: *Operations Research* 58(6), pp. 1650–1665.

Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., and Abramson, D (2000). "Scheduling aircraft landingsthe static case". In: *Transportation science* 34(2), pp. 180–197.

Bianco, L., Dell'Olmo, P., and Giordani, S. (1999). "Minimizing total completion time subject to release dates and sequencedependentprocessing times". In: *Annals of Operations Research* 86(0), pp. 393–415.

Ernst, A. T., Krishnamoorthy, M., and Storer, R. H. (1999). "Heuristic and exact algorithms for scheduling aircraft landings". In: *Networks* 34(3), pp. 229–241.

Furini, F., Kidd, M. P., Persiani, C. A., and Toth, P. (2015). "Improved rolling horizon approaches to the aircraft sequencing problem". In: *Journal of Scheduling* 18(5), pp. 435–447.

Ghoniem, A. and Farhadi, F. (2015). "A column generation approach for aircraft sequencing problems: a computational study". In: *Journal of the Operational Research Society* 66(10), pp. 1717–1729.

Ghoniem, A., Farhadi, F., and Reihaneh, M. (2015). "An accelerated branch-and-price algorithm for multiple-runway aircraft sequencing problems". In: *European Journal of Operational Research* 246(1), pp. 34 –43.

Girish, B. (2016). "An efficient hybrid particle swarm optimization algorithm in a rolling horizon framework for the aircraft landing problem". In: *Applied Soft Computing* 44, pp. 200 –221.

Gurobi Optimization, I. (2016). *Gurobi Optimizer Reference Manual*.

ILOG, I. (2012). *IBM ILOG CPLEX V12.4.0: User's manual for CPLEX*.

Pinol, H. and Beasley, J. (2006). "Scatter Search and Bionomic Algorithms for the aircraft landing problem". In: *European Journal of Operational Research* 171(2), pp. 439 –462.

Sabar, N. R. and Kendall, G. (2015). "An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem". In: *Omega* 56, pp. 88 –98.

Salehipour, A. and Ahmadian, M. M. (2017). "A heuristic algorithm for the Aircraft Landing Problem". In: *The 22nd International Congress on Modelling and Simulation (MODSIM2017)*.

Salehipour, A., Naeni, L. M., and Kazemipoor, H. (2009). "Scheduling aircraft landings by applying a variable neighborhood descent algorithm: Runway-dependent landing time case". In: *Journal of Applied Operational Research* 1(1), pp. 39 –49.

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

Salehipour, A., Ahmadian, M. M., and Oron, D. (2018). "Efficient and simple heuristics for the Aircraft Landing Problem". In: *Matheuristic 2018 International Conference*.

Vadlamani, S. and Hosseini, S. (2014). "A novel heuristic approach for solving aircraft landing problem with single runway". In: *Journal of Air Transport Management* 40, pp. 144 –148.

Wen, M., Larsen, J., and Clausen, J. (2005). "An exact algorithm for aircraft landing problem". In: