Constraint-Guided Local Search for Single Mixed-Operation Runway

Vahid Riahi, M A Hakim Newton, and Abdul Sattar

Institute for Integrated and Intelligent Systems (IIIS)
Griffith University, Australia
{vahid.riahi}@griffithuni.edu.au
{mahakim.newton,a.sattar}@griffith.edu.au

Abstract. Aircraft sequencing problem (ASP) is to schedule the operation times of departing and arriving aircraft such that their deviation from the desired operation times are minimised. There are two types of hard constraint which make this problem very challenging: time window constraint for the operation time of each aircraft, and minimum separation time between each pair of aircraft. ASP is known to be NP-Hard. Although some progress has been made in recent years in solving ASP, existing techniques still rely on generic algorithms that usually lack problem specific knowledge. This leads to either finding low quality solutions or scrambling with large-sized problems. In this work, we propose a constraint-guided local search algorithm that advances ASP search by injecting the specific knowledge of the problem into its different phases. In the intensification phase, we propose a greedy approach that gives more priorities to aircraft that are more problematic and create more delays. In the diversification phase, we employ a bounded-diversification technique that controls the new position of each selected aircraft and does not allow them to move very far away from their current positions. Computational results show that the proposed algorithm outperforms the existing state-of-the-art methods with considerable margin.

Keywords: Aircraft Scheduling · Constraints · Local Search.

1 Introduction

Air transport has significantly developed over the last few decades with the increase of the demand for the air travelling and freight services. In Australia, 58.93 million passengers travelled by aircraft in 2016, which was 2.5% more compared to that in 2015 [1]. In this situation, air transport systems may face congestion and some have already reached their capacity limits; which causes many problems including flight delays. In Europe in March 2015, 7% more departing flights were delayed by about 4 more minutes compared to that in March 2014 [4].

To overcome such problems and to keep pace with this demand, one possible solution could be to increase the airport capacities by building more runways. However, this process needs availability of the space and more importantly, the

huge investments. For example, Brisbane airport is constructing a new parallel runway; which is expected to take 8 years and AU$1.35 billion investment [2]. It has been showed that operational delays at a major US airport can be reduced up to four hours a day by optimising aircraft landing and takeoff sequences [8] . These results highlight the importance of the aircraft sequencing problem (ASP) and therefore ASP algorithms. By producing optimal or near-optimal sequences of aircraft, these scheduling algorithms will utilise the runways efficiently to increase the overall capacity of an airport and reduce the air traffic.

ASP contains a set of departing and arriving aircraft. Each aircraft belongs to a weight class (e.g. heavy, large, and small). Solving a given ASP instance with a single runway has two steps. The first step is sequencing the aircraft allocated to the runway. The second step is determining the operation times of each aircraft. These two steps must be carried out by satisfying constraints and optimising the objective. There are both **hard** and **soft** constraint in ASP. The hard constraints are ***time window*** and ***minimum separation time***. The former forces each aircraft to be operated within a specified time window, while the latter forces each aircraft to have a minimum separation time with other aircraft. On the other hand, the soft constraint is ***deviation of actual operation times from desired operation times***. All hard constraints must be satisfied in order to obtain a feasible schedule while soft constraints could be violated if necessary, but each instance of violation is penalised. The smaller the overall penalty value, the better the quality of the scheduling. An efficient aircraft sequencing technique can reduce flight times and fuel burn, thereby reduce traffic delays and increase airspace capacity. ASP has been classified as an NP-hard problem [7].

ASP recently has made some progress and a number of methods have been proposed for this problem including scatter search [10], simulated annealing (SA) with variable neighbourhood search (VNS) and variable neighbourhood descent (VND) [16], iterated local search (ILS) [15], and SA and metaheuristic for randomized priority search (Meta-RaPS) [6].

Nevertheless, existing scheduling algorithms take the typical way of using generic techniques that usually lack problem specific structural knowledge, i.e., they use random neighbourhood generation strategies rather than carefully crafted ones or use constraints only in the calculation of the objective function. In this paper, we design a search algorithm injecting the constraint awareness into different steps of the algorithm.

Our search algorithm, called Constraint-Guided Local Search (CGLS), includes two main steps: intensification and diversification. In the intensification phase, we use two neighbourhood operators with problem-specific aircraft selection procedure instead of the typical random one. The idea behind this procedure is that an aircraft with a higher objective value would have priority over an aircraft with a lower objective value (i.e., fix the more problematic part of a solution). In the diversification phase, a bounded-diversification technique is proposed that does not allow the selected aircraft to move very far away from its current position. The idea behind this is that because of the hard constraints,

moving an aircraft at a position that is far from its current position might not be effective and reasonable.

In the rest of the paper, the problem is discussed in section 2, the proposed CGLS technique is described in Section 3, computational results are provided in Section 4, and conclusions are presented in Section 5.

2 Problem Description

Assume there are N aircraft $\{1, \ldots, N\}$ either arriving or departing and one runway to perform the operations on. At any time, the runway can be used by only one aircraft. To solve ASP, we have to determine the *operation time* OT_j of the aircraft j. There are two generic constraint categories: **hard** and **soft** constraints that are to be satisfied to produce a feasible schedule. The aim is to satisfy all the hard constraints and attempt to accommodate the soft constraints as much as possible in order to produce a high-quality schedule.

2.1 Hard constraints

- ***Time window:*** Because of several factors such as fuel restriction, airspeed, and possible manoeuvres, the operation time of each aircraft must lie within a specified time window. This time window is bounded by the *desired operation time* DOT_j and *latest operation time* LOT_j i.e. $OT_j \in [DOT_j, LOT_j]$.
- ***Safety separation time:*** Since each aircraft creates wake turbulence that the following aircraft need to avoid, a certain minimum separation time is required between any pair of aircraft. The separation times depend on the aircraft classes (heavy, large, and small) and the aircraft operation types (landing or takeoff). The separation times are determined by appropriate aviation authorities such as Federal Aviation Administration (FAA) in the United States or Civil Aviation Authority (CAA) in the United Kingdom [5]. Note that although the separation time constraints must hold between each pair of aircraft, it has been showed that using FAA standard, the separation times are automatically satisfied between two aircraft when there are three other aircraft in between. [17].

2.2 Soft constraints

Deviation from desired operation times: For each aircraft j, DOT_j is its desired operation time; which means that operation at that time has no delays and extra fuel burn. However, because of the capacity limit of runways and the hard constraints mentioned, some flights cannot operate at their DOT_j. Therefore, the operation times of some flights are deferred from their desired times. If an aircraft j operates after its desired time DOT_j, it would be penalised by $(OT_j - DOT_j)$.

2.3 Objective function

The objective function of ASP is to minimise total delay cost of the aircraft resulting from the deviation of their operation times from the respective desired times. However, the delay cost of all aircraft is not the same. So, a *penalty weight*

w_j per unit time delay from DOT_j for aircraft j. This *penalty weight* depends on two priorities: *operation priorities* and *aircraft size priorities*. Based on the *operation priorities*, arriving aircraft have greater priorities than departing aircraft because of the higher average fuel burn and safety measures. On the other hand, based on the *size priorities*, heavier aircraft get more weights than the lighter ones owing to again higher average fuel burn and safety measures.

One of the main challenges is how to calculate the operation time OP_j of each aircraft j. Assume π is the current sequence, $[k]$ represents the aircraft at the position k, and $s(j, j')$ shows the separation time between aircraft j and j'. So the operation times of aircraft can be calculated as follows:

$$s([k],[k']) = 0,\ OP_{[k]} = 0 \qquad k < 1 \vee k' < 1 \tag{1}$$

$$OP_{[k]} = \max\{DOT_{[k]}, OP_{[k-1]} + s([k-1],[k]),$$
$$OP_{[k-2]} + s([k-2],[k]), OP_{[k-3]} + s([k-3],[k])\} \qquad \forall k \in [2,n] \tag{2}$$

The objective function is the total weighted tardiness of a schedule TWT $= \sum_{j=1}^{N} w_j(OT_j - DOT_j)$. This objective allows reduction of delays, maximisation of the runway capacity, and reduction of congestion at the airport [14].

3 Methodology

In order to solve this problem, we propose a Constraint-Guided Local Search (CGLS) algorithm. CGLS has two main steps: intensification and diversification. As the main contribution, unlike the most existing techniques in the literature, we use the specific knowledge of the problem to design our algorithm. In the following sections, each step is described in detail.

The proposed local search algorithm is given in Algorithm 1. It starts with an initial solution. The initial solution is then improved by the intensification method. The algorithm next goes through the loop in which the search restarts with the diversification method. The new solution would be considered as the current solution if it is better in terms of the objective value.

Algorithm 1: Proposed CGLS Algorithm

```
π ← Generate an initial solution
π ← Use the intensification method on π
while termination criteria not satisfied do
    π' ← Use the diversification method on π
    π'' ← Use the intensification method on π'
    if TWT(π'') < TWT(π) then π ← π''
return π
```

3.1 Solution representation and initial solution

A single runway ASP solution is represented by a string of numbers containing a permutation of N aircraft, i.e., $\pi = \{[1],[2],...,[N]\}$. The $[k]$ represents the aircraft at the kth position of the permutation. For example, for a problem

with 7 aircraft, one possible solution is $\pi = \{3, 5, 4, 2, 7, 1, 6\}$; which means that aircraft 3 must be operated first, followed by aircraft 5, 4, 2, 7, 1 and aircraft 6.

As the initial solution, we use the simplest and the most common heuristic of aircraft sequencing, first-come-first-served (FCFS). In this method, the permutation of aircraft is based on a non-decreasing order of their *desired operation time*. Note that this very simple heuristic is still used in the air traffic control these days, e.g., in Doha International Airport [5]. However, it is not an efficient heuristic and can lead to the waste of resources and make the congestion in the terminal area severer [3]. Using this heuristic in the initialisation of the proposed algorithm can help find out how much improvement would be obtained by using the proposed method instead of the typical FCFS.

3.2 Intensification Method

To intensify the search, we propose an intensification method that is made up of two neighbourhood operators, instead of a single one. The reason is that different neighbourhood operators produce different landscapes and hence different local optima. We use insert and swap operators since they are widely used when solutions are represented as permutations e.g., in the flowshop scheduling problems [11, 12] and the order scheduling problems [13].

Let π be a permutation of the given N aircraft. In the operator $\mathsf{Insert}(\pi, j, k)$, aircraft at position j is selected and then inserted at a different position k. In the $\mathsf{Insert}(\pi, j)$, aircraft at position j is inserted at all k ($k \neq j$). On the other hand, in the $\mathsf{Swap}(\pi, j, k)$ operator, an aircraft at position j is exchanged with another aircraft at position k. However, in the $\mathsf{Swap}(\pi, j)$ operator, an aircraft at position j is exchanged with all aircraft at positions k ($k \neq j$).

In this paper, we use $\mathsf{Insert}(\pi, j)$ and $\mathsf{Swap}(\pi, j)$ operators mentioned in an iterative procedure. It means that they would be applied for all N aircraft in the permutation π, one by one, in a given order and as soon as a better solution is found, it would be considered as the current solution and the procedure is restarted with the new solution. We refer them to $\mathsf{Insert}(\pi)$ and $\mathsf{Swap}(\pi)$.

***Greedy Aircraft Selection*:** Our main contribution in the intensification method is to employ a greedy aircraft selection by using a constraint guidance. In the proposed greedy selection procedure, first, the weighted tardiness (WT_j) of each aircraft j in the current solution π is calculated. Then, the aircraft are sorted in a non-increasing order of WT_j in a reference list $\pi^{\mathcal{L}}$. Next, in the $\mathsf{Insert}(\pi)$ and $\mathsf{Swap}(\pi)$ operators, the aircraft are selected based on their order in the reference list $\pi^{\mathcal{L}}$. For instance, suppose for a problem with 6 aircraft, the current sequence is $\pi = \{2, 3, 6, 4, 5, 1\}$ and the reference list is $\pi^{\mathcal{L}} = \{4, 2, 6, 1, 3, 5\}$. The $\mathsf{Insert}(\pi)$ or $\mathsf{Swap}(\pi)$ operator first selects aircraft 4 for insertion or swap process from the sequence π. Then, it selects aircraft 2 from the sequence π. This process is continued until all aircraft in the reference list $\pi^{\mathcal{L}}$ are selected. The idea behind this greedy procedure is that aircraft with higher objective values should get more priorities over aircraft with lower objective values. Our idea is to reschedule these aircraft and thus fix the sequence.

The proposed $\mathsf{Insert}(\pi)$ and $\mathsf{Swap}(\pi)$ operators with greedy aircraft selection are given in Algorithm 2. With the use of the $\mathsf{Insert}(\pi)$ and the $\mathsf{Swap}(\pi)$ operators,

the proposed intensification method is shown in Algorithm 3. At each iteration, it first applies N_1 on the current solution π. If the new solution obtained by N_1 is better than the current solution, it would be considered as the current solution and the process is again continued with N_1; otherwise the algorithm moves to N_2. The current solution would be updated if the new solution obtained by N_2 is better and algorithm also goes back to N_1; otherwise intensification phase is finished. Note that, in the intensification method, $\mathsf{Insert}(\pi)$ and $\mathsf{Swap}(\pi)$ operators are selected as N_1 and N_2 respectively based on the results obtained in the literature [18].

Algorithm 2: $\mathsf{Insert}(\pi)$ and $\mathsf{Swap}(\pi)$ operators

1 Let π be the input solution
2 **foreach** aircraft j, calculate weighted tardiness WT_j, and sort them in the non-increasing order of WT_j to get a reference list $\pi^{\mathcal{L}} = (\pi_1^{\mathcal{L}}, \pi_2^{\mathcal{L}}, ..., \pi_N^{\mathcal{L}})$.
3 **for** k = *1 to* N **do**
4 $\quad \pi_j \leftarrow$ The position of the aircraft $\pi_k^{\mathcal{L}}$ in π
5 $\quad$ Apply $\mathsf{Insert}(\pi, \pi_j)$ or $\mathsf{Swap}(\pi, \pi_j)$ and take the permutation π' with the lowest total weighted tardiness.
6 $\quad$ **if** π' has a lower objective than π **then** let $\pi = \pi'$ and go to Step 2
7 **end**
8 **return** π as the output solution

Algorithm 3: Intensification Method

1 **Input:** sequence π
2 Set $\mathsf{Insert}(\pi)$ as N_1 and $\mathsf{Swap}(\pi)$ as N_2. Also $l = 1$
3 **while** $l \leq 2$ **do**
4 $\quad$ Find the best neighbor π' of π in $N_l(\pi)$.
5 $\quad$ **if** π' is better than π **then** Set $\pi = \pi'$ and $l = 1$
6 $\quad$ **else** $l = l + 1$
7 **end**
8 **Output:** sequence π

3.3 Diversification Method

The proposed algorithm uses a diversification method to avoid getting stuck and convergence towards local optima and also to explore new areas in the solution space. Diversification method helps the algorithm generate new solutions for the intensification method by modifying the current solution instead of a fully random solution. The diversification procedure includes a number of moves, diversification strength λ, that are applied to the current local optimum. In this paper the diversification method also used two neighbourhood operators: $\mathsf{Swap}(\pi, j, k)$ and $\mathsf{Insert}(\pi, j, k)$. In this phase, for each diversification move, with 50%-50% probabilities, we apply either $\mathsf{Swap}(\pi, j, k)$ or $\mathsf{Insert}(\pi, j, k)$ operators.

The value of the parameter λ is very important. A small λ may lead to the stagnation of the search and cycling among the previously visited solutions.

On the other hand, a large λ may lead the algorithm to conduct like a random restart algorithm which in most cases generates low quality solutions. Therefore, we carefully calibrate the parameter λ which can be seen in Section 4.1.

***Bounded-Diversification Technique*:** Unlike the typical diversification procedure that moves the selected aircraft to the completely randomly selected positions, we inject the problem specific knowledge into this method to find diverse as well as reasonable positions for the selected aircraft. As mentioned already, ASP has two types of hard constraint including *time window* constraint that forces each aircraft j to be operated within a window, i.e, $OT_j \in [DOT_j, LOT_j]$. Being operated the more closer to DOT_j leads to the less penalty value. Therefore, moving an aircraft to a position that is far from its current position could not be very effective and reasonable. Therefore, in this paper, we propose a bounded-diversification technique that does not allow a selected aircraft to move far away from its current position. To that end, we introduce a parameter γ that controls the position of each selected aircraft. In detail, when an aircraft at position j is selected for diversification, it could be moved just to the position k such that $\mathsf{max}(1, j-\gamma) \leq k \leq \mathsf{min}(N, j+\gamma)$. Similar to λ, this parameter is also carefully calibrated which can be seen in Section 4.1. The procedure of the proposed diversification method is given in Algorithm 4.

Algorithm 4: Proposed bounded diversification method

```
Input: Solution π, the diversification strength λ, the diversification bound γ.
for h = 1 to λ do
    j ← pick a random position
    k ← pick another random position from [max(1, j − γ), min(N, j + γ)]
    if rand() ≤ 0.5 then π ← Insert(π, j, k) else π ← Swap(π, j, k)
end
return π
```

4 Experimental Results

In order to evaluate the performance of the proposed algorithm, we use 20 well-known instances generated based on the Doha International Airport parameters [5]. These instances are made up of 50 aircraft and time windows of 30 minutes. We compare our algorithm with ILS algorithm [15] (called here as ILS-SK) as one of the leading algorithms for the single runway ASP. The ILS-SK algorithm uses a variant of FCFS as initialisation. In this paper, to have a fair comparison, we use the FCFS as initialisation of the ILS-SK as well. Both algorithms have been implemented in C++ language and on top of the constraint-guided local search system, Kangaroo [9]. The functions and the constraints are defined by using invariants in Kangaroo. Invariants are special constructs that are defined by using mathematical operators over the variables. Algorithms are also tested on the same computer.

To compare the performance of the algorithms, we use the relative percentage deviation $RPD = \frac{\mathrm{TWT}^A - \mathrm{TWT}^{BEST}}{\mathrm{TWT}^{BEST}} \times 100$ where TWT^A is the total weighted

tardiness obtained by algorithm A and TWT^{BEST} is the best total weighted tardiness achieved by any of the algorithms compared. We run each algorithm on each instance 5 times and compute average RPD (ARPD) over the 5 runs. We also compute a further average of RPDs or ARPDs over all instances in a benchmark set. As a stopping criterion, the algorithms were run for $20N$ ms CPU time.

4.1 CGLS Parameter Calibration

The proposed CGLS contains two parameters: the diversification strength λ, and the diversification bound γ. To analyse the effect of these two parameters, a full factorial design is used by considering 3 different values for each parameter: $\lambda \in \{10, 20, 30\}$ and $\gamma \in \{3, 4, 5\}$. For this experiment, we randomly select 8 instances from those 20 instances in our benchmark. Our algorithm is run 5 times for each of the $3 \times 3 = 9$ settings and for each instance with the same stopping criterion as already mentioned.

The 95% confidence interval plots of the parameters are shown in Fig. 1. The results of Fig. 1 says that CGLS algorithm is robust with respect to λ and γ as the tested values are statistically equivalent and each of them could be selected. However, since the λ and γ have lower ARPD in 20 and 4 respectively, these values are selected for further experiments.

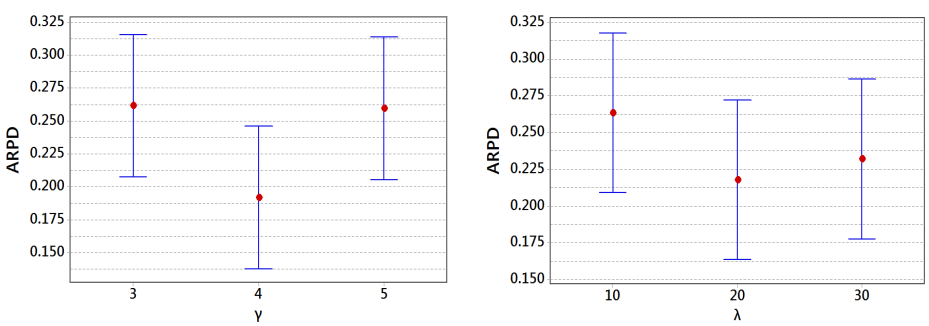

Fig. 1. Mean and 95% confidence intervals for parameters.

4.2 Effectiveness of Multi Neighbourhood

The proposed intensification method includes two neighbourhoods N_1 and N_2. In this paper, we use insertion and swap operators with greedy aircraft selection, GI and GS respectively. In this section, we are to evaluate the efficiency of the greedy neighbourhoods against the random insertion and swap operators, RI and RS, and also to find the best order for the neighbourhood operators mentioned. To that end, the following four cases are considered:

1. **Case 1:** Consider GI as N_1 and GS as N_2.
2. **Case 2:** Consider GS as N_1 and GI as N_2.
3. **Case 3:** Consider RI as N_1 and RS as N_2.

4. **Case 4:** Consider RS as N_1 and RI as N_2.

In this experiment, the proposed CGLS is tested by considering each of the cases mentioned as the intensification method on those 8 instances used already for parameter tuning. The 95% confidence interval plot for each case is given in Fig. 2. From this figure, it can be seen that cases 1 and 2 are significantly better than cases 3 and 4. It can be concluded that the proposed problem-dependent greedy strategies for Insertion and Swap moves statistically outperform the random cases. In addition, although cases 1 and 2 are statistically equivalent, we use case 1 for the intensification phase due to its lower ARPD.

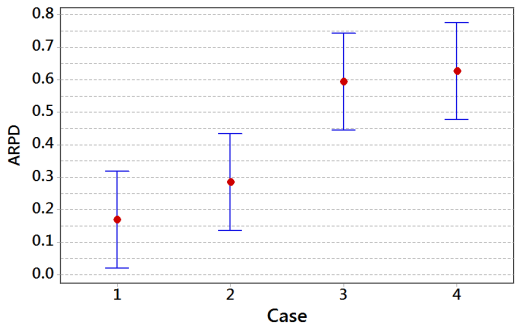

Fig. 2. 95% Confidence intervals for CGLS variants.

4.3 Effectiveness of CGLS Components

CGLS has two main contributions: a new constraint based greedy aircraft selection in the neighbourhood operators of the intensification method, and a constraint based bounded-diversification procedure. To test the effectiveness of each component mentioned, we create three variants of CGLS as follows:

1. **CGLS:** Proposed CGLS that includes both greedy intensification and bounded-diversification.
2. **CGLS_R:** CGLS but greedy intensification is replaced by a random one.
3. **CGLS_NB:** CGLS but no bound in the diversification phase.

The algorithms are tested on the 8 instances which are the same as the ones for parameter tuning. A 95% confidence interval plot in Fig. 3 is carried out to show the effectiveness of the three variants. Note that non-overlapping confidence intervals of each two methods represent a statistically significant difference between them. From Fig. 3, we can see that both new components significantly affect the performance of CGLS. Among these two components, the bounded-diversification is more crucial as the algorithm obtained worse performance with the absence of this method.

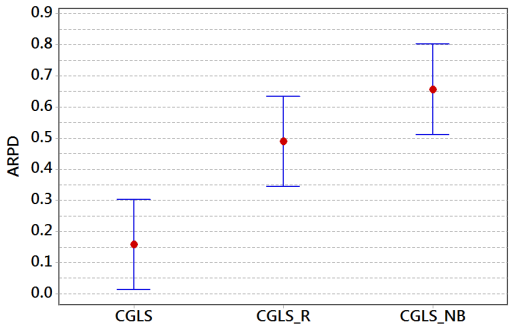

Fig. 3. 95% Confidence interval for CGLS variants.

4.4 Comparison with FCFS Method

As mentioned before, the first-come-first-served (FCFS) heuristic is the simplest and the most common heuristic for aircraft sequencing, and is still applied in the air traffic control these days, e.g., in Doha International Airport [5]. As a result, comparing CGLS with FCFS can show how much the proposed CGLS improves over FCFS. The results are shown in Table 1. As can be seen from this table, CGLS hugely outperforms the FCFS obtaining ARPD of 0.157% compared to 99.353% of FCFS.

Table 1. Comparison of CGLS and FCFS algorithms.

instance	1	2	3	4	5	6	7	8	9	10	11
FCFS	155.67	138.61	212.54	160.93	178.10	102.55	133.63	75.99	90.06	82.97	68.75
CGLS	**0.29**	**0.60**	**0.26**	**0.14**	**0.39**	**0.00**	**0.07**	**0.20**	**0.23**	**0.00**	**0.23**

instance	12	13	14	15	16	17	18	19	20	Average
FCFS	67.29	71.41	79.03	43.40	59.31	60.14	75.83	69.23	61.64	99.35
CGLS	**0.10**	**0.06**	**0.07**	**0.00**	**0.08**	**0.05**	**0.28**	**0.04**	**0.13**	**0.16**

4.5 Comparison with the State-of-the-art Method

We compare the results of CGLS with the results of ILS-SK algorithm [15] shown in Table 2. In this table, besides the ARPD, we also show the number of times each algorithm finds the TWT^{BEST} (the best total weighted tardiness achieved by any of the tested algorithms) for each instance out of 5 runs. As can be seen, CGLS outperforms ILS-SK i.e., it achieves lower ARPD in 19 instances out of 20. In addition, except in instance 7, CGLS obtains the TWT^{BEST} in all instances at least once, while ILS-SK finds the TWT^{BEST} only in 6 instances out of the 20. To examine the difference of the algorithms statistically, we also perform a student t-test with significance level of $\alpha = 0.05$. Statistical results confirm a significant difference between CGLS and ILS-SK since p-value $= 0.00 < 0.05$

Table 2. Comparison of CGLS and ILS-SK algorithms

Instance	CGLS		ILS-SK	
	ARPD	#best	ARPD	#best
1	**0.287**	3	3.222	0
2	**0.597**	1	1.119	0
3	**0.257**	3	1.427	0
4	**0.140**	2	1.440	1
5	**0.387**	2	4.230	0
6	**0.000**	5	0.361	1
7	**0.067**	0	0.149	2
8	**0.199**	2	0.668	0
9	**0.229**	1	1.415	0
10	**0.000**	5	0.641	0
11	0.225	2	**0.149**	1
12	**0.103**	1	1.117	0
13	**0.061**	2	0.240	0
14	**0.071**	2	0.150	1
15	**0.000**	5	0.207	0
16	**0.007**	3	0.292	0
17	**0.053**	1	0.244	1
18	**0.275**	2	0.431	0
19	**0.044**	2	0.751	0
20	**0.134**	1	0.355	0
Average	**0.157**		0.930	

5 Conclusion

In this paper, we proposed a Constraint-Guided Local Search (CGLS) for aircraft sequencing problem (ASP) with a single mixed-operation runway considering the total weighted tardiness as criterion. Unlike the other existing algorithms in the literature, CGLS injects the specific knowledge of the problem in its different phases. In the intensification phase, it uses a greedy approach that gives more priorities to aircraft that are more problematic and create more delays. In the diversification phase, it employs a bounded-diversification technique that controls the new position of each selected aircraft in this phase and do not allow aircraft to move very far away from their current position. The results show that the good performance of the proposed CGLS hugely depends on these two proposed main contributions. Moreover, the computational results show that CGLS significantly outperforms existing state-of-the-art methods.

References

1. bitre: Bureau of Infrastructure, Transport and Regional Economics (2017), `https://bitre.gov.au/`

2. BNE: Brisbane Airport Corporation (2017), http://www.bne.com.au/
3. Caprı, S., Ignaccolo, M.: Genetic algorithms for solving the aircraft-sequencing problem: the introduction of departures into the dynamic model. Journal of Air Transport Management **10**(5), 345–351 (2004)
4. Eurocontrol: Eurocontrol - Driving excellence in ATM performance (2017), http://www.eurocontrol.int/
5. Farhadi, F., Ghoniem, A., Al-Salem, M.: Runway capacity management–an empirical study with application to doha international airport. Transportation Research Part E: Logistics and Transportation Review **68**, 53–63 (2014)
6. Hancerliogullari, G., Rabadi, G., Al-Salem, A.H., Kharbeche, M.: Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem. Journal of Air Transport Management **32**, 39–48 (2013)
7. Lawler, E.L., Lenstra, J.K., Kan, A.R.: Recent developments in deterministic sequencing and scheduling: a survey. In: Deterministic and stochastic scheduling, pp. 35–73. Springer (1982)
8. Mehta, V., Reynolds, T., Ishutkina, M., Joachim, D., Glina, Y., Troxel, S., Taylor, B., Evans, J.: Airport surface traffic management decision support: Perspectives based on tower flight data manager prototype. Tech. rep. (2013)
9. Newton, M.H., Pham, D.N., Sattar, A., Maher, M.: Kangaroo: An efficient constraint-based local search system using lazy propagation. In: International Conference on Principles and Practice of Constraint Programming. pp. 645–659. Springer (2011)
10. Pinol, H., Beasley, J.E.: Scatter search and bionomic algorithms for the aircraft landing problem. European Journal of Operational Research **171**(2), 439–462 (2006)
11. Riahi, V., Khorramizadeh, M., Newton, M.H., Sattar, A.: Scatter search for mixed blocking flowshop scheduling. Expert Systems with Applications **79**, 20–32 (2017)
12. Riahi, V., Newton, M.H., Su, K., Sattar, A.: Local search for flowshops with setup times and blocking constraints. In: ICAPS. pp. 199–207 (2018)
13. Riahi, V., Polash, M., Newton, M.H., Sattar, A.: Mixed neighbourhood local search for customer order scheduling problem. In: Pacific Rim International Conference on Artificial Intelligence. pp. 296–309. Springer (2018)
14. Rodríguez-Díaz, A., Adenso-Díaz, B., González-Torre, P.L.: Minimizing deviation from scheduled times in a single mixed-operation runway. Computers & Operations Research **78**, 193–202 (2017)
15. Sabar, N.R., Kendall, G.: An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem. Omega **56**, 88–98 (2015)
16. Salehipour, A., Modarres, M., Naeni, L.M.: An efficient hybrid meta-heuristic for aircraft landing problem. Computers & Operations Research **40**(1), 207–213 (2013)
17. Sherali, H., Ghoniem, A., Baik, H., Trani, A.: A combined arrival-departure aircraft sequencing problem. Manuscript, Grado Department of Industrial and Systems Engineering (0118). Virginia Polytechnic Institute and State University **250** (2010)
18. Soykan, B., Rabadi, G.: A tabu search algorithm for the multiple runway aircraft scheduling problem. In: Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling, pp. 165–186. Springer (2016)