Abstract

The Aircraft Landing Problem is a classical combinatorial optimization problem in which a set of arriving aircraft must be sequenced and scheduled to land on airport's runways, with the objective of minimizing total deviations from the estimated time of arrival. This work proposes two simple and efficient heuristics to solve the Aircraft Landing Problem on single runway. The proposed heuristics are based on rolling horizon, and iterated greedy frameworks. The motivation behind our work is to develop simple heuristics that are capable of delivering high quality solutions in a short amount of time. This is important because due to the dynamic nature of the problem, i.e. flights arriving early, or late or even being canceled, operators will need to re-solve the problem on a regular basis and update the landing schedules. Therefore, fast algorithms are paramount. The computational experiments over a set of standard instances demonstrate that we fulfilled this aim, and that the proposed heuristics can obtain satisfactory and promising solutions in a short amount of time.

Keywords: Destruction and construction, Sequence relaxation, Iterated greedy, Rolling horizon

Efficient and simple heuristics for the Aircraft Landing Problem

Amir Salehipour [*] Mohammad M. Ahmadian [†] Daniel Oron [‡]

1 Introduction

The Aircraft Landing Problem (ALP) is an important class of scheduling problems, and models problems involving ready times, due dates, sequence-dependent setup times, and penalties for completing jobs either earlier or later than the due dates. It is well-known that problems with these characteristics are strongly NP-Hard (Pinedo, 2008). Due to practical considerations, the objective of the ALP is to minimize total deviations from the target (preferred) landing times by penalizing early and late landings. The amount of deviation from the target landing time between the earliest and latest landing times is the basis for determining the penalty. Hence, the penalty function includes two components: penalizing early landings and penalizing late landings. Not surprisingly, an optimal solution of the ALP has applications in the areas of transportation and scheduling, providing significant benefits to systems experiencing long delays.

Given a set of aircraft $I = \{1, \dots, n\}$, and runways, the aim of the ALP is to obtain a sequence for aircraft landings and the associated 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 α_i holds if the decision variable x_i lies within the range $[E_i, T_i]$, while β_i holds if it lies within the range $[T_i, L_i]$.

In addition, there is a safety *separation* time, which is also known as *wake vortex*, $s_{ij} \in \mathbb{R}^+$ between every two aircraft $i, j \in I^2, i \neq j$, if they land one after another.

Exact solution methods can only be applied to solve small instances of the ALP

[*] ARC DECRA Fellow, University of Technology Sydney, Australia. Email: amir.salehipour@gmail.com
[†] Bu-Ali Sina University, Iran. Email: mmahmadian65@yahoo.com
[‡] The University of Sydney, Australia. Email: daniel.oron@sydney.edu.au

(Beasley et al., 2000; Ernst et al., 1999; Ghoniem and Farhadi, 2015). Therefore, heuristics and meta-heuristics are practical options for large instances. For example, Pinol and Beasley (2006) applied Scatter Search (SS) and Bionomic Algorithms (BA) to instances with up to 500 aircraft. Salehipour et al. (2013) combined Simulated Annealing (SA) and Variable Neighborhood Search (VNS) algorithms into one hybrid algorithm, and applied this to the same instances. Girish (2016) developed several meta-heuristics for the ALP, and reported promising results. Recently, Salehipour and Ahmadian (2017) developed a Variable Neighborhood Descent (VND) algorithm with new relaxation neighborhoods. They reported very promising solutions for the same instances on single runway. Those studies found optimal solution for small instances with up to 50 aircraft, and high quality solutions for larger instances. Special cases of the problem have also been studied; for an example see Salehipour et al. (2009).

This work aims to develop simple heuristics, by combining local search methods and exact solver generated solutions, that are capable of delivering high quality outputs in a short time. Our work is motivated by the dynamic nature of the ALP, as flights may arrive early, be delayed or even end up being canceled. Hence, operators will be required to re-solve the problem on a regular basis and update the landing schedules. Therefore, fast algorithms are crucial. Available heuristic solution methods discussed above require long computation times. Moreover, they include the combination of different neighborhood structures as well as heuristics and meta-heuristics, and hence, their implementation requires advanced algorithmic techniques. We believe that the most important advantage of the algorithms proposed in this study is their conceptual simplicity. This is important because it allows the proposed algorithms to be easily tuned, implemented, and extended to other problems. In addition, available optimization solvers such as CPLEX and Gurobi have greatly improved in recent years, making solver based heuristic algorithms significantly faster and more effective.

This study contributes into solving the ALP by developing simple and efficient algorithms that are capable of delivering high quality solutions in a short time. Section 2 explains those solution methods for the ALP, followed by their computational outcomes in Section 3. Section 4 concludes the paper.

2 Proposed solution methods

This study proposes two heuristic solution methods to solve the Aircraft Landing Problem (ALP). Both methods are based on the Iterative Greedy (IG) framework. The major difference between two methods is the inclusion of the Rolling Horizon (RH) framework in the first method. We name the first method IG-1, and the second IG-2. Both methods are easy to tune and implement, and deliver satisfactory and promising results.

The IG includes two components of destruction and construction, and works by partially destructing, and re-constructing the solution. A similar framework has been applied by Ruiz and Stützle (2007) for the permutation flowshop problem. The IG proposed in this study destructs a small subset of the sequence (a set of aircraft), in which those aircraft are allowed to be re-ordered within the subset, and re-constructs the sequence by solving the problem, in which the landing order of all aircraft except those in the destructed subset are fixed. Therefore, the landing order for the destructed aircraft may change within the subset. Hence, an improved landing sequence may be constructed. This process is iterated until the stopping criterion is met.

The RH has already been applied to solve the dynamic ALP (Bennell et al., 2017). The RH framework decomposes the problem into a set of smaller overlapping sub-problems, and solves a single sub-problem at every iteration. This greatly reduces the computational burden, while ensuring that the obtained solutions are very close to optimal. Two parameters of RH are *window* (w), which defines the size of the sub-problems, and *roll* (r), which defines the overlapping size for sub-problems. Obviously, $r \leq w < n$, and the case $r = w$ results in non-overlapping sub-problems. In the proposed RH for the ALP, both w and r are in terms of number of aircraft rather than time intervals. Because we keep the size of the sub-problems relatively small, we are able to optimally solve each sub-problem by utilizing available exact solvers such as CPLEX (ILOG, 2012), and Gurobi (Gurobi Optimization, 2016). More precisely, the following model (Beasley et al., 2000; Pinol and Beasley, 2006; Salehipour et al., 2013) is solved for each sub-problem.

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

Subject to

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

$$x_i - T_i = \alpha_i - \beta_i \qquad i = 1, \ldots, n \tag{3}$$

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

$$x_j - x_i \geq s_{ij} y_{ij} - M y_{ji} \qquad i, j = 1, \ldots, n, i \neq j \tag{5}$$

$$y_{ij} \in \{0, 1\} \qquad i, j = 1, \ldots, n, i \neq j \tag{6}$$

$$x_i, \alpha_i, \beta_i \geq 0 \qquad i = 1, \ldots, n \tag{7}$$

The objective function (Equation (1)) minimizes the total cost of early and late landings, where c_i^- and c_i^+ are the per unit penalties for earliness and tardiness of aircraft i. Equation (2) ensures every aircraft lands in its time window. Equation (3) links the decision variables x_i and parameters T_i to decision variables α_i and β_i. Given a pair of aircraft, Equation (4) ensures one lands before the other. Equation (5) imposes the separation time condition. Equations (6) and (7) ensure decision variables y_{ij} only take binary values, and decision variables x_i, α_i and β_i only take non-negative values.

Both RH and IG require an initial sequence to begin with. Recently, the study of Salehipour and Ahmadian (2017) used the Earliest Target Landing Time (ETLT) dispatching rule introduced by Salehipour et al. (2013) to generate an initial sequence for the ALP. The ETLT obtains an initial sequence by sorting aircraft on a non-decreasing order of their target landing times. In this study, we used a modification of the NEH construction algorithm of Nawaz et al. (1983) to generate an initial sequence for the ALP. We realized that the NEH produces superior solutions than those produced by the ETLT. Algorithm 1 summarizes the NEH algorithm for the ALP. Algorithm 2 and Algorithm 3 summarize the heuristics IG-1 and IG-2.

Algorithm 1: An NEH-inspired construction algorithm for the Aircraft Landing Problem.

```
Input: An instance of the ALP.
Step 1: Let $\pi' = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, where $T_{\sigma(1)} \leq T_{\sigma(2)} \leq \cdots \leq T_{\sigma(n)}$ be a
sequence for aircraft landings (sorted in non-decreasing order of target landing times);
Step 2: Take the first two aircraft from $\pi'$ and sequence them such that the partial landing
cost is minimized. Maintain the relative position of those two aircraft in the remaining
steps;
Step 3: Consider the next aircraft, and find the minimum landing cost for inserting the
aircraft in the three last possible positions in the partial sequence. Maintain the relative
position of those sequenced aircraft in the remaining steps;
Step 4: Repeat step 3 until a complete sequence is constructed.
return the constructed sequence;
```

Algorithm 2: An Iterative Greedy algorithm (IG-1), with the Rolling Horizon (RH) framework for the Aircraft Landing Problem (ALP).

```
Input: An initial sequence; parameters $w, r$.
while stopping condition is not met do
    $J'$ ←relax($w, r$) (extracting a set of sub-sequences to be destructed);
    optimize($J'$) (constructing the sub-sequence through solving by an exact solver);
    $J^*$ ←update() (updating the landing sequence and schedule);
end
$J^*$ ← post_process($J^*$) (further optimizing the sequence);
Report $J^*$;
```

The solution method IG-1 destructs (relaxes) a set of overlapping sub-sequences $J' \subset J$, and re-constructs (optimizes) J'. To improve the efficiency of IG-1, the sub-sequences are prioritized on their contribution into the objective function value. This ensures the sub-sequence with the highest contribution will be relaxed and optimized early during the algorithm. The post-process ignores this cost contribution scheme.

The solution method IG-2 destructs a sub-sequence through Relax-1 and Relax-2 neighborhoods, and re-constructs a solution. The difference between Relax-1 and Relax-2 lies in the way they carry out the optimization; while Relax-1 imposes re-constructing only the destructed sub-sequence, Relax-2 imposes re-constructing the whole sequence. Relax-1 and Relax-2 are performed several times until the stopping criterion is met.

Note that both IG-1 and IG-2 ensure that the landing order of all aircraft except those in the destructed subset are fixed. Therefore, the connection with the already positioned aircraft is automatically maintained.

Algorithm 3: An Iterative Greedy algorithm (IG-2), with two neighborhood structures of Relax-1 and Relax-2, for the Aircraft Landing Problem (ALP).

```
Input: An initial sequence; parameters $r_1$ and $r_2$.
while stopping condition is not met do
    if $i_1 > 0$ then
        $J' \leftarrow$ Relax-1() (destructing a sub-sequence);
        $i_1 := i_1 - 1$;
    end
    if $i_2 > 0$ then
        $J' \leftarrow$ Relax-2() (destructing a sub-sequence);
        $i_2 := i_2 - 1$;
    end
    Optimize($J'$) (constructing the sub-sequence through solving by an exact solver);
    $J^* \leftarrow$update() (updating the landing sequence and schedule);
    if $i_1 = 0$ then
        $i_2 := r_2$;
    end
    if $i_2 = 0$ then
        $i_1 := r_1$;
    end
end
Report $J^*$;
```

3 Computational results

We tested IG-1 and IG-2 heuristic algorithms on 13 benchmark instances of the Aircraft Landing Problem (ALP) available at OR Library. The instances range from small problems (10 aircraft) to large ones (500 aircraft). In this study, we set $w = 10$, and $r = 8$. Therefore, each sub-problem includes $w = 10$ aircraft, and each rollover includes only 2 aircraft of the last sub-problem. For IG-2, we set $r_1 = 5$, and $r_2 = 20$, i.e. Relax-1 and Relax-2 neighborhoods will be applied 5 and 20 times. Moreover, we initialize $i_1 = 10 + \frac{n}{25}$, and $i_2 = 0$, where n is the number of aircraft. In addition to this, we set $w = \frac{n}{U(15,20)}$ for Relax-1, and $w = \frac{n}{U(20,30)}$ for Relax-2, where $U(a, b)$ represents discrete uniform distribution with parameters a and b. The computation time limit of both algorithms was set to 400 seconds, though, only Airland13, which includes 500 aircraft, benefits from longer computation time (see Table 3).

The reason for these parameter values, which were chosen after extensive testing, is because they result in very good quality solutions in a short time, which is in line with the aims of the study. We should add that greater values of w lead to larger sub-problems but fewer of them, which are more difficult to optimally solve in a short time. Also, smaller values of r leads to a larger number of sub-problems, and we observed that not only this does not contribute to the solution's quality, it greatly increases the computation time. Moreover, due to earliness and tardiness penalties we make the observation that it is not beneficial to sequence aircraft too far from their initial positions, neither earlier nor later, implying larger values for r may not be beneficial here.

In all instances, we consider a single runway. Generally speaking, the ALP with single runway is more difficult to solve than with multiple runways. This is further evidenced by the previous studies (Pinol and Beasley, 2006; Salehipour et al., 2013; Girish, 2016). Finally, optimization solvers CPLEX version 12.4.0 (ILOG, 2012), and Gurobi version 7.5 (Gurobi Optimization, 2016) were used to perform re-construction.

The outcomes of IG-1 and IG-2 heuristics have been depicted in Figures 1 and 2. For comparison purposes, we also reported the outcomes of SA+VND (Simulated Annealing and Variable Neighborhood Descent) and SA+VNS (Simulated Annealing and Variable Neighborhood Search) algorithms of Salehipour et al. (2013), SS (Scatter Search) algorithm of Pinol and Beasley (2006), CG (Column Generation) algorithm of Ghoniem and Farhadi (2015), and HPSO-LS (Hybrid Particle Swarm Optimization and Local search) algorithm of Girish (2016), as appeared in those studies. Figure 1 illustrates the best, average, and worst performance of these solution methods (evaluated by minimum, average, and maximum values of gap) and the number of best solutions obtained by the methods. The gap is computed as $\frac{z^{*}-z_{cplex}}{z_{cplex}} \times 100$, where z^{*} is the best objective function value reported by the method, and z_{cplex} is the best objective function value obtained by the solver CPLEX, and as reported in Table 1. According to Figure 1, CG, HSPO-LS, IG-1 and IG-2 resulted in lower values of gap compared to SA+VND, SA+VNS, and SS. Also, while SA+VND, SA+VNS, SS, and CG cannot obtain more than 9 best solutions (out of 13), the HSPO-LS obtains 13, followed by IG-2, and IG-1, which obtain 12, and 11. The greater number of best solutions obtained by the HSPO-LS has paid its price by significant computation time. This is further illustrated by Figure 2, which shows the performance of the solution methods in terms of the best, average, and worst computation times, and the number of best solutions obtained. According to Figure 2, the HSPO-LS has the highest computation time.

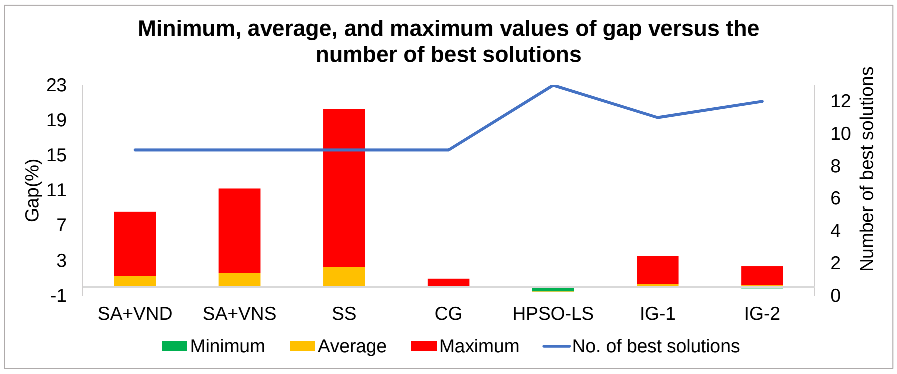

Figure 1: Minimum, average, and maximum values of gap (in percent), and the number of best solutions obtained by different solution methods.

To demonstrate the computational difficulty of solving the ALP, Table 1 reports the details of the solver CPLEX over solving the model discussed in Section 2 on a High Performance Machine.

Table 2 summarizes the outcomes of all solutions methods discussed in this study

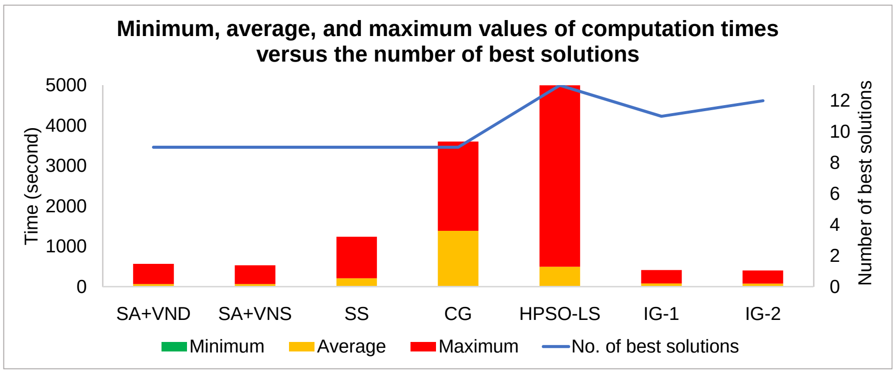

Figure 2: Minimum, average, and maximum values of computation times (in second), and the number of best solutions obtained by different solution methods.

Table 1: Outcomes of the solver CPLEX. The results were obtained by running the CPLEX on a High Performance Machine with 10 threads of Intel®Xeon CPU E5-1650 at 3.50GHz, and 32GB of memory, and allowing the CPLEX to run until it is stopped due to lack of memory.

Instance	n	Best available	CPLEX			
			z^*	Time(s)	Nodes left	Gap(%)
Airland1	10	700.00	700.00	0.66	0	0.00
Airland2	15	1480.00	1480.00	0.49	0	0.00
Airland3	20	820.00	820.00	0.39	0	0.00
Airland4	20	2520.00	2520.00	5.12	0	0.00
Airland5	20	3100.00	3100.00	20.44	0	0.00
Airland6	30	24442.00	24442.00	0.10	0	0.00
Airland7	44	1550.00	1550.00	0.86	0	0.00
Airland8	50	1950.00	1950.00	0.98	0	0.00
Airland9	100	5611.70	5611.70	35439.15	114,888,161	15.08
Airland10	150	12292.20	12292.20	58182.04	98,406,232	50.77
Airland11	200	12418.32	12418.32	30713.95	77,355,422	36.80
Airland12	250	16122.18	16152.73	33798.37	61,092,575	44.07
Airland13	500	37067.11	37268.12	51666.16	38,294,894	53.22

in terms of their capability to obtain the best known solutions. It is clear that the IG-2 heuristic yields high quality solutions in short computational times. Also, as observed by Figure 2 both IG-1 and IG-2 algorithms have an average computation time of less than 1.5 minutes, and are able to obtain 11 and 12 best known solutions (out of 13) for the ALP, respectively. Notice that all solution methods are able to obtain optimal solution for instances with up to 50 aircraft. For the larger instances, while the HSPO-LS of Girish (2016) obtains best solutions for all instances, on average it takes almost seven times longer than the IG-1 and IG-2 algorithms. This is a further testament to the efficiency of the algorithms of this study, particularly IG-2, in delivering high quality solutions for the ALP. Indeed, due to the practical changes in flights schedule, delivering satisfactory schedules in a short time is very important.

Table 2: The solution methods' capability in obtaining the best known solutions.

Instance	n	Best available	Solution Method
Airland1	10	700.00	All
Airland2	15	1480.00	All
Airland3	20	820.00	All
Airland4	20	2520.00	All
Airland5	20	3100.00	All
Airland6	30	24442.00	All
Airland7	44	1550.00	All
Airland8	50	1950.00	All
Airland9	100	5611.70	CPLEX, HPSO-LS, IG-1, IG-2
Airland10	150	12292.20	CPLEX, HPSO-LS, IG-1, IG-2
Airland11	200	12418.32	All
Airland12	250	16122.18	HPSO-LS, IG-2
Airland13	500	37067.11	HPSO-LS

4 Conclusion

Following the difficulty of the Aircraft Landing Problem (ALP), we proposed two efficient heuristics to solve the problem. The major idea behind the algorithms is iterative deconstruction and re-construction of a sub-sequence (of aircraft landings). Our major goal in this study has been developing fast and efficient heuristics for the ALP. We showed that while the proposed algorithms are quick enough to be implemented in practice, they are capable of obtaining high quality solutions. Through solving a set of 13 benchmark instances and comparing across major algorithms, we showed that our algorithms obtain optimal solutions for all eight instances with up to 50 aircraft, and that in a few seconds, and for four larger instances, and that in less than 1.5 minutes on average.

References

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.

Bennell, J. A., Mesgarpour, M., and Potts, C. N. (2017). "Dynamic scheduling of aircraft landings". In: *European Journal of Operational Research* 258(1), pp. 315 – 327. ISSN: 0377-2217.

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. ISSN: 1097-0037.

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. ISSN: 1476-9360.

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. ISSN: 1568-4946.

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

ILOG, I. (2012). *IBM ILOG CPLEX V12.5: User’s manual for CPLEX*.

Nawaz, M., Enscore, E. E., and Ham, I. (1983). “A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem”. In: *Omega* 11(1), pp. 91 –95. ISSN: 0305-0483.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall international series in industrial and systems engineering. Springer. ISBN: 9780387789347.

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. ISSN: 0377-2217.

Ruiz, R. and Stützle, T. (2007). “A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem”. In: *European Journal of Operational Research* 177(3), pp. 2033 –2049. ISSN: 0377-2217.

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 metaheuristic for aircraft landing problem”. In: *Computers & Operations Research* 40(1), pp. 207 –213. ISSN: 0305-0548.