Constraint Guided Search for Aircraft Sequencing

Vahid Riahi, M A Hakim Newton, M M A Polash, Kaile Su, Abdul Sattar

Institute for Integrated and Intelligent Systems (IIIS), Griffith University, Australia

Abstract

Aircraft sequencing problem (ASP) is an NP-Hard problem. It involves allocation of aircraft to runways for landing and takeoff, minimising total tardiness. ASP has made significant progress in recent years. However, within practical time limits, existing incomplete algorithms still either find low quality solutions or struggle with large problems. One key reason behind this is the typical way of using generic heuristics or metaheuristics that usually lack problem specific structural knowledge. As a result, existing such methods use either an exhaustive or a random neighbourhood generation strategy. So their search guidance comes only from the evaluation function that is used mainly after the neighbourhood generation. In this work, we aim to advance ASP search by better exploiting the problem specific structural knowledge. We use the constraint and the objective functions to obtain such problem specific knowledge and we exploit such knowledge both in a constructive search method and in a local search method. Our motivation comes from the constraint optimisation paradigm in artificial intelligence, where instead of random decisions, constraint-guided more informed optimisation decisions are of particular interest. We run our experiments on a range of standard benchmark problem instances that include instances from real airports and instances crafted using real airport parameters, and contain scenarios involving multiple runways and both landing and takeoff operations. We show that our proposed algorithms significantly outperform existing state-of-the-art aircraft sequencing algorithms.

Keywords: Aircraft Sequencing; Runway Operations; Local Search.

1. Introduction

Airline industries today play a key role in transportation of people and freight. About 58.93 million passengers travelled in year 2016 in Australia, which was 2.5% more compared to that in year 2015 (bitre, 2017). World wide more than 6 billion passengers travelled with domestic and international flights in 2012, which is anticipated to be more than double by 2025 (ACI, 2012). Unfortunately, in Europe in March 2015, 7% more departing flights were delayed by about 4 more minutes compared to the 28% departing flights with an average delay of 24 minutes in March 2014. Moreover, 32% arrival flights in the same region were delayed with an average of 32 minutes in March 2015 compared to 25 minutes in March 2014 (Eurocontrol, 2017). Although weather condition is a great factor for these delays, 32% of flight delays between 2010 and 2015 in the

Email addresses: vahid.riahi@griffithuni.edu.au (Vahid Riahi), mahakim.newton@griffith.edu.au (M A Hakim Newton), mdmasbaulalam.polash@griffithuni.edu.au (M M A Polash), k.su@griffith.edu.au (Kaile Su), a.sattar@griffith.edu.au (Abdul Sattar)

US were because of air traffic volume (BTS, 2017). Flight delays cause additional fuel burning, disrupts flight connections, cause passenger dissatisfactions, and severely affect crew scheduling. According to Federal Aviation Administration (FAA), the total cost of all US air transportation delays in 2007 was estimated to be $31.2 billion (Ball et al., 2010). Flight delays are thus a very important growing challenge that needs to be tackled with proper emphasis.

To keep pace with the increasing demand for air transportation, airports have been increasing capacities by building more runways or allowing more flights per runway per unit of time. However, building more runways needs huge investments and availability of space. For example, Brisbane airport in Australia has been constructing a new parallel runway, which is expected to take 8 years with AU$1.35 billion investment (BNE, 2017). On the other hand, allowing more flights arbitrarily is simply not possible because of the technical restrictions. Each aircraft creates wake turbulence that the subsequent aircraft must avoid, otherwise dangerous consequences might be imminent. Mandatory time separations between pairs of flights are therefore enforced by aviation authorities such as Federal Aviation Administration (FAA) in the United State or Civil Aviation Authority (CAA) in the United Kingdom. However, the separation times essentially restrict the available choices for aircraft sequencing and cause delays in landing and takeoff.

Fortunately, Mehta et al. (2013) showed that given a significant level of diversity of aircraft, operational delays at a major US airport can be reduced approximately up to four hours a day by optimising landing and takeoff sequences. As a result, developing effective aircraft scheduling algorithms to produce optimal or near-optimal sequences of aircraft has appeared to be a promising technique. These scheduling algorithms will utilise the runways more efficiently to increase the overall capacity of the airports and to smoothen the flow of air traffic. In this work, we develop efficient constraint optimisation search algorithms for *aircraft sequencing problems* (ASP). For this, we use existing aircraft sequencing models from the literature.

Each ASP comprises a set of departing aircraft and a set of arriving aircraft. Each landing or takeoff operation must be performed within a given time window (known as the *time window constraint*). Each aircraft belongs to a class (e.g. heavy, large, and small). Each pair of flights depending on their aircraft classes must be separated by a given time period (known as the *separation time constraint*). Besides the aforementioned two hard constraints that must be satisfied, ASPs might have a number of soft constraints that should be satisfied as much as possible. The soft constraints include restricting the number of flights allocated to a particular runway, scheduling each landing or takeoff at the corresponding given ready time, consideration of safety issues and fuel burnt, prioritising all landing aircraft over departing ones and all heavier aircraft over lighter ones. Interestingly, each soft constraint can be considered as an objective function and vice versa. Among many possible alternatives, as an objective function in ASP and also in this paper, *total weighted tardiness* of all flights is minimised. Obviously, the smaller the tardiness of a given schedule, the better the quality of the schedule. ASP has been classified as an NP-hard problem (Lawler et al., 1982).

ASP has made significant progress in recent years. However, within practical time limits, existing incomplete algorithms still either find low quality solutions or struggle with large problems. One key reason behind this is the typical way of using generic heuristics or metaheuristics that usually lack problem specific structural knowledge available from the constraints or the objectives of the problem instance. Moreover, existing such methods, for example those by (Sabar and Kendall, 2015) and (Salehipour et al., 2013) among others, use either an exhaustive or a random neighbourhood generation strategy. So their search guidance comes only from the evaluation

function that is used mainly after the neighbourhood generation. As a result, when such a method performs well, it is typically not known why the method performs well and what problem specific aspects are behind the performance. To be more specific, in ASP, the time window or separation time constraints, or the total weighted tardiness objective should affect decision making during the search process including the neighbourhood generation; unfortunately, this is not the case with the existing approaches.

In this work, we aim to advance ASP search by better exploiting the problem specific structural knowledge. We use the constraint and the objective functions to obtain such problem specific knowledge and we exploit such knowledge both in a constructive search method and in a local search method. Our motivation comes from the constraint optimisation paradigm in artificial intelligence, where instead of random decisions, constraint-guided more informed optimisation decisions are of particular interest; this allows finding justifications for the decisions. Our way of using problem specific knowledge is in a way generic enough to be used in similar other problems. To be more specific, in ASP, we use the objective function to optimise even the partial solutions during the constructive search. During the local search, we design operators taking the constraints into account, we select runways to generate revised schedules taking the objective function into account.

We run our experiments on a range of standard benchmark problem instances that include actual instances from a real airport, crafted instances using parameters from a real airport, and contain scenarios involving multiple runways and both landing and takeoff operations. We show that our proposed algorithms significantly outperform existing state-of-the-art aircraft sequencing algorithms. Although our algorithm is for a static ASP model where all landing or takeoff aircraft are given before hand, it is efficient enough to be used in a dynamic model that considers time windows of practical lengths.

The rest of the paper is organised as follows: Section 2 describes the problem; Section 3 explores existing state-of-the art methods; Sections 4, 5, and 6 respectively describe our solution representation method, the proposed constructive search method, and the proposed local search method; Section 7 presents our experimental results; and Section 9 presents our conclusions.

2. Problem Description

For problem formulation, we refer the readers to (Farhadi, 2014; Hancerliogullari et al., 2013), which provide a mixed-integer programming (MIP) model for ASP. Only for notations, in this paper in Section 2.1, we provide a compact constraint optimisation model, which is equivalent to the MIP model. It is worth noting again that our contribution in this paper is in the use of constraint-based strategies within generic heuristic or metaheuristic search algorithms. While experienced reader could quickly look into Section 2.1 and go to Section 3, for other readers, we also provide details of the constraint and the objective functions in Sections 2.2–2.4.

2.1. Constraint Optimisation Model

Assume an ASP with N aircraft $\{1, \ldots, N\}$ arriving or departing and M runways $\{1, \ldots, M\}$ to perform landing or takeoff operations on. At any time, each runway can be used by only one aircraft and each aircraft can operate on only one runway. There are W aircraft class operation weights $\{1, \ldots, W\}$ for prioritisation of one aircraft's operation over another's. The aircraft class operation weight w_j for an aircraft j depends both on the aircraft class (e.g. small, large, or heavy) and the type of operation (e.g. landing or takeoff). For each pair of aircraft operating on the same

runway, there is a minimum separation time $s(w, w')$, where w and w' are the respective aircraft class operation weights. Table 1 summarises the separation times that we use in this work. Each aircraft j has a time window $[r_j, l_j]$ defined by ready time and latest time in which the respective landing or takeoff operation for the aircraft is to be performed. Because of separation times, an aircraft might not be scheduled at its ready time. If an aircraft j is scheduled to operate at time $t_j \in [r_j, l_j]$, its weighted tardiness (WT) in that case would be $w_j(t_j - r_j)$. To solve the ASP described above, for each aircraft j, we have to determine the runway $\rho_j \in [1, M]$ to perform the operation on and the start time t_j to perform the operation at. The objective in this case is to minimise the total weighted tardiness (TWT) over all aircraft to be scheduled.

Table 1: Aircraft classes (e.g. small, large, heavy), aircraft operations (e.g. takeoff, landing), aircraft class operation weights (1–6), and safety separation times (in seconds) between each pair of aircraft operating on the same runway based on the FAA standard.

Operation			takeoff			landing		
	Class		small	large	heavy	small	large	heavy
		Weight	1	2	3	4	5	6
takeoff	small	1	60	60	60	60	60	60
	large	2	60	60	60	60	60	60
	heavy	3	120	120	90	60	60	60
landing	small	4	75	75	75	82	69	60
	large	5	75	75	75	131	69	60
	heavy	6	75	75	75	196	157	96

rows: leading aircraft columns: following aircraft

For completeness of the solutions in terms of the optimal objective values, below we provide the constraint optimisation model.

$$\text{find } \forall_{j\in[1,N]} t_j \in [r_j, d_j] \qquad \text{ready time}$$

$$\text{and } \forall_{j\in[1,N]} \rho_j \in [1, M] \qquad \text{runway assigned}$$

$$\text{minimising TWT} = \textstyle\sum_{j=1}^{N} w_j(t_j - r_j) \qquad \text{objective function}$$

$$\text{subject to } \forall_{j\in[1,N]}(t_j \in [r_j, d_j]) \qquad \text{time windows}$$

$$\forall_{i\in[1,M]} \forall_{j\neq j' : \rho_j = \rho_{j'} \wedge t_j < t_{j'}} (t_{j'} - t_j > s(w_j, w_{j'})) \qquad \text{separation times}$$

With the complete model described above, in this work, we add runway capacity constraints as soft constraints. Arguably in theoretically optimal solutions, distributions of aircraft operations over runways will be closed to uniform rather than being very highly skewed. Moreover, for various practical reasons (see Section 2.5), a (nearly) uniform distribution of aircraft operation over the runways is desired. So the soft constraint in this case is to keep C_i the number of aircraft allocated to each runway i in the range $[\lfloor \frac{N}{M} \rfloor - \nu, \lceil \frac{N}{M} \rceil + \nu]$, where v is a given parameter. This soft constraint might help save considerable amount of search. Moreover, different combination of capacity limits on the runways could be tried in parallel and the best result obtained could be returned as the final result.

2.2. Time Window Constraints

When an aircraft j enters the radar range for landing or departure, the air traffic controller assigns a runway to it and a start time for landing/takeoff. That start time *must* be within a specified *time window*, bounded by the *earliest* and *latest* landing/takeoff time e_j and l_j i.e. $t_j \in [e_j, l_j]$. The earliest time is calculated from the minimum time needed for the aircraft to approach the runway from its current position using its maximum speed. The latest time is based on the longest time the aircraft can keep flying depending on its operational and service level constraints. Also, each landing or takeoff aircraft has a *planned operating time*, which is also known as the ready time r_j. Usually, a penalty is incurred if the aircraft is scheduled to operate before or after its ready time. The $[e_j, l_j]$ time window has been used previously by Beasley et al. (2000); Salehipour et al. (2013); Vadlamani and Hosseini (2014); Sabar and Kendall (2015). However, this time window is more theoretical since scheduling an aircraft before its ready time is usually considered to be not practical (Rodríguez-Díaz et al., 2017a). This is because operating an aircraft before its ready time might have security risks and for a landing aircraft, it might need more fuel to accelerate beyond its optimal cruise speed (Benlic et al., 2016). In other words, an aircraft is normally not allowed to be operated ahead of its ready time. Consequently, the practical time window $[r_j, l_j]$ is used in aircraft sequencing. In fact, this practical time window has recently attracted much more attention (Samà et al., 2017; Benlic et al., 2016; Lieder and Stolletz, 2016; Ghoniem et al., 2015). As mentioned in Section 2.1, in this paper, we use this practical time window.

2.3. Separation Time Constraints

Each aircraft performing landing or takeoff creates wake turbulence that subsequent aircraft on the same runway needs to avoid. For this, a certain minimum period of separation time is required between any two aircraft operating on the same runway. These separation times depend on the operation types and the aircraft classes (e.g. heavy, large, and small). However, to ensure certain standard safety levels, the exact separation times that are to be enforced 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. Table 1 shows the separation times introduced by the FAA (FAA, 2015) and we use these in this work.

Assume two aircraft $j \neq j'$ operate on the same runway and aircraft j operates before aircraft j' i.e. $\rho_j = \rho_{j'} \wedge t_j < t_{j'}$. Therefore, we need to ensure $t_{j'} - t_j \geq s(w_j, w_{j'})$, where $s(w_j, w_{j'})$ is the minimum separation time for aircraft j' with weight $w_{j'}$ from aircraft j with weight w_j. Given the weights w_j and $w_{j'}$ of the aircraft j and j' respectively, the value of $s(w_j, w_{j'})$ could be found at row w_j and column $w_{j'}$ of Table 1.

Although the separation time constraints must hold between every two aircraft operating on the same runway, Sherali et al. (2010) showed that using the FAA standard depicted in Table 1, these constraints are satisfied automatically when there are at least three aircraft in between two given aircraft, all operating on the same runway. We demonstrate this in Figure 1. In our implementation, we therefore, use this finding.

2.4. Minimising Total Tardiness

As described before, each aircraft has an optimal scheduled operation time, known as its ready time. This implies performing the operation at that time has no delay and no extra fuel burn. However, the time separation constraints could mean that some flights cannot operate at their

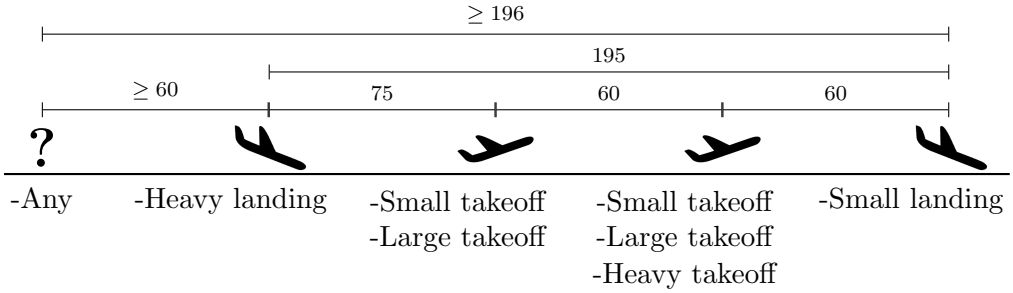

Figure 1: When separation times in FAA standards could be automatically satisfied

ready times. Therefore, the scheduled times for some flights are different from their optimal times. Moreover, if an aircraft j operates after its ready time r_j, its tardiness is $(t_j - r_j)$ which becomes $w_j(t_j - r_j)$ if the aircraft class operation weight w_j is considered. The weight of an aircraft depends on the operation type (e.g. landing and takeoff) and the class of the aircraft (e.g. heavy, large, and small). A landing aircraft has greater priority than a departing aircraft due to higher average fuel burning and safety measures. Departing aircraft can wait at the ground with lower risk. On the other hand, heavier aircraft get more weights than the lighter ones, again based on higher average fuel burning and safety measures. In this paper, we use the weights shown in Figure 1 as these appear in the benchmark problems (Farhadi, 2014) that we use in our experiments. Note that operation types are indirectly encoded in the weights (e.g. smaller weights for takeoff and larger weights for landing) and hence are not explicitly specified in the problem description.

Overall, the objective function of the ASP is to minimise the cost resulting from the deviation of the start times from the respective ready times. In this paper, we use the total weighted tardiness of a schedule TWT $= \sum_{j=1}^{N} w_j(t_j - r_j)$. This objective allows not only reducing delays, but also maximising runway usage, thus reducing congestion at airports (Rodríguez-Díaz et al., 2017a). Also, this objective function can be referred as the total excess fuel cost, as if all start times t_j equal their associated ready times r_j in a given schedule, then no excess fuel cost is incurred (Farhadi, 2014).

2.5. Runway Capacity Constraints

Idris et al. (1998) mentioned that runways are the main constrained resource and thus the source of delays in all airport systems. For airports with multiple runways, the distribution of flights over runways could be uniform. Given uniform distributions of types of operation, types of aircraft, and times of operation, an equal distribution of flights over runways ideally will achieve the optimum solutions in terms of performance. An equal distribution will also improve the longevity and the maintenance of the runways. However, there are plenty of other factors that need to be considered as well.

One such important factor is noise (Heblij and Wijnen, 2008). According to Single European Sky ATM Research (SESAR), the two main environmental issues associated with aviation are emissions and noise (SESAR, 2017). Each runway has specific paths for landing and takeoff. So using mostly one runway in an airport will cause human health problems such as hearing loss, communication interference, sleep interference, anxiety and stress, for those who live in the vicinity of that runway (Rodríguez-Díaz et al., 2017b; Janssen et al., 2014; Ozkurt et al., 2014). Noise pollution these days become more crucial with the rise of populations in cities and their territorial expansion, particularly when residential areas become closer to airports. An equal distribution of

flights over runways thus also addresses this noise issue. Now the question is how to achieve the equal distribution.

The Heathrow Airport, London, has two runways, that are used in a segregated fashion, i.e. one runway for landings and the other for takeoffs. The reason is that the number of arrival and departure flights are almost the same. However, they are using a Runway Alternation rule, which can be described as Half Day Noise Relief (Heathrow, 2017). Although takeoffs need much higher engine thrust than landings, landing noise is frequently more annoying to the ear because of dominant fan noise (Frair, 1984). So, to share the disturbance and to give everyone periods of relief from aircraft noise, based on Runway Alternation mechanism, aircraft lands on one runway between 07.00-15.00, and on the other between 15.00-23.00. The morning/afternoon rota changes weekly, on Mondays (Heathrow, 2017). So, as can be observed, the distribution and allocation of aircraft over runways are very important.

While obtaining an uniform distribution of flights over the runways is a very desired objective, for a given ASP instance, the optimal solution in terms of total tardiness may not have an equal number of flights allocated to each runway. To achieve a balance to a certain extent, in this paper, we consider an upper bound U and a lower bound L on the numbers of flights that could be allocated to each runway. These bounds are computed from $\frac{N}{M} \pm \nu$, where ν is a given number. These bounds essentially also eliminates many solutions that are very suboptimal. It is worth noting that these runway capacity constraints are at the end soft constraints, whose violation could be tolerated. Note that for a given ν, there could be a large number of runway capacity combinations. For example with $N = 15, M = 3, \nu = 1$, possible combinations are $\langle 5, 5, 5\rangle$ where all capacities are equal and $\langle 4, 5, 6\rangle$ and its permutations where variations are allowed. Eliminating symmetries to improve search, we can however only consider $\langle 5, 5, 5\rangle$ and $\langle 4, 5, 6\rangle$. In this paper, we propose to perform separate search with each combination, perhaps in parallel, and return the best solution found. We therefore do not strictly include this as a constraint in our model. For convenience, henceforth, we describe our algorithms and provide theoretical analysis using $\lceil \frac{N}{M} \rceil$ as the maximum number of flights per runway.

3. Existing Related Methods

Extensive overviews of aircraft scheduling are available in several articles (Adler et al., 2013; Bennell et al., 2013, 2011). Samà et al. (2017) divided the air traffic flow management (ATFM) literature into three classifications. The first classification is based on the traffic control between airports (Castelli et al., 2011; Balakrishnan and Chandran, 2014) and the traffic control in the terminal manoeuvring area (TMA) of an individual airport (Furini et al., 2015; Lieder and Stolletz, 2016; Samà et al., 2017). In this paper, we focus on the traffic control in the TMA of an individual airport.

The second classification is based on the type of information: static (Pinol and Beasley, 2006; Ernst et al., 1999; Girish, 2016) and dynamic (Ciesielski and Scerri, 1997; Bennell et al., 2017; Murça and Müller, 2015; Moser and Hendtlass, 2007). In the static case, it is assumed that all data such as ready times, and time windows are known upfront and can be taken into account in the process. However, in the dynamic case, all these data become known only when an aircraft is ready to land or depart. In this research, the static case of the ASP is considered. Note that although modelling approaches for dynamic case tend to be quite different from the ones for the static case, researchers working on these problems still are attracted to the static case. This is because a static approach can be inserted in a dynamic system that iteratively solves an ASP

problem with static information available in each of a series of sliding time windows (Samà et al., 2017).

The third classification is based on the algorithmic approaches and is the main focus of this paper. Since landing is more crucial than takeoff, some of the algorithms only consider landing operations and thus solve the aircraft landing problem (ALP). Comparatively, only a few algorithms consider aircraft takeoff problem (ATP) although ATP is regarded as more difficult to solve compare to the ALP (De Maere and Atkin, 2015). Other algorithms however solve the aircraft sequencing problem (ASP) considering both landing and takeoff operations. Nevertheless, in the literature of these problems, several MIP formulations (Abela et al., 1993; Bianco et al., 1987, 1997; Beasley et al., 2000; Al-Salem et al., 2012; Ghoniem et al., 2014) are proposed that are solved with exact solvers; branch-and-bound (B&B) algorithms (Abela et al., 1993; Ernst et al., 1999; Artiouchine et al., 2008; Wen et al., 2005; D'Ariano et al., 2012; Ghoniem et al., 2015), and dynamic programming (DP) approaches (Trivizas, 1998; Ravidas et al., 2013; Montoya et al., 2014; Balakrishnan and Chandran, 2010; Briskorn and Stolletz, 2014). However, as already mentioned, these problems are NP-hard. Thus, it is difficult to obtain optimal solutions for a problem in a reasonable time. Consequently, it becomes necessary to develop qualified approximate solutions. Below we provide a wide but non-exhaustive review of the recent literature focussing on static cases of ALP, ATP, and ASP separately.

3.1. Aircraft Landing Problem (ALP)

Ernst et al. (1999) addressed the single-runway ALP and pointed out that it could be extended to multiple-runway case. They proposed a genetic algorithm for these two cases considering the penalty costs for landing before and after ready times as the objective function. Fahle et al. (2003) proposed two local search-based algorithms: hill climbing (HC), and simulated annealing (SA) for single-runway ALP. They compared these two algorithms with exact algorithms such as constraint programming (CP) approaches on benchmarks up to 123 aircraft. The results showed the efficiency of SA and HC algorithms compared to the exact methods. In addition, the SA method obtained better solutions although HC was faster.

Pinol and Beasley (2006) addressed the multiple-runway ALP and develop two population-based metaheuristics: scatter search (SS) and a bionomic algorithm (BA). The objective was to achieve effective runway utilisation, where two different objective functions (a non-linear and a linear) were used during the experiments. Hu and Di Paolo (2008) designed binary-representation-based genetic algorithms for ALP rather than a common permutation representation. They showed that this new representation make it possible to easily adopt a uniform crossover operator. The experimental results indicated the efficiency of the new GA against common permutation-based GAs. Wang (2009) proposed a hybrid algorithm that integrated Bee Evolutionary Genetic algorithm with modified clustering method (named BEGA-CM) for solving single-runway ALP. They compared their algorithm with GA and showed that their proposed algorithm by far faster than GA.

Bencheikh et al. (2009) studied multiple-runway ALP and formulated it as a Job Shop Scheduling Problem (JSSP) based on a graphical representation. Then proposed a hybrid algorithm called ACOGA, combination of ant colony optimization (ACO) with GA, and showed the efficiency of proposed algorithm against GA. Salehipour et al. (2013) studied multiple-runway ALP and presented two algorithms hybridising SA with Variable Neighbourhood Search (VNS) and Variable Neighbourhood Descent (VND), namely SA-VNS and SA-VND. They compared their algorithms with SS and BA (Pinol and Beasley, 2006) and the results indicated the effectiveness of SA-VNS and SA-VND against other algorithms compared.

Recently Sabar and Kendall (2015) proposed an Iterated Local Search (ILS) for single and multiple-runway ALP made up of an VND algorithm as local search, multiple operators for perturbation with time-varying perturbation strength. They compared their algorithm with SA-VNS (Salehipour et al., 2013) and SS (Pinol and Beasley, 2006) and their results showed the effectiveness of ILS. Girish (2016) investigated single and multiple runway cases of the ALP and proposed a hybrid particle swarm optimization algorithm (PSO) in a rolling horizon approach. The objective function was the same as is used by Pinol and Beasley (2006). In order to test their algorithm, they used a set of well-known benchmark instances containing up to 500 aircraft and 5 runways, and showed that their algorithm obtained better solution than SA-VNS and SA-VND (Salehipour et al., 2013) and SS and BA (Pinol and Beasley, 2006).

3.2. Aircraft Take-Off Problem (ATP)

This problem has attracted less attention compared to ALP. ATP not only consists of more operational constraints but also is heavily related to taxi-out scheduling problem, so that they need to be combined each other. Atkin et al. (2007) investigated the ATP at London Heathrow Airport which has a single runway to be used for departures with the objective of maximising the runway throughput. They developed different metaheuristics such as Steeper Descent(SD), Tabu Search (TS), and SA to solve the problem. The results showed that the Tabu Search is better than the others with a small margin. Stiverson (2010) proposed a greedy algorithm and a 2-interchange heuristic algorithm for ATP. After testing proposed methods on randomly generated datasets, it was reported that both methods are better than an FCFS method, and 2-interchange heuristic outperforms the greedy algorithm.

3.3. Aircraft Scheduling Problem (ASP)

This problem is also known as integrated landing and takeoff problem. Hancerliogullari et al. (2013) proposed three greedy algorithms and two metaheuristics: SA and Metaheuristic for Randomized Priority Search (Meta-RaPS) to solve multiple-runway ASP. They tested the algorithms on 55 instances based on Doha International Airport (DIA) generated by Farhadi (2014). D'Ariano et al. (2015) showed that ASP can be viewed as a job shop scheduling problem with additional practical constraints such as holding circle constraints, and blocking constraints for runways. They proposed several heuristics and tested them on instances from the Roma Fiumicino airport, Italy. The results showed the efficiency of the heuristics against the FCFS method. Soykan and Rabadi (2016) dealt with multiple-runway ASP and proposed a TS algorithm that includes a hybrid neighbourhood structure. The results showed the ability of this algorithm to find good solutions in reasonable computing times although when compared with the SA (Hancerliogullari et al., 2013), it had worse performance than SA.

As can be seen, although the number of articles on the static case of ASP is still small, most of them are focused on exact methods (Al-Salem et al., 2012; Ghoniem et al., 2015; Ghoniem and Farhadi, 2015; Lieder and Stolletz, 2016; Samà et al., 2017). However, as mentioned before, ASP being NP-Hard, exact methods are not the reasonable choices particularly when practical-sized problem instances are considered. So, we propose constraint optimisation based local search algorithms that are both effective and simple to implement. We then compare our algorithms with two best local search-based metaheuristic that are proposed for the ASP and the related problems. The first one is an SA algorithm by Hancerliogullari et al. (2013) for ASP. The second one is adapted to ASP from the ILS-based best known ALP algorithm by Sabar and Kendall (2015). It is noteworthy again that neither the SA nor the ILS algorithm uses problem specific structural

knowledge from the constraint or the objective functions while our proposed search methods do. While local search algorithms share many aspects with each other, yet certain key aspects make them unique on their own right. Our proposed algorithms with the exploitation of problem specific structural knowledge is clearly and conceptually separable from the existing local search methods. Below we briefly describe the SA and the ILS algorithms.

3.3.1. The Simulated Annealing Approach

The SA algorithm proposed by Hancerliogullari et al. (2013) starts from an initial solution. Next, in a loop, it performs several steps that are typical in a simulated annealing algorithm. It first explores the neighbourhood of the current solution using swap operators. Then, it selects the best neighbouring solution as the new solution. If the new solution is better than the current solution, the new solution is accepted. If the new solution is worse than the current solution, it could still be accepted with some probability. This acceptance probability diminishes with the increase of loop iterations. For initialisation, the SA algorithm proposed three heuristics: Earliest Ready Time (ERT), Fast Priority Index (FPI), and Adapted Apparent Tardiness Cost with Separation and Ready Times (AATCSR). The experimental results showed that the AATCSR performed better than the other two. In AATCSR, starting from an empty solution, every time an unscheduled aircraft with the largest index obtained from a complex formula is appended.

3.3.2. The Iterated Local Search Approach

The ILS algorithm proposed by Sabar and Kendall (2015) starts with an initial feasible solution followed by local search. Then, it iteratively invokes the perturbation phase, the local search phase, and the acceptance phase. To generate an initial solution, it uses a randomised greedy heuristic. For perturbation, it randomly performs either swap or insertion for a number of times, but the number of times the operators are used decreases with the number of ILS iteration completed. For local search, it uses a variable neighbourhood descent (VND) algorithm (Hansen and Mladenović, 2001) with a number of swap and insertion operators. For acceptance criteria, if the solution returned by the local search is better than the current solution, it is always accepted; otherwise, it is accepted with a small probability.

4. Our Solution Representation

Given the problem definition in Section 2.1, a typical ASP solution representation could consider directly assigning a specific time to each aircraft. As in many other optimisation search problems, the ASP decision variables could thus be the time variables taking values from the time horizon. However, in ASP, the weighted tardiness of an aircraft specifically depends on how delayed its operation is from its ready time. Considering the minimisation of the weighted tardiness, each aircraft's starting time therefore can be determined from the preceding aircraft's starting times and the separation times between the aircraft and those aircraft. These specific ASP facts that are directly encoded in the constraints and the objective functions have an influence on the way ASP solutions could be represented. For instance, one can represent ASP solutions by using sequences of aircraft assigned to the runways. The search in this case will be on the permutations of the given aircraft with a view to minimising the total weighted tardiness.

Assume $\pi = \langle \vec{1}, \vec{2}, \ldots, \vec{M} \rangle$ be an ASP solution where $\vec{i}$ denotes the sequence of aircraft that will operate on runway i. Further, assume $|\vec{i}|$ denotes the number of such aircraft, and $i[k]$ denotes such an aircraft at position k. The lower the position k, the earlier the start time $t_{i[k]}$. Below is an

example solution for an ASP with 7 aircraft and 3 runways. Aircraft 3, 7 are scheduled to runway 1 in the order. Similarly, aircraft 4, 1, 5 are scheduled to runway 2 and aircraft 6, 2 to runway 3.

$$\begin{cases} 3, 7 \\ 4, 1, 5 \\ 6, 2 \end{cases}$$

For convenience of theoretical analysis and algorithm description, we assume each runway has a capacity of $\lceil N/M \rceil$ aircraft. However, as mentioned before, in our implementation and experiments, we use upper and lower bounds on the runway capacities to produce a number of combinations and run our algorithm for each combination.

4.1. Calculating Total Tardiness

Assume a runway i with the sequence $\vec{i}$ of aircraft scheduled to operate on. As is mentioned before, since the tardiness $t_{i[k]} - r_{i[k]}$ of a aircraft $i[k]$ in the sequence $\vec{i}$ is calculated using its ready time $r_{i[k]}$ as the reference, the earlier we can schedule the aircraft $i[k]$'s start time, the lesser is the cost. We therefore can directly calculate the start time $t_{i[k]}$ of the aircraft $i[k]$ from the start times of the previous aircraft in the sequence $\vec{i}$ of the same runway i. As is already shown in Figure 1 and by Sherali et al. (2010), we just have to use the start times of at most the previous 3 aircraft.

$$t_{i[1]} = r_{i[1]} \tag{1}$$

$$t_{i[2]} = \max\{r_{i[2]}, t_{i[1]} + s(w_{i[1]}, w_{i[2]})\} \tag{2}$$

$$t_{i[3]} = \max\{r_{i[3]}, t_{i[2]} + s(w_{i[2]}, w_{i[3]}), t_{i[1]} + s(w_{i[1]}, w_{i[3]})\} \tag{3}$$

$$\begin{aligned} t_{i[k]} = \max\{&r_{i[k]}, t_{i[k-3]} + s(w_{i[k-3]}, w_{i[k]}), t_{i[k-2]} + s(w_{i[k-2]}, w_{i[k]}), \\ &t_{i[k-1]} + s(w_{i[k-1]}, w_{i[k]})\} \qquad \forall k \in [4, |\vec{i}|] \end{aligned} \tag{4}$$

Using the above formulas, once we obtain the start times of each aircraft in each runway, we can calculate the total weighted tardiness of a given schedule from the formula shown below.

$$\text{TWT} = \sum_{i=1}^{M} \sum_{k=1}^{|\vec{i}|} w_{i[k]}(t_{i[k]} - r_{i[k]}) \tag{5}$$

For convenience, we also define $\text{TWT}(i) = \sum_{k=1}^{|\vec{i}|} w_{i[k]}(t_{i[k]} - r_{i[k]})$ to be the total weighted tardiness for runway i and $\text{WT}(j) = w_j(t_j - r_j)$ to be the weighted tardiness for aircraft j.

4.2. An Example

Consider an example with 8 aircraft and 2 runways. The required data are given in Table 2. One possible solution is shown in Figure 2 where aircraft 2, 4, 1, 7 in the order are assigned to runway 1. Also, aircraft 5, 8, 3, 6 in the order are assigned to runway 2. Using equations (1)–(4), the start times of the aircraft are obtained as follows:

Table 2: An example problem instance with 8 aircraft and 2 runways

j	1	2	3	4	5	6	7	8
r_j	46	93	117	135	229	250	256	308
d_j	646	693	717	735	829	850	856	908
w_j	1	2	4	6	4	1	1	4

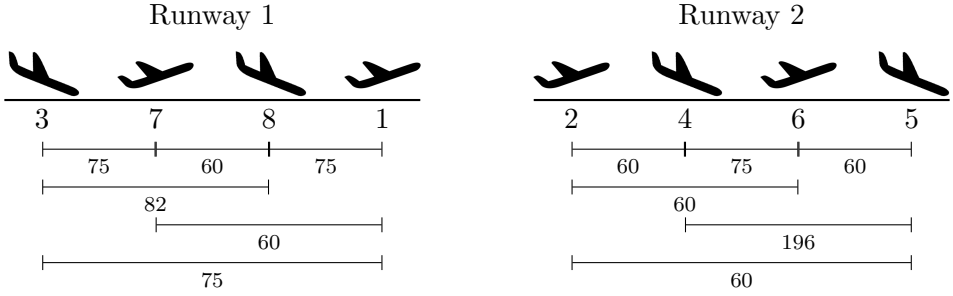

Figure 2: An example schedule for 8 aircraft and 2 runways

- Runway 1:

$$t_{1[1]} = r_{1[1]} = r_3 = 117$$

$$\begin{aligned} t_{1[2]} &= \max\{r_{1[2]}, t_{1[1]} + s(w_{1[1]}, w_{1[2]})\} \\ &= \max\{r_7, t_3 + s(w_3, w_7)\} \\ &= \max\{256, 117 + 75\} = 256 \end{aligned}$$

$$\begin{aligned} t_{1[3]} &= \max\{r_{1[3]}, t_{1[2]} + s(w_{1[2]}, w_{1[3]}), t_{1[1]} + s(w_{1[1]}, w_{1[3]})\} \\ &= \max\{r_8, t_7 + s(w_7, w_8), t_3 + s(w_3, w_8)\} \\ &= \max\{308, 256 + 60, 117 + 82\} = 316 \end{aligned}$$

$$\begin{aligned} t_{1[4]} &= \max\{r_{1[4]}, t_{1[3]} + s(w_{1[3]}, w_{1[4]}), t_{1[2]} + s(w_{1[2]}, w_{1[4]}), t_{1[1]} + s(w_{1[1]}, w_{1[4]})\} \\ &= \max\{r_1, t_8 + s(w_8, w_1), t_7 + s(w_7, w_1), t_3 + s(w_3, w_1)\} \\ &= \max\{46, 316 + 75, 256 + 60, 117 + 75\} = 391 \end{aligned}$$

- Runway 2:

$$t_{2[1]} = r_{2[1]} = r_2 = 93$$

$$\begin{aligned} t_{2[2]} &= \max\{r_{2[2]}, t_{2[1]} + s(w_{2[1]}, w_{2[2]})\} \\ &= \max\{r_4, t_2 + s(w_2, w_4)\} \\ &= \max\{135, 93 + 60\} = 153 \end{aligned}$$

$$\begin{aligned} t_{2[3]} &= \max\{r_{2[3]}, t_{2[2]} + s(w_{2[2]}, w_{2[3]}), t_{2[1]} + s(w_{2[1]}, w_{2[3]})\} \\ &= \max\{r_6, t_4 + s(w_4, w_6), t_2 + s(w_2, w_6)\} \\ &= \max\{250, 153 + 75, 93 + 60\} = 250 \end{aligned}$$

$$\begin{aligned} t_{2[4]} &= \max\{r_{2[4]}, t_{2[3]} + s(w_{2[3]}, w_{2[4]}), t_{2[2]} + s(w_{2[2]}, w_{2[4]}), t_{2[1]} + s(w_{2[1]}, w_{2[4]})\} \\ &= \max\{r_5, t_6 + s(w_6, w_5), t_4 + s(w_4, w_5), t_2 + s(w_2, w_5)\} \\ &= \max\{229, 250 + 60, 153 + 196, 93 + 60\} = 349 \end{aligned}$$

From the start times of the aircraft obtained above, we then calculate the total weighted tardiness in the following way:

$\text{TWT} = \sum_{i=1}^{M} \sum_{k=1}^{|\vec{i}|} w_{i[k]}(t_{i[k]} - r_{i[k]}) = (4\times(117-117)) + (1\times(256-256)) \quad +(4\times(316-308)) + (1\times(391-46)) + (2\times(93-93)) + (6\times(153-135)) \quad +(1\times(250-250)) + (4\times(349-229)) = 965$

Lemma 1. *Computing TWT from scratch using Equations (1)–(5) has a time complexity of $O(N)$ and a space complexity of $O(N)$ as well.*

Proof: For convenience of the worst case analysis, we assume Equation 4 to be the generalised case for all aircraft. Clearly, computation of $t_{i[k]}$ from Equation 4 needs $O(1)$ time and $O(1)$ space. Thus, for N aircraft, computing the ready times needs $O(N)$ time and $O(N)$ space. In Equation 5, we compute weighted tardiness for each aircraft. So the total time is $O(N)$ and space is $O(1)$. Hence, computing TWT has a time complexity $O(N)$ and a space complexity $O(N)$.

5. Proposed Constructive Search Method

Our objective is to minimise total weighted tardiness TWT of a schedule. For each given aircraft, its weighted tardiness depends on how early it can be scheduled and is zero if it is scheduled at its ready time. *In our constraint-guided constructive search, we exploit the weighted tardiness criterion explicitly by selecting the next aircraft based on its ready time and then scheduling it at its best position in the partial solution already constructed.* For this the next aircraft is scheduled at each possible position in a given partial solution with k aircraft and each potential partial solutions with $k+1$ aircraft are evaluated. Our constructive method is thus search based and is guided by the objective function while existing constructive methods, e.g. in (Hancerliogullari et al., 2013), are mere appending of the next aircraft using a predefined criterion, which is not the objective function directly. Although our final constructive search method uses swap operators, we also implement an insertion operator based one. We call our approaches swap and insertion based heuristics (SBH and IBH) respectively and describe them below.

Swap Based Heuristic (SBH)

In our SBH method described in Algorithm 1, we first arrange all the aircraft in an increasing order of their ready times and get a list θ. Next, we take the first M aircraft from θ and assign each of them to a different runway to obtain an initial solution π. Then, we consider an iteration for each aircraft in θ. In each iteration, the next aircraft in θ is appended to the list of aircraft in all the runways. This produces M new solutions. Assume π_i is one such new solution and the aircraft was added to the ith runway. Now for each π_i, the last aircraft in the ith runway is swapped with all other aircraft in the same runway to obtain further newer solutions. Among all the new and newer solutions, the best one is selected for the next iteration.

For an example, assume an ASP in Table 3 with 5 aircraft and 2 runways. At first, based on the ready times, the initial sequence becomes $\theta = \langle 1, 2, 3, 4, 5\rangle$. Aircraft 1 and 2 are first assigned to runway 1 and 2 respectively. Note that the runways are identical here.

$$\begin{cases} 1 \\ 2 \end{cases}$$

Then, the aircraft at the third position of θ is 3. Appending 3 separately at the end of both runways and then performing swaps within the same runway, we obtain the following solutions:

Algorithm 1: SBH

1. Arrange all aircraft in the given problem in ascending order of their ready times and put them in a list θ. We use θ_k $(1 \leq k \leq N)$ to denote the aircraft at position k in θ.
2. Without loss of generality, let $\pi = \langle \vec{1}, \ldots \vec{M} \rangle$ be the initial solution where each $\vec{i}$ has one aircraft θ_i.
3. For each $k \in [M+1, N]$, take aircraft θ_k from θ
   (a) For each runway $i : |\vec{i}| < \lceil N/M \rceil$, append θ_k to the end of $\vec{i}$ of π to obtain a solution π_i. We have at most M new solutions.
   (b) For each solution π_i of the M solutions from the previous step, for each $1 \leq k' < |\vec{i}|$, swap $i[k']$ with the appended aircraft $i[|\vec{i}|]$ to obtain $|\vec{i}| - 1$. Thus, from this step, in total, we get $\sum_i (|\vec{i}| - 1) = k - 1$ new solutions
   (c) Let π be the best solution (in terms of weighted tardiness) from the $M + k - 1$ new solutions found in the previous two steps. Solution π is used in the next iteration of k
4. Return solution π.

Table 3: An example problem with 5 aircraft and 2 ruways

j	1	2	3	4	5
r_j	32	43	97	112	154
d_j	632	643	697	712	754
w_j	5	6	4	2	3

$$\left\{ \begin{array}{l} 1,3 \\ 2 \end{array} \right. \qquad \left\{ \begin{array}{l} 3,1 \\ 2 \end{array} \right. \qquad \left\{ \begin{array}{l} 1 \\ 2,3 \end{array} \right. \qquad \left\{ \begin{array}{l} 1 \\ 3,2 \end{array} \right.$$

The weighted tardiness for the above partial solutions are 0, 1110, 168 and 900 from left to right. Therefore, the first partial solution from above is selected. Next the subsequent aircraft 4 from the sequence θ needs to be considered for insertion into the selected partial solution.

$$\left\{ \begin{array}{l} 1,3,4 \\ 2 \end{array} \right. \qquad \left\{ \begin{array}{l} 1,4,3 \\ 2 \end{array} \right. \qquad \left\{ \begin{array}{l} 4,3,1 \\ 2 \end{array} \right. \qquad \left\{ \begin{array}{l} 1,3 \\ 2,4 \end{array} \right. \qquad \left\{ \begin{array}{l} 1,3 \\ 4,2 \end{array} \right.$$

The weighted tardiness for the above 5 partial solutions are 120, 150, 1345, 12 and 774 from left to right. Therefore, the fourth partial sequence is selected for the next and final step. The fifth aircraft in θ is aircraft 5 which can be inserted into the selected partial solution in the following ways:

$$\left\{ \begin{array}{l} 1,3,5 \\ 2,4 \end{array} \right. \qquad \left\{ \begin{array}{l} 1,5,3 \\ 2,4 \end{array} \right. \qquad \left\{ \begin{array}{l} 5,3,1 \\ 2,4 \end{array} \right. \qquad \left\{ \begin{array}{l} 1,3 \\ 2,4,5 \end{array} \right. \qquad \left\{ \begin{array}{l} 1,3 \\ 2,5,4 \end{array} \right. \qquad \left\{ \begin{array}{l} 1,3 \\ 5,4,2 \end{array} \right.$$

The weighted tardiness for the above partial solutions are from left to right 66, 480, 1735, 84, 324 and 2310. As a result, the first solution becomes the final solution; which will be used as the initial solution in the EPS.

Lemma 2. *The SBH algorithm shown in Algorithm 1 has a time complexity $O(N^3)$ and a space complexity $O(N)$.*

Proof: The first step in Algorithm 1 takes $O(N \lg N)$ time and $O(N)$ space to sort the aircraft. The second step needs $O(M)$ time and $O(M)$ space. For the worst case analysis, let us ignore the condition in $i : |\vec{i}| < \lceil N/M \rceil$ and try all runways. The worst case appears when each subsequent aircraft is placed in the runway that is the next one in a round robin fashion. Therefore, for a given $k \in [M+1, N]$, we generate $M + k - 1$ new partial solutions each having k aircraft. If we compute the TWT from scratch for every partial solution, it will take $O(k)$ time and $O(k)$ space. Since in each swap, aircraft only in one runway get affected, we can however compute TWT incrementally in $O(\lceil N/M \rceil)$ time and $O(\lceil N/M \rceil)$ space. So the overall time complexity becomes $O(N \lg N + M + \sum_{k=M+1}^{N}(M + k - 1)k)$ when TWT is computed form scratch and $O(N \lg N + M + \sum_{k=M+1}^{N}(M + k - 1)\lceil N/M \rceil)$ when TWT is computed incrementally. Simplifying, the time complexity comes to $O(N^3)$ assuming $N \gg M$. The overall space complexity however remains $O(N)$.

Insertion Based Heuristic (IBH)

In our IBH method described in Algorithm 2, we first arrange all the aircraft in an increasing order of their ready times and get a list θ. Next, we take the first M aircraft from θ and assign each of them to a different runway to obtain an initial solution π. Then, we consider an iteration for each aircraft in θ. In each iteration, the next aircraft in θ is inserted in all possible positions in all the runways π and from the new solutions, the best one (in terms of weighted tardiness) is selected for the next iteration.

Algorithm 2: IBH

1. Arrange all aircraft in the given problem in ascending order of their ready times and put them in a list θ. We use θ_k $(1 \leq k \leq N)$ to denote the aircraft at position k in θ.
2. Without loss of generality, let $\pi = \langle \vec{1}, \ldots \vec{M} \rangle$ be the initial solution where each $\vec{i}$ has one aircraft θ_i.
3. For each $k \in [M+1, N]$, take aircraft θ_k from θ
   (a) For each runway $i : |\vec{i}| < \lceil N/M \rceil$, insert θ_k in all possible positions k' of $\vec{i}$ of π to obtain solutions $\pi_{i,k'}$, where $1 \leq k' \leq \hat{n} + 1$ and $\hat{n} = |\vec{i}|$ before inserting θ_k. Thus we have $\sum_i (\hat{n} + 1)$ new solutions
   (b) Let π be the best solution (in terms of weighted tardiness of each $\pi_{i,k'}$) from the new solutions found above (if any). Solution π is used in the next iteration of k
4. Return solution π.

Lemma 3. *The IBH algorithm shown in Algorithm 2 has a time complexity $O(N^3)$ and a space complexity $O(N)$.*

Proof: The proof is very similar to that of Lemma 2.

6. Proposed Local Search Method

Local search involves generation of neighbouring solutions around the current solution and evaluation of the neighbouring solutions. From the current solution, local search then moves to one of the neighbouring solution, preferably to the best one as per given criteria. A typical local search algorithm would consider the neighbour generation and evaluation procedures to be independent of each other. As such, it would generate the neighbouring solutions randomly or exhaustively, so not much purposefully, although using predefined neighbourhood generation operators.

A more effective approach would be to be selective in neighbourhood generation either to save the effort required to explore the unpromising areas of the search space or to quickly take the search to the promising areas. In the constraint optimisation paradigm in artificial intelligence, neighbourhood generation and evaluation are considered to be part of an integrated process. So the evaluation functions i.e. the constraints and the objective functions are used more purposefully in the generation of the neighbourhoods. The current solution is analysed to identify the problematic part in terms of the constraint violation or the objective value and neighbourhood solutions are generated to improve that part of that solution. In our proposed local search method for ASP, we will design neighbourhood operators, even if those are swap and inserts they will be customised, such that unpromising neighbouring solutions are generated as few as possible, and we will apply the neighbourhood operators on runways that have the most weighted tardiness i.e. part of the problematic part of the solution. To test the effectiveness of our constraint guided strategies, we implement them on top of a generic explorative perturbative search (EPS) method shown in Algorithm 3. Note that our contribution is not in the EPS algorithm itself, rather in the difference that we make in terms of its performance when we use our constraint guided strategies within the EPS framework.

Algorithm 3: Explorative Pertubative search (EPS)

```
π ← generateInitialSolution() // IBH or SBH
while termination criteria not satisfied do
    π′ ← performExploration(π)
    π ← acceptAndUpdateBest(π′)
    π ← performPerturbation(π)
endwhile
Return the best solution
```

At first Algorithm 3 obtains a heuristically constructed solution by using our proposed constructive method. Within a loop, it then performs explorative local search to find a new improving solution. With certain acceptance criteria, the new solution is then accepted as the current solution or is discarded. Inside the loop, the current solution is then perturbed, meaning partially destroyed and then the destroyed part is constructed again, to obtain a new solution. The new solution becomes the current solution for the next iteration. The perturbation in effect serves the purpose of a partial restart to take the search out of a local optimum or a plateau. It is worth mentioning that in all phases of our algorithms, feasibility of a solution is checked as soon as the solution is generated and any infeasible solution is immediately discarded.

6.1. Performing Exploration

The kinds of move used in a local exploration play a significant role in its effectiveness. Given the permutation type representation of the ASP solutions, the typical search operators are swap and insert moves e.g. in travelling salesman problem Osaba et al. (2016) and in vehicle routing problem (Belhaiza et al., 2014). In general, the kinds of insert and swap moves used in the literature are inter-insert, intra-insert, inter-swap, and intra-swap. The inter-insert operator removes one aircraft from a runway and inserts it into another runway. The intra-insert operator removes one aircraft from a runway and inserts it into the same runway. The inter-swap performs swaps of aircraft pairs where two aircraft are in two different runways. The intra-swap performs swaps of aircraft pairs where two aircraft are from the same runway.

In this paper, instead of the intra-swap, we use another type of swap called adjacent swap that performs swapping of two adjacent aircraft in the same runway; in most cases the ready times of the two aircraft do not change by a very large margin making it preferable to intra-swap. With intra-swap operators, most of the new solutions become infeasible when the two aircraft to be swapped are far apart. Adjacent swap will generate fewer new solutions than intra-swap but most of the new solutions will be feasible. For the same reason, intra-insert operator is not preferable either. However, adjacent insert is the same as the adjacent swap. *Note that the consideration of the adjacent swap or adjacent insert is based on the analysis of the constraints here and is part of our constraint guided strategies.*

As we will discuss later in Section 6.4, we perform ASP search on a given combination of runway capacities. Hence, we do not use inter-insert operator in this work. Below we describe the exact implementation of all the insert and swap operators.

- **Inter-Swap Operator:** Given a solution π, one runway i with the most weighted tardiness and another random runway $i' \neq i$ are selected and then each aircraft $i[k]$ in π is then swapped with each aircraft $i'[k']$ in π to obtain $|\vec{i}| \times |\vec{i'}|$ new solutions. Only the best new solution is however returned. Algorithm 4 provides the pseudocode of the inter-swap operator and Figure 3 (a) shows an example. Clearly, Algorithm 4 runs in $O(\lceil N/M \rceil^3)$ time and $O(N)$ space when TWT is computed incrementally.

Algorithm 4: Inter-swap operator

1. Let π be the current solution given as a parameter
2. Select a runway $i = \text{argmax}_i \sum_{k=1}^{|\vec{i}|} w_{i[k]}(t_{i[k]} - r_{i[k]})$ and another random runway $i' \neq i$ randomly.
3. For each $k \in [1, |\vec{i}|]$ for each $k' \in [1, |\vec{i'}|]$, every time in π, swap aircraft $i[k]$ and $i'[k']$ to obtain a new solution.
4. From the $|\vec{i}| \times |\vec{i'}|$ new solutions, return the best one (in terms of weighted tardiness)

- **Adjacent-Swap Operator:** Given a solution π, two runways $i' \neq i''$ are first chosen uniformly randomly. Then for each runway $i \in \{i', i''\}$, for each two adjacent aircraft $i[k]$ and $i[(k+1)]$ in $\vec{i}$ of π are swapped with each other to obtain $|\vec{i}| - 1$ new solutions. Only the best new solution is however returned. Algorithm 5 provides the pseudocode of the adjacent-swap operator and Figure 3 (b) shows an example. Note that we use two runways here although the swapping of aircraft are done within a single runway. This is just to match with the inter-swap operator that involves two runways every time. Also, note that we select both

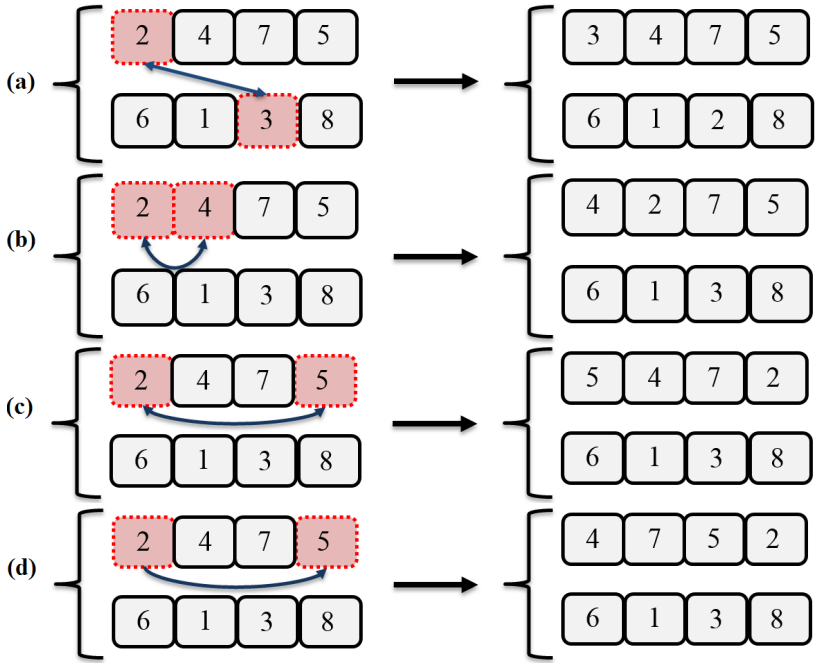

Figure 3: Operators: (a) inter-swap, (b) adjacent-swap, (c) intra-swap, (d) intra-insert

runways randomly because the runway with the highest weighted tardiness is already considered in the perturbation phase. Clearly, Algorithm 5 runs in $O(\lceil N/M \rceil^2)$ time and $O(N)$ space when TWT is computed incrementally.

Algorithm 5: Adjacent-swap operator

1. Let π be the current solution given as a parameter
2. Randomly choose two runways i' and i'' from all runways
3. For each $i \in \{i', i''\}$, for each $k \in [1, |\vec{i}|)$, every time in π, swap aircraft $i[k]$ and $i[k+1]$ to obtain a new solution.
4. From the $|\vec{i'}|+|\vec{i''}|-2$ new solutions, return the best one (in terms of weighted tardiness)

- **Intra-Swap Operator:** Given a solution π, two runways $i' \neq i''$ are first chosen uniformly randomly. Then for each runway $i \in \{i', i''\}$, each pair of aircraft $i[k]$ and $i[k']$ in $\vec{i}$ of π are swapped with each other to obtain $|\vec{i}| \times (|\vec{i}| - 1)$ new solutions. Only the best new solution is however returned. Algorithm 6 provides the pseudocode of the intra-swap operator and Figure 3 (c) shows an example. Note that we use two runways here although the swapping of aircraft are done within a single runway. This is just to match with the inter-swap operator that involves two runways every time. Also, note that we select both runways randomly because the runway with the highest weighted tardiness is already considered in the perturbation phase. Clearly, Algorithm 6 runs in $O(\lceil N/M \rceil^3)$ time and $O(N)$ space when TWT is computed incrementally.

- **Intra-Insert Operator:** Given a solution π, two runways $i' \neq i''$ are first chosen uniformly

Algorithm 6: Intra-swap operator

1. Let π be the current solution given as a parameter
2. Randomly choose two runways i' and i'' from all runways
3. For each $i \in \{i', i''\}$, for each $k, k' \in [1, |\vec{i}|)$, every time in π, swap aircraft $i[k]$ and $i[k']$ to obtain a new solution.
4. From the $|\vec{i'}| \times (|\vec{i''}| - 1)$ new solutions, return the best one (in terms of weighted tardiness)

randomly. Then for each runway $i \in \{i', i''\}$, each $i[k]$ is removed and inserted at each position in $\vec{i}$ of π to obtain $(|\vec{i}| - 1)$ new solutions. Only the best new solution is however returned. Algorithm 7 provides the pseudocode of the intra-insert operator and Figure 3 (d) shows an example. Note that we use two runways here although the removal and insertion of aircraft are done within a single runway. This is just to match with the inter-swap operator that involves two runways every time. Also, note that we select both runways randomly because the runway with the highest weighted tardiness is already considered in the perturbation phase. Clearly, Algorithm 7 runs in $O(\lceil N/M \rceil^3)$ time and $O(N)$ space when TWT is computed incrementally.

Algorithm 7: Intra-insert operator

1. Let π be the current solution given as a parameter
2. Randomly choose two runways i' and i'' from all runways
3. For each $i \in \{i', i''\}$, for each $k, k' \in [1, |\vec{i}|)$, every time in π, remove aircraft $i[k]$ and insert at position k' to obtain a new solution.
4. From the $|\vec{i'}| \times (|\vec{i''}| - 1)$ new solutions, return the best one (in terms of weighted tardiness)

We use at most two operators in the perform exploration procedure shown in Algorithm 8. We will later choose the operators through experimentation. Algorithm 8 is basically a Variable Neighbourhood Descent (VND) algorithm (Hansen and Mladenović, 2001). The main advantage of VND is due to the use of multiple neighbourhood operators in local search. The search might lead to a different local optimum when using each of these operators separately. However, using them within a combination has the advantage of escaping a local optimum associated with one operator through the use of another operator.

6.2. Performing Perturbation

There could be various ways to perturb (destroy and reconstruct some parts of) an ASP solution. Existing methods mostly perform random swapping or random removal and reinsertion for a number of times. *In our constraint guided strategy, we make an attempt to fix the most problematic part in the current solution.* This could mean selecting the aircraft j having the most weighted tardiness $\mathrm{WT}(j)$ and placing it to the best possible position. However, selecting such aircraft does not work well because mostly the aircraft that are towards the end of the schedule of the runways get selected. This is because the tardiness of the previous aircraft has a cascaded effect on the aircraft scheduled later. We therefore select a runway i having the maximum weighted tardiness

Algorithm 8: Perform exploration

1. Let π be the current solution given as a parameter
2. Let $\mathcal{N}_1$ and $\mathcal{N}_2$ are two given neighbourhood operators
3. Let $l = 1$ // to denote operator $\mathcal{N}_l$ will be used
4. While $l \leq 2$ do // since we are using two operators here
   (a) π' = the best solution (in terms of weighted tardiness) among all the solutions obtained by applying $\mathcal{N}_l$ on π
   (b) If π' is better than π then $\pi = \pi', l = 1$ else $l = l + 1$
5. Return solution π

$\text{WT}(i) = \sum_{k=1}^{|\vec{i}|} w_{i[k]}(t_{i[k]} - r_{i[k]})$ of the aircraft assigned to the runway. Then, we take c aircraft from c randomly selected positions in $\vec{i}$ and remove them from $\vec{i}$. Then we append each of the c aircraft to the end of $\vec{i}$ and consider swapping the aircraft with all other aircraft in the same runway. Among the solutions produced, the one with the best $\text{WT}(i)$ is selected to be used for the next d. We repeat the above steps for a given d times. Algorithm 9 shows the perturbation procedure.

Algorithm 9: Swap based perturbation

1. Let π be the current solution given as input
2. For $d' = 1$ to d do // d is given as input
   (a) $i = \text{argmax}_i \sum_{k=1}^{|\vec{i}|} w_{i[k]}(t_{i[k]} - r_{i[k]})$
   (b) For each $c' = 1$ to c do // c is a given input,
   remove job $j_{c'}$ from $\vec{i}$ where $j_{c'} = i[k]$, k is randomly selected from $[1, |\vec{i}|]$, and at least one $j_{c'}$ for this d' should be different from all $j_{c'}$ selected for the immediate previous d'.
   (c) For each $c' = 1$ to c do, // add to $\vec{i}$ in SBH style
   append $j_{c'}$ at the end of $\vec{i}$ to get a new $\vec{i}$ and then for each $k' \in [1, |\vec{i}|]$ but $k' \neq k$, swap aircraft $j_{c'}$ with the aircraft at $i[k']$ to obtain a new $\vec{i}$. Let $\vec{i}$ be the best (in terms of weighted tardiness) among all the new $\vec{i}$.
3. Return solution π

Instead of swapping the selected aircraft with the other aircraft scheduled in the same runway, we also experiment with a removal and reinsertion based approach. In this case, as part of the destruction phase, we remove c aircraft from $\vec{i}$. Then, as part of the construction phase, each of the c aircraft is inserted into $\vec{i}$ in all possible positions—at the beginning, at the end, and in between two successive aircraft, but not at position k again—to obtain new solutions. The best one (in terms of weighted tardiness) among the new solutions is then selected on which similar perturbation steps are repeated. Details of this approach is shown in Algorithm 10.

In our experiments, we also use random runway selection in Algorithms 9 and 10 instead of the greedy selection keeping all other steps in those algorithms the same. This is to show the effectiveness of the greedy selection.

Lemma 4. *Algorithms 9 and 10 both runs in $O(dN^2)$ time and $O(N)$ space.*

Algorithm 10: Insertion based perturbation

1. Let π be the current solution given as input
2. For $k = 1$ to d do // d is given as input
   (a) $i = \text{argmax}_i \sum_{k=1}^{|\vec{i}|} w_{i[k]}(t_{i[k]} - r_{i[k]})$
   (b) For each $c' = 1$ to c do // c is a given input,
   remove job $j_{c'}$ from π where $j_{c'} = i[k]$, k is randomly selected from $[1, |\vec{i}|]$, and at least one $j_{c'}$ for this d' should be different from all $j_{c'}$ selected for the immediate previous d'.
   (c) For each $c' = 1$ to c do, // add to $\vec{i}$ in IBH style
   insert $j_{c'}$ at the beginning, at the end, or at any middle position of $\vec{i}$ to obtain new $\vec{i}$s. Let $\vec{i}$ be the best (in terms of weighted tardiness) among all the new $\vec{i}$.
3. Return solution π

Proof: In each of the d times, selection of the runway takes $O(M)$ time and $O(M)$ space, then $O(\lceil N/M \rceil)$ new solutions are created and computing the TWTs take $O(N)$ time if from scratch or $O(\lceil N/M \rceil)$ time if incrementally. Assuming $N \gg M$, the time complexity is thus $O(dN^2)$. The space complexity is certainly related to the number of aircraft.

6.3. Acceptance criterion

The new solution π' returned by the exploration method is accepted in Algorithm 3 if it is better than the current solution π; otherwise it is accepted with a probability p where p is a parameter of our algorithm.

6.4. Handling Runway Capacity

Assume 15 aircraft, 3 runways and $\nu = 2$. So the maximum and minimum numbers of aircraft to be allocated to each runway are 7 and 3 respectively. Disregarding possible symmetries over permutations, we can distribute 15 aircraft over 3 runways in the following 4 ways $\langle 5, 5, 5\rangle$, $\langle 4, 5, 6\rangle$, $\langle 3, 6, 6\rangle$, $\langle 3, 5, 7\rangle$. These 4 combinations have standard deviations 0, 1, 1.732, and 2 respectively. We can use the standard deviation of a combination to denote the diversity level within that combination.

As could be easily understood, there are quite a large number of possible combinations for moderate sizes of N, M, ν. Finding an optimal solution for an ASP is very difficult even for a given combination, let alone trying all possible combinations. We therefore propose a practical approach. Since in theory, with uniform distributions of types of aircraft, types of operations, and times of operations, optimal solutions lie in the combinations that have standard deviations very close to zero, we should consider only such combinations. Moreover, separate runs could be made, preferably in parallel, for each combination than allowing change of combinations within the same run. This is to produce good solutions very quickly given the short time windows available in ASP instances and also considering the typical time performance of our algorithms. The length of typical time windows could be understood from the following information.

For a landing aircraft, the scheduling process of the aircraft starts once the aircraft enters the TMA, about 30-40 minutes before its planned landing time. The start time (operation time) is finalised in advance of reaching the final approach path, about 20-30 minutes before landing (Soykan, 2016). For a departing aircraft, the scheduling process of each aircraft starts once the

aircraft enters the holding area. The start time (operation time) is finalised in advance of entering the taxiway, since it is not possible to change the sequence during taxiing or at the holding area. Thus, departing operations are scheduled about 20 minutes before planned departure time (Bennell et al., 2013). The planning horizon for ASP is usually 20-30 minutes.

In this work, we propose very efficient local search algorithms that run on a given combination of runway capacities. These algorithms are fast enough to be used within a dynamic system that solves a series of static instances to deal with the sliding time windows in a dynamic system (Samà et al., 2017, 2014). In our experiments, we however evaluate our algorithms over various combinations of runway capacities having various standard deviations.

7. Experimental Results

We have implemented our algorithm in C++ language and on top of the constraint-based local search system, Kangaroo (Newton et al., 2011). The functions and the constraints are defined using invariants in Kangaroo. Invariants are special constructs that are defined by using mathematical operators over the variables. While computation of constraint violations and objective functions, and temporary evaluation of neighbourhood solutions are performed incrementally by Kangaroo, we mainly focus on the search algorithms. It is worth noting that by using Kangaroo, we automatically get speed-up methods that achieve performances similar to that achieved by the acceleration methods proposed by Taillard (1990) for flowshop scheduling. As mentioned in Section 3, we compare our algorithm with the SA algorithm (Hancerliogullari et al., 2013) and the ILS algorithm for ASP (Sabar and Kendall, 2015). These two algorithms have been reconstructed in C++ and on top of Kangaroo. All experiments reported in this work have been run on a MAC computer 2.70 GHz Intel Core i5-4570 processor and 8 GB RAM 1600MHz memory using the OS X operating system.

In this work, we test our algorithms on 12 real-world instances based on Milano Linate Airport (Furini et al., 2012). These instances include 60 aircraft whereas weights are determined based on the number of seats and the fuel consumption of each aircraft. These instances are publicly available at http://www.or.deis.unibo.it/research.html. In this work, we also have used 75 instances that were generated by Farhadi (2014) considering the realistic parameters of the Doha International Airport. Using real-world instances overs the crafted ones are often preferable but depending on exactly what scenarios are used the real-world instances might not have all the characteristics to test the strengths and weaknesses of a particular algorithm. Therefore, crafted instances particularly when generated using realistic parameters are good benchmarks for a thorough empirical evaluation of an algorithm.

This benchmark set used by Farhadi (2014) using Doha International Airport parameters is made up of problem instances with a number of aircraft $n = 15, 20, 25, 50$ and a number of runways $r = 2, 3, 4, 5$. These instances are available from the website http://ahmed.ghoniem.info/download/MASP-SET.txt. In this benchmark, the ready times of the aircraft are randomly generated using a discrete uniform distribution over the interval $(0, \gamma \times \frac{n}{m})$ where γ is a parameter selected in the interval (30, 100). In addition, operations (landing and takeoff) and aircraft classes (small, large and heavy) are selected randomly. The minimum separation times are taken from actual numbers based on FAA standard that can be seen in Table 1.

Before starting the comparison, it is worth noting that in this paper, the main instances for evaluation are those proposed by Farhadi (2014) using Doha International Airport parameters. These benchmarks vary in sizes of the aircraft and runways, which allows us to have a comprehensive

analyse and find the impact of each of the factors of the tested algorithms. These instances have been widely used in the literature e.g. in Ghoniem and Farhadi (2015); Ghoniem et al. (2015); Soykan and Rabadi (2016). Aircraft scheduling becomes important because finding an optimal schedule is really hard when a large number of aircraft are to be operated within a small period of time. In real instances taken from Milan airport, 60 aircraft should be operated in almost 2.5 hours (two and half hours) while in Farhadi's benchmark 50 aircraft should be operated in almost 40 min in a case with two runways. Interestingly, it is clear that when more runways are added in an instance, solving the problem becomes easier. This has been considered in Farhadi's instances. For example, in instances with 50 aircraft and 3 runways, 50 aircraft should be operated in almost 25 minute. All of aforementioned reasons encourage us to use Farhadi's benchmark instead of real-world instances.

The performance of each algorithm is evaluated by means of Error (ERR) defined by the following formula where TWT is the average of the total weighted tardiness obtained by the given algorithm over 5 runs and Best is the best weighted tardiness obtained by any of the tested algorithms.

$$\text{ERR} = \frac{\text{TWT} - \text{Best}}{\text{Best}} \times 100\% \tag{6}$$

As a termination criterion, all local search algorithms are run for $T_{\max} = N \times M \times k$ ms CPU time, where N and M are respectively the numbers of aircraft and runways while $k = 10$.

7.1. Proposed Constructive Search Method

We use two constructive heuristics IBH and SBH. For both, we first sort the aircraft on an increasing order of the ready times. As an alternative, we could arrange the aircraft on a decreasing order of the weights breaking ties randomly. Combining these alternatives, we have four constructive heuristic methods: IBHR, IBHW, SBHR, and SBHW. Among these, SBHW mostly leads to infeasible solutions, so we do not consider it for further evaluation. With the three remaining heuristics, we also compare the AATCSR heuristic proposed by Hancerliogullari et al. (2013), who showed that AATCSR is better than two other heuristics. AATCSR allocates aircraft to runways using a complex index formula.

Table 4 shows performance of the heuristics. Based on the results, we see that the proposed three heuristics are better than AATCSR and the ready time criterion is better than the weight criterion. Moreover, SBHR is better than IBHR implying swapping is better than insertion. However, to check the statistical significance of these conclusions, we also show a 95% confidence interval plot in Figure 4. From this plot, we see that the difference between AATCSR and IBHW is not statistically significant. The same is true for the difference between IBHR and SBHR. However, the difference between the performance of IBHR or SBHR is statistically significant from that of IBHW or AATCSR.

It is worth noting that the average CPU time taken by three proposed heuristics are around 0.03 second, which is larger than the CPU time 0.001 second taken by AATCSR. However, 0.03 second is practically a negligible time, particularly in the context of the aircraft sequencing problem. Overall, AATCSR heuristic allocates aircraft to runways following a complex ranking, while proposed heuristics, although start with an initial ranking, perform all possible insertion or swapping of a given aircraft. The difference in the CPU time is therefore reasonable and pays off.

Table 4: Performance of the constructive heuristics.

Instance	**AATCSR**	**IBHW**	**IBHR**	**SBHR**
$N, M, \#$	ERR (%)	ERR (%)	ERR (%)	ERR (%)
15, 2, 5	40.9	11.2	10.7	**5.8**
15, 3, 5	35.2	**1.2**	15.9	12.8
15, 4, 5	15.8	**0.0**	15.7	10.8
20, 2, 5	58.0	25.8	9.0	**3.1**
20, 3, 5	30.9	17.6	**1.6**	5.0
20, 4, 5	40.9	11.2	10.7	**5.8**
20, 5, 5	21.1	**2.3**	9.0	9.0
25, 2, 5	61.2	39.7	**3.8**	**3.8**
25, 3, 5	28.8	18.3	14.3	**10.0**
25, 4, 5	44.1	19.1	19.9	**10.1**
25, 5, 5	20.5	21.3	03.4	**0.5**
50, 2, 5	189.7	44.9	**0.0**	**0.0**
50, 3, 5	151.5	78.9	**7.2**	**07.2**
50, 4, 5	11.9	183.1	164	**5.1**
50, 5, 5	112.8	213.9	2.2	**0.0**
average	58.9	45.4	8.9	**6.7**

Number of instances for a given N, M pair Bold: minimum error

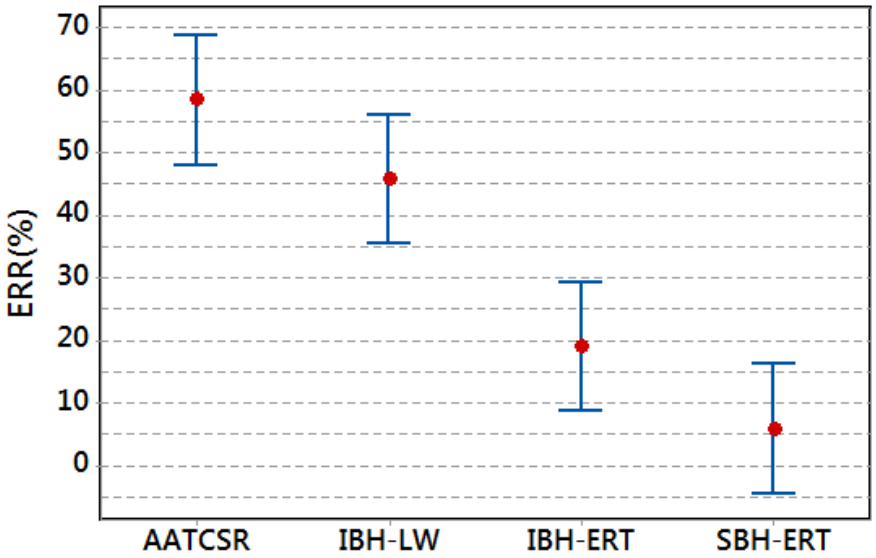

Figure 4: 95% confidence interval plot for four heuristic algorithms

Given the conclusions above and unless stated otherwise, for the rest of the experiments, we will use SBHR as our initialisation method for the explorative perturbative search and will also denote this simply by SBH.

7.2. Local Search Parameter Tuning

There are three parameters in our EPS algorithm: one is d the number of times to perform the perturbation steps, another is c the number of aircraft to be selected in each iteration of the perturbation steps, and the other is p the probability of accepting the solution returned by the

exploration procedure. To determine the suitable values of these parameters, we follow the method of full factorial design. For each parameter, we consider 4 different values, which are listed below. For this experiment, we randomly select 15 instances from the 75 instances in our benchmark. For each of the $4 \times 4 \times 4$ settings and for each instance, we then run our algorithm 5 times with the same timeout as mentioned before. Note for this experiment, in the VND based perform exploration procedure, we use inter-swap as $\mathcal{N}_1$ and adjacent-swap as $\mathcal{N}_2$.

- $d \in \{1, 2, 3, 4\}$
- $c \in \{1, 2, 3, 4\}$
- $p \in \{0.3, 0.4, 0.5, 0.6\}$

The ANOVA results are given in Table 5. From this table, c is the most significant factor due to the highest F-ratio, followed by p and d factors. The interval plots of these factors considering 95% Tukey's Honest Significant Difference (HSD) confidence intervals are shown in Figure 5. As we see, among the values tested, our EPS statistically significantly performs the best when $d = 2$, $c = 3$, and $p = 0.5$. In the rest of the paper, we will use these parameter values.

Table 5: ANOVA table for the three parameters.

Source	DF	SS	MS	F	P-value
d	3	3376.4	1125.46	4977.36	**0.000**
c	3	8758.4	2919.46	12911.32	**0.000**
p	3	3659.8	1219.93	5395.14	**0.000**
$d * c$	9	1649.4	183.27	810.51	**0.000**
$d * p$	9	954.1	106.01	468.83	**0.000**
$c * p$	9	349.3	38.81	171.65	**0.000**
Error	4763	1077.0	0.23		
Total	4799	19824.3			

As already mentioned, we compared the proposed EPS algorithm against the SA algorithm (Hancerliogullari et al., 2013) and the ILS algorithm (Sabar and Kendall, 2015). The SA algorithm was proposed for the same problem and was even evaluated on the same instances. So, its parameters are taken from the paper. However, ILS is originally proposed for ALP. So, in this paper, we again did a parameter calibration for this algorithm. There are two parameters in this algorithm that need to be tuned, minimum time varying mintv, and maximum time varying maxtv. In the original paper, the authors tested three values for mintv: 0.1-0.3, and two values for maxtv: 0.8 and 0.9. We also consider these values for testing. The Anova table and The interval plots of these factors considering 95% Tukey's Honest Significant Difference (HSD) confidence intervals are shown in Table 6 and Figure 6. As can be seen, ILS has its best performance with mintv = 0.1 and maxtv = 0.9, although there is no significant difference between maxtv values.

7.3. Greediness in Perturbation

In our EPS, in the perturbation phase, we use a greedy runway selection instead of a typical random selection. To test the effectiveness of our greedy selection over the random selection strategy, we have run experiments on all 75 instances with the initialisation method and the parameter values chosen above. Figure 7 shows a 95% confidence interval plot of the average errors. However, in this set of experiments, we separately use both insertion and swap based

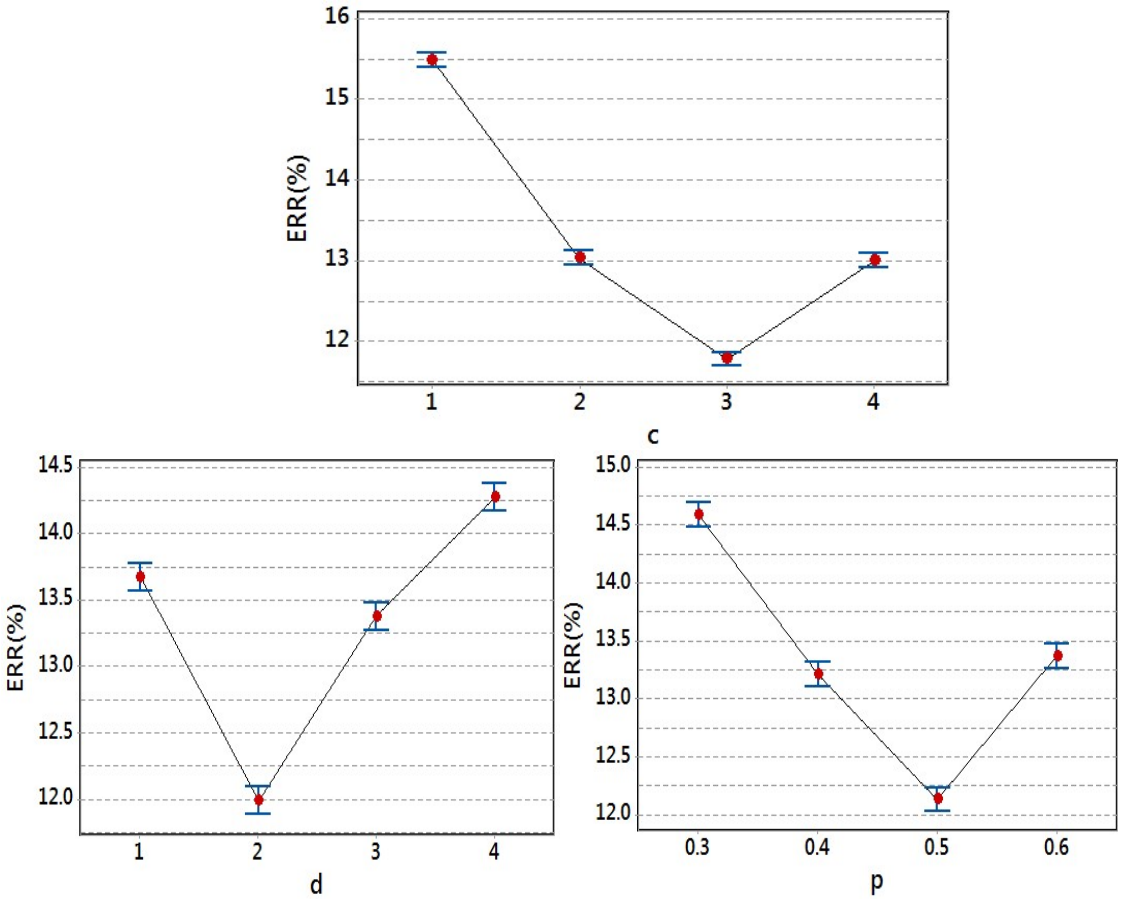

Figure 5: Interval plots with 95% Tukey's Honest Significant Difference (HSD) confidence intervals for c, d and p

Table 6: ANOVA table for the three parameters.

Source	DF	SS	MS	F	P-value
$mintv$	2	1798.22	899.109	466.29	**0.000**
$maxtv$	1	15.92	15.919	8.26	**0.004**
$mintv * maxtv$	2	4.10	2.052	1.06	**0.346**
Error	444	856.12	1.928		
Total	449	2674			

moves. From the plot, the greedy selection with swap based moves appears to be significantly better than the other three configurations.

7.4. Neighbourhood Operator Combination

In our EPS, in the VND based exploration procedure, we use two neighbourhood operators: inter-swap and adjacent swap. We mentioned before that using inter-swap as $\mathcal{N}_1$ and adjacent-swap as $\mathcal{N}_2$ produces better solutions than using the other way around. To test this, we have run experiments on all 75 instances with the initialisation method and other settings chosen so far. For this reason, four common neighbourhood move is tested: Inter Swap, Intra Insert, Adjacent Swap,

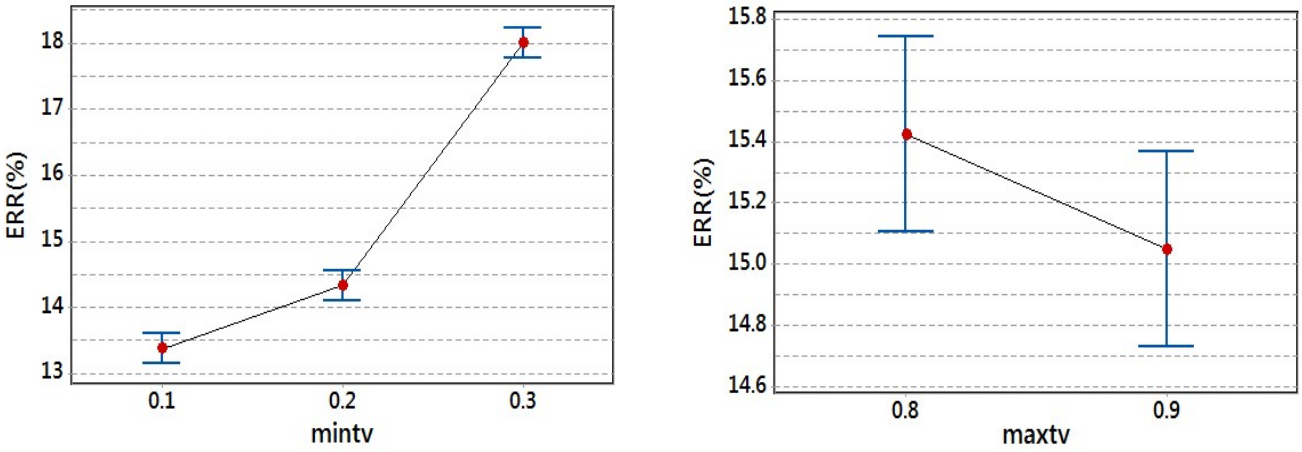

Figure 6: Interval plots with 95% Tukey's Honest Significant Difference (HSD) confidence intervals for mintv and maxtv

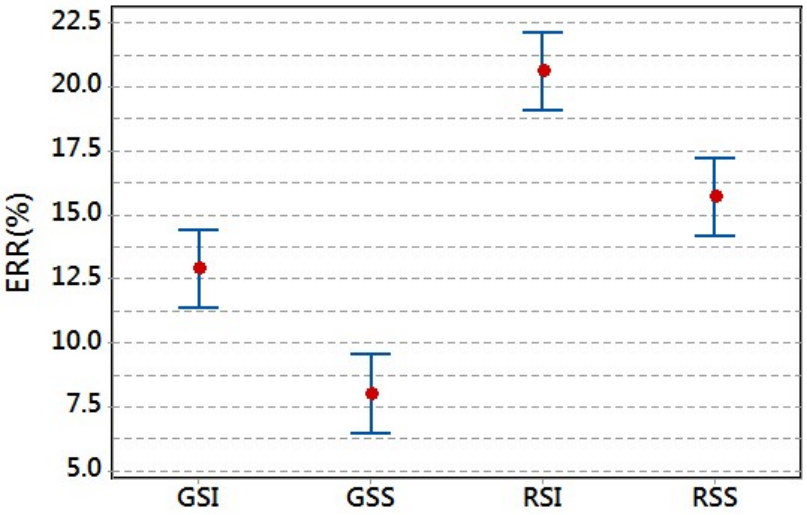

Figure 7: Comparison of greedy runway selection with swap move (GSS), greedy runway selection with insertion move (GSI), random runway selection with swap move (RSS), and random runway selection with insertion move (RSI)

and Intra Swap. Firstly, only one of them is considered as the solo move. Then three different Intra moves are tested with the only inter move. Figure 8 shows a 95% confidence interval plot of the average errors. From the plot, we see that firstly, using two moves is better than using only one, and among all cases with two neighbours, employing Adjacent Swap and Inter Swap has the better results than other cases. In addition, due to the nature of VND that first move is used more than the second one, the order of them will affect the performance of the algorithm. So, in all cases, we test two possible orders. As can be seen, our aforementioned selection is significantly better than the other way around, except for the Adjacent Swap/Intra Swap case. So in further experiments, we use inter-swap as $\mathcal{N}_1$ and adjacent-swap as $\mathcal{N}_2$ in the VND algorithm. Besides, in

the Intra Swap move, we are selecting one of the runways randomly and for another, the one with the highest objective is selected. Here, we are to show the effectiveness of this selection compared to two other possible selection approaches: (1) selecting both runways randomly, and (2) selecting both runways greedily: the one with the highest and the one with the lowest objectives. A 95% confidence interval plot of the average errors is given in Figure 9. As can be observed, our runway selection is significantly better than the other cases.

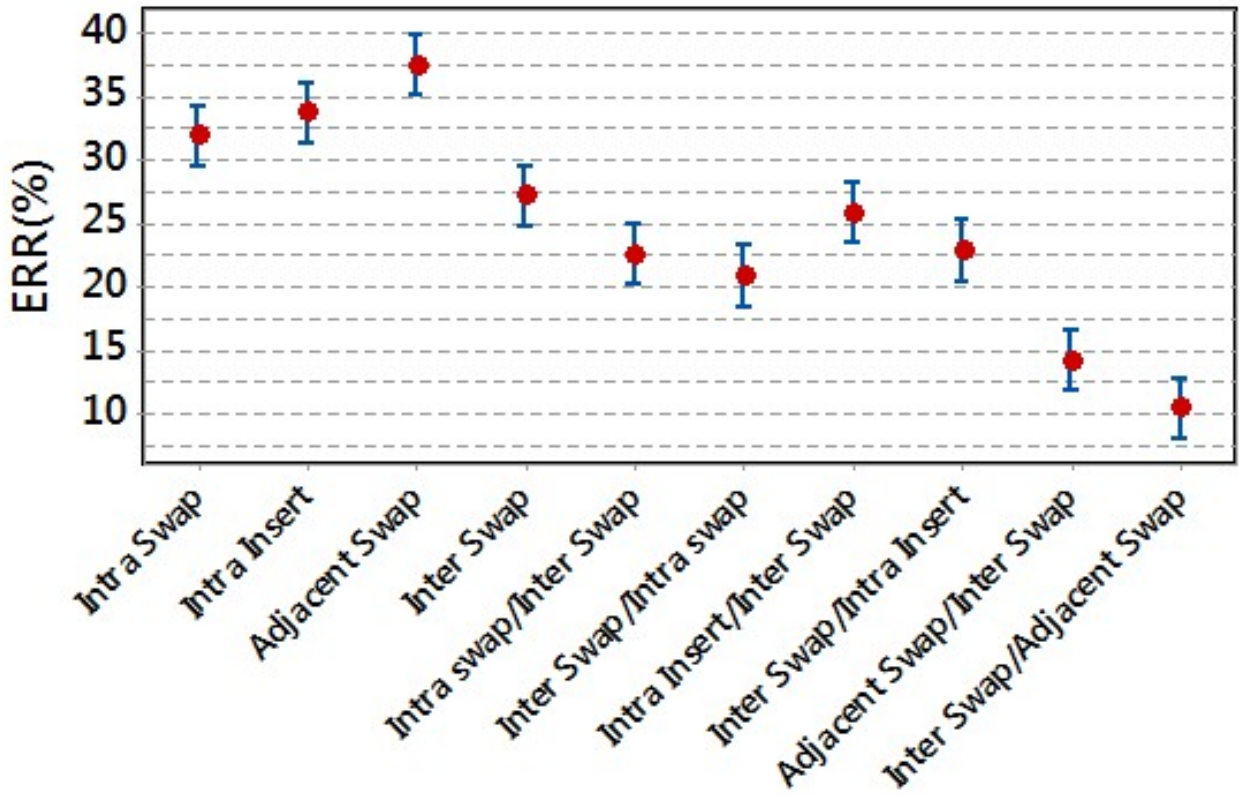

Figure 8: 95% confidence interval plot for different neighbourhood operator combinations

7.5. Overall Performance Comparison

We compare our final EPS algorithm with the SA algorithm (Hancerliogullari et al., 2013) and the ILS algorithm (Sabar and Kendall, 2015). Since initial solutions normally have impacts on the overall performance obtained by a metaheuristic algorithm, we combine the existing AATCSR and our SBH (SBHR to be exact) methods for initialisation with the ILS, the SA, and our EPS algorithms. These give us 6 combinations in total. We use all 75 instances in this experiment. To make it clearer, in our EPS, we use $p = 0.5$, $d = 2$, $c = 3$, SBH for initialisation, greedy runway selection in perturbation, and Inter-Swap/Adjacent-Swap combination in the VND based exploration with greedy selection in the Inter-Swap move.

For runway capacities restricted by $\lceil N/M \rceil$, the experimental results are presented in Table 7 and a 95% confidence interval plot is in Figure 10. SBH heuristic helps all three algorithms obtain better results than when using AATCSR heuristic, albeit the differences are not statistically

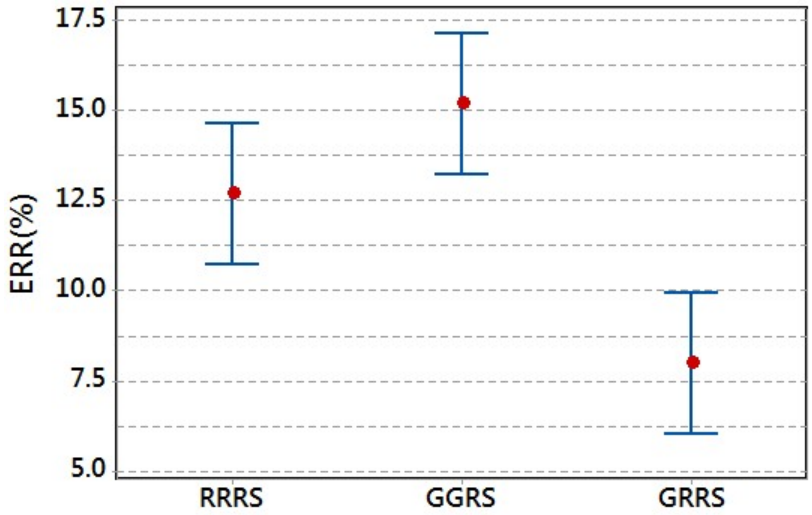

Figure 9: Comparison of highest and lowest runway selection in inter swap move (GGRS), greedy runway (highest one) and random runway selection in inter swap move (GRRS), random runway selection in inter swap move (RRRS)

significant for ILS and SA. Among all variants of the three algorithms, the EPS-SBH has the best performance and the difference is statistically significant except EPS-AATCSR. The EPS-AATCSR is significantly better than other algorithms except ILS-SBH. There is no statistically significant difference among SS-SBH, ILS-AATCSR and ILS-SBH.

Table 7: Performance of EPS, ILS, and SA using AATCSR and SBH for ASP problems

Problem	**EPS**		**ILS**		**SA**	
Instances	**AATCSR**	**SBH**	**AATCSR**	**SBH**	**AATCSR**	**SBH**
$N, M, \#$	ERR (%)	ERR (%)	ERR (%)	ERR (%)	ERR (%)	ERR (%)
15,2,5	2.74	**2.07**	4.50	4.31	10.12	8.04
15,3,5	1.21	**1.11**	15.07	9.83	11.64	7.32
15,4,5	5.10	**0.38**	16.85	13.66	9.24	13.66
20,2,5	9.85	**3.65**	16.48	8.05	47.06	15.11
20,3,5	7.16	**0.71**	17.71	24.32	44.90	34.90
20,4,5	2.60	**0.06**	23.24	17.99	36.92	24.24
20,5,5	11.81	**2.76**	28.71	19.28	30.62	20.68
25,2,5	9.81	**1.82**	9.81	18.98	76.83	36.02
25,3,5	10.08	**8.57**	18.68	24.12	56.89	45.48
25,4,5	8.20	**6.96**	48.43	42.22	87.15	70.50
25,5,5	6.59	**0.63**	39.58	25.76	45.33	30.72
50,2,5	15.40	**0.22**	24.94	7.67	185.81	21.66
50,3,5	19.67	**1.95**	57.22	17.05	204.59	52.18
50,4,5	13.73	**0.87**	53.24	51.14	95.47	78.62
50,5,5	9.71	**0.51**	146.61	53.43	250.78	73.70
Average	8.91	**2.15**	34.74	22.52	79.56	35.52

Number of instances for a given N, M pair Bold: minimum error

To investigate the convergence profiles of the algorithms over time, we also show a plot in Figure 11. In this plot, the time is represented by the values of k where our timeout is $T_{\max} =$

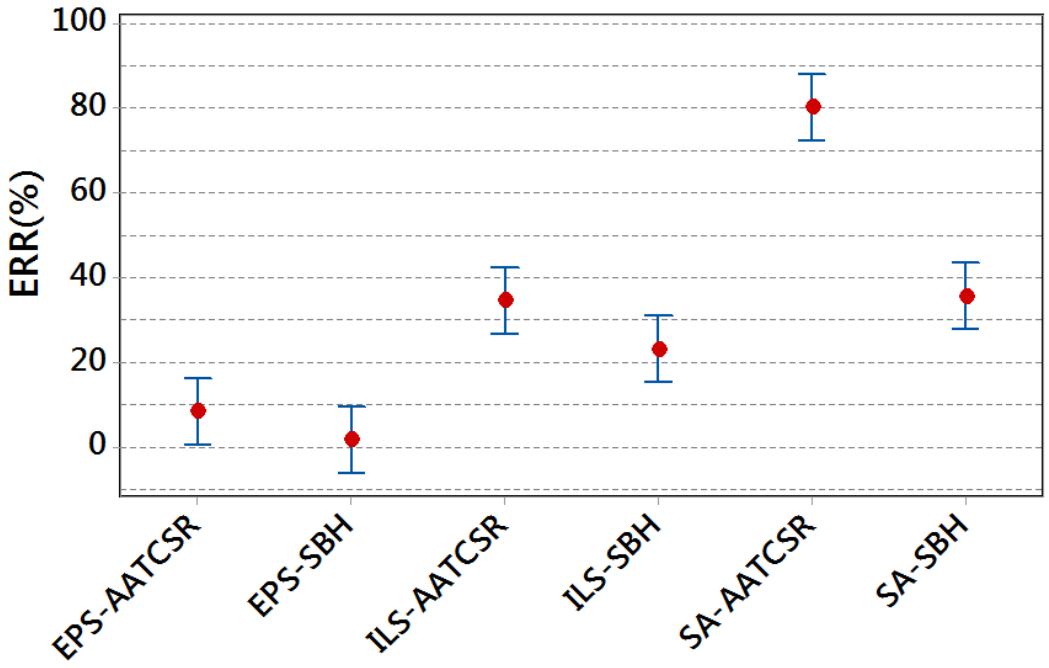

Figure 10: 95% confidence interval plot for ASP on Farhadi (2014) instances

$N \times M \times k$ ms CPU time. From this plot, we see that our EPS is much faster than SA and ILS algorithms with both SBH and AATCSR initialisation heuristics to converge to global or local optima. Moreover, with SBH heuristic, both SA and ILS algorithms do not make much progress over time. We ran the experiments up to $k = 20$ to check whether the progress profile changes, but could not find any.

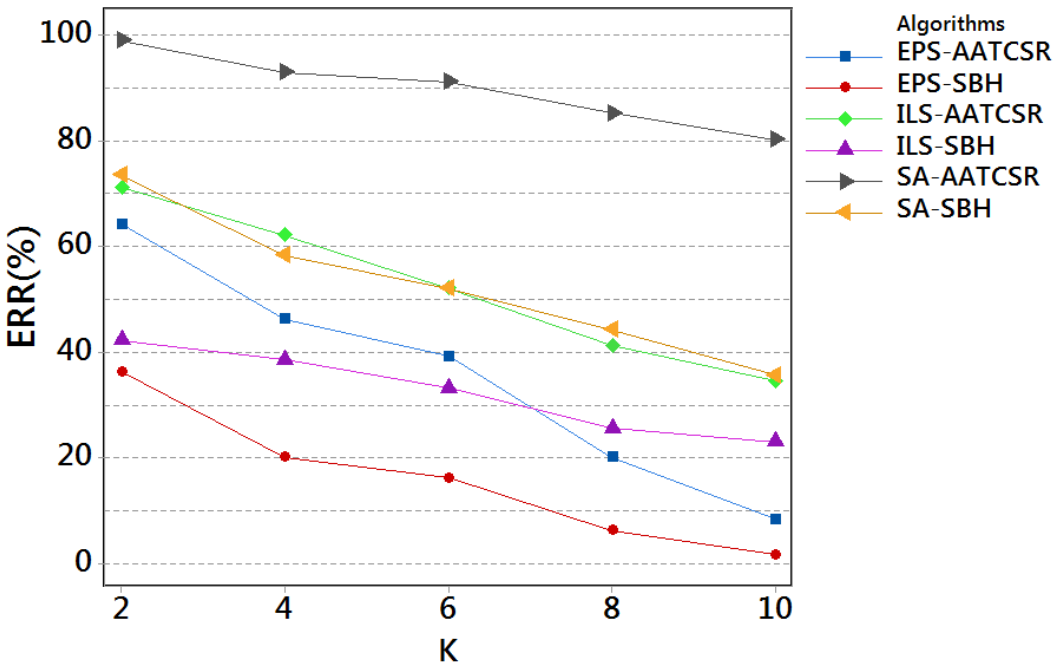

Figure 11: Covergence profiles of the algorithms for ASP on Farhadi (2014) instances

So far we have shown results for only one runway capacity combination where the number of aircraft per runway is at most $\lceil N/M \rceil$. In Table 8, and Figure 12, we show the results for a number of combinations with various standard deviations ranging from very low to very high. However, the conclusions that could be drawn from these results remain the same.

Table 8: Performance of EPS, ILS, and SA using AATCSR and SBH for a number of runway capacity combinations

Problem Instances	**Runway Capacity**	**EPS**		**ILS**		**SA**	
		AATCSR	**SBH**	**AATCSR**	**SBH**	**AATCSR**	**SBH**
$N, M, \#$	**Combination**	ERR (%)	ERR (%)	ERR (%)	ERR (%)	ERR (%)	ERR (%)
15,2,5	8-7	2.74	**2.07**	4.50	4.31	10.12	8.04
15,2,5	9-6	5.43	**0.53**	3.56	9.52	14.49	12.39
15,2,5	10-5	3.84	**1.07**	3.66	7.28	13.10	14.62
15,3,5	5-5-5	1.21	**1.11**	15.07	9.83	11.64	7.32
15,3,5	6-4-5	1.01	**0.79**	12.74	10.18	11.32	9.54
15,3,5	7-4-4	1.12	**0.07**	15.67	12.63	13.00	11.81
15,3,5	7-5-3	1.15	**0.58**	19.24	15.61	12.66	14.97
15,4,5	4-4-4-3	5.10	**0.38**	16.85	13.66	9.24	13.66
15,4,5	5-3-4-3	2.69	**0.83**	12.54	9.71	7.09	9.71
15,4,5	6-3-3-3	3.65	**0.08**	17.34	15.67	7.94	15.67
20,2,5	10-10	9.85	**3.65**	16.48	8.05	47.06	15.11
20,2,5	11-9	5.05	**2.34**	16.11	14.97	52.18	44.68
20,2,5	12-8	8.33	**1.72**	13.97	7.77	50.67	33.40
20,2,5	13-7	8.97	**8.86**	19.06	25.95	58.92	60.90
20,3,5	7-7-6	7.16	**0.71**	17.71	24.32	44.90	34.90
20,3,5	8-6-6	5.62	**3.01**	18.62	19.04	42.37	27.81
20,3,5	9-6-5	9.94	**2.74**	22.47	28.52	46.32	42.30
20,3,5	10-5-5	5.14	**1.60**	29.04	38.37	58.99	56.51
20,3,5	10-6-4	3.18	**1.46**	25.16	34.06	61.16	48.44
20,3,5	10-7-3	8.40	**1.99**	26.18	26.05	44.27	36.46
20,4,5	5-5-5-5	2.60	**0.06**	23.24	17.99	36.92	24.24
20,4,5	6-4-5-5	3.41	**0.80**	20.34	18.89	31.39	26.80
20,4,5	7-4-4-5	4.17	**0.18**	34.10	18.62	42.84	23.64
20,4,5	7-5-3-5	2.07	**0.12**	35.83	25.46	37.48	29.08
20,4,5	7-6-3-4	5.04	**1.39**	41.82	19.10	43.36	23.95
20,4,5	7-7-3-3	6.31	**0.38**	38.10	38.46	41.19	35.93
20,5,5	4-4-4-4-4	11.81	**2.76**	28.71	19.28	30.62	20.68
20,5,5	5-3-4-4-4	4.95	**0.30**	31.23	22.69	29.17	22.59
20,5,5	6-3-3-4-4	4.15	**0.28**	31.59	21.78	33.73	24.42
20,5,5	7-3-3-3-4	5.23	**0.00**	41.48	31.17	39.46	34.08
25,2,5	13-12	9.81	**1.82**	9.81	18.98	76.83	36.02
25,2,5	14-11	20.49	**0.61**	19.33	17.35	85.64	31.84
25,2,5	15-10	37.90	**0.86**	35.76	37.81	115.36	82.76
25,2,5	16-9	30.37	**3.05**	30.37	61.93	111.32	296.01
25,3,5	9-8-8	10.08	**8.57**	18.68	24.12	56.89	45.48
25,3,5	9-9-7	3.08	**1.92**	16.01	32.99	67.52	47.98
25,3,5	10-8-7	7.60	**1.07**	36.44	42.51	72.99	60.20
25,3,5	11-8-6	9.98	**0.54**	29.43	70.05	69.02	111.89
25,3,5	12-7-6	12.00	**3.32**	48.62	88.19	97.98	117.06
25,3,5	13-7-5	19.01	**2.38**	61.49	52.78	100.71	118.25
25,3,5	13-8-4	19.19	**1.82**	76.17	103.94	100.86	160.09
25,4,5	7-6-6-6	8.20	**6.96**	48.43	42.22	87.15	70.50
25,4,5	7-7-6-5	**6.22**	9.62	55.72	35.09	86.64	48.30
25,4,5	8-6-6-5	11.86	**1.09**	54.27	43.29	108.16	58.80
25,4,5	7-7-7-4	15.32	**0.51**	46.28	45.65	84.17	58.68
25,4,5	8-6-7-4	8.99	**1.74**	57.27	63.55	102.22	79.74

25,4,5	9-6-6-4	6.65	**4.12**	63.62	54.58	101.37	67.79
25,4,5	9-7-5-4	4.98	**0.78**	57.44	51.01	101.04	61.42
25,4,5	10-6-5-4	3.02	**2.64**	72.00	55.25	112.98	67.01
25,4,5	10-7-4-4	2.10	**0.85**	56.13	52.47	90.69	82.57
25,5,5	5-5-5-5-5	6.59	**0.63**	39.58	25.76	45.33	30.72
25,5,5	6-4-5-5-5	4.73	**1.61**	36.05	25.72	48.68	29.72
25,5,5	7-4-4-5-5	9.45	**0.13**	49.35	48.63	53.33	51.04
25,5,5	7-5-3-5-5	6.69	**0.22**	45.76	35.95	55.61	40.50
25,5,5	7-6-3-4-5	4.62	**1.97**	44.54	44.06	67.53	54.73
25,5,5	7-7-3-4-4	6.30	**1.01**	49.19	44.10	61.04	49.99
25,5,5	7-7-3-5-3	1.59	**1.37**	53.93	39.09	59.05	50.19
50,2,5	25-25	15.40	**0.22**	24.94	7.67	185.81	21.66
50,2,5	26-24	6.08	**0.95**	10.37	7.12	119.18	19.52
50,2,5	27-23	7.72	**0.00**	12.39	10.92	109.93	22.64
50,2,5	28-22	21.58	**2.58**	28.18	13.62	136.18	36.66
50,2,5	29-21	**3.40**	3.44	13.71	7.31	130.98	42.83
50,2,5	30-20	**0.22**	0.56	0.78	1.35	21.20	15.30
50,3,5	17-16-16	19.67	**1.95**	57.22	17.05	204.59	52.18
50,3,5	18-16-16	10.83	**3.40**	31.57	16.14	92.34	59.57
50,3,5	19-16-15	13.05	**4.67**	36.75	30.63	106.59	75.89
50,3,5	20-15-15	23.50	**4.18**	53.65	31.04	170.64	86.86
50,3,5	21-15-14	23.49	**3.81**	53.23	28.26	140.31	94.11
50,3,5	21-16-13	24.09	**9.56**	61.18	34.54	164.39	92.88
50,3,5	21-17-12	14.73	**3.10**	48.76	30.60	152.98	74.20
50,4,5	13-13-12-12	13.73	**0.87**	53.24	51.14	95.47	78.62
50,4,5	13-13-13-11	6.67	**1.92**	43.85	61.60	97.24	97.08
50,4,5	14-12-13-11	7.08	**0.67**	44.32	93.52	93.38	140.89
50,4,5	15-12-12-11	9.19	**6.35**	45.89	97.12	86.15	118.42
50,4,5	16-12-12-10	18.94	**11.19**	50.53	107.27	103.11	162.59
50,4,5	16-13-11-10	**8.22**	15.65	54.90	105.73	113.95	174.80
50,4,5	16-14-11-9	28.61	**8.37**	99.55	128.69	168.96	228.58
50,4,5	16-15-10-9	13.62	**0.45**	81.87	101.91	124.53	193.14
50,4,5	16-16-10-8	11.29	**6.89**	48.63	122.18	114.08	167.33
50,4,5	16-16-9-9	23.24	**5.94**	85.02	163.81	147.35	297.49
50,5,5	10-10-10-10-10	9.71	**0.51**	146.61	53.43	250.78	73.70
50,5,5	11-9-10-10-10	0.76	**0.18**	144.05	49.42	229.51	73.50
50,5,5	12-9-9-10-10	9.34	**0.70**	119.15	133.64	248.57	228.25
50,5,5	13-9-9-9-10	25.32	**2.08**	164.46	482.81	264.77	326.51
50,5,5	13-10-8-9-10	30.04	**4.30**	175.80	584.66	232.15	596.51
50,5,5	13-11-8-8-10	30.96	**11.30**	176.05	573.35	260.13	461.59
50,5,5	13-12-8-8-9	4.62	**2.06**	130.83	115.51	197.69	583.96
50,5,5	13-13-7-8-9	**6.31**	6.72	88.92	208.44	160.72	254.27
50,5,5	13-13-8-8-8	**1.99**	4.78	119.28	200.03	178.36	234.43
50,5,5	13 13 7 10 7	44.08	**8.93**	174.81	188.23	266.08	208.91

Number of instances for a given N, M pair Bold: minimum error

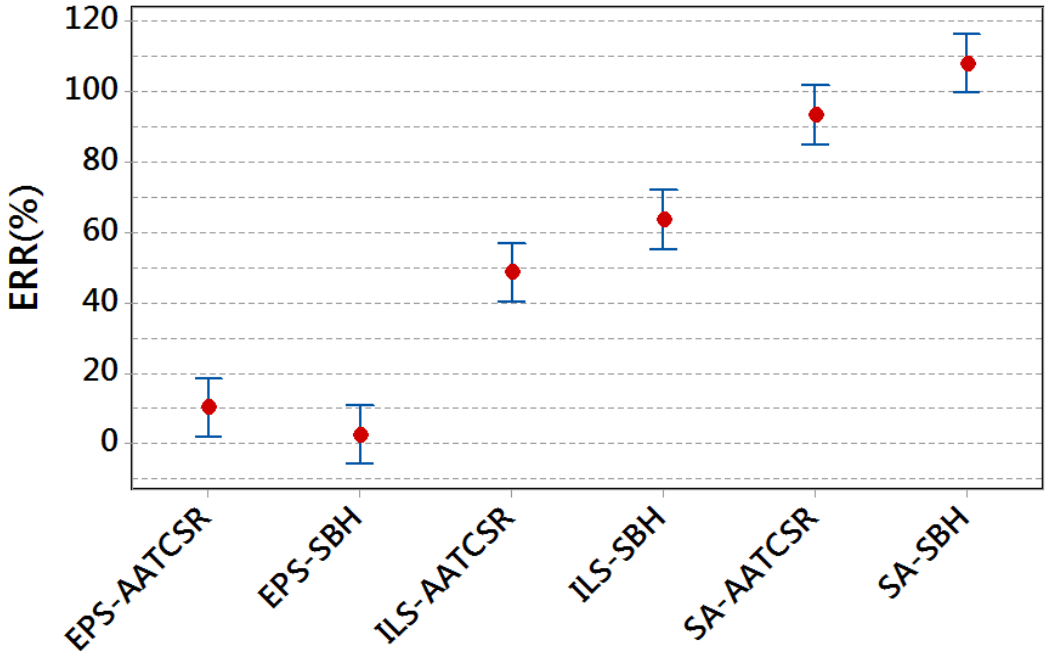

Figure 12: 95% confidence interval plot for ASP on Farhadi (2014) instances for different runway capacity combination.

To investigate which runway capacity combinations help any of the competitor algorithms return the best found solutions, in Figure 13, we show the average of the best found solutions against the standard deviations of runway capacity combinations. It is clear that in all problem sizes, the lowest standard deviations help find the best solutions. So for further analysis, we only show the results for those cases where the runway capacities are restricted by $\lceil N/M \rceil$.

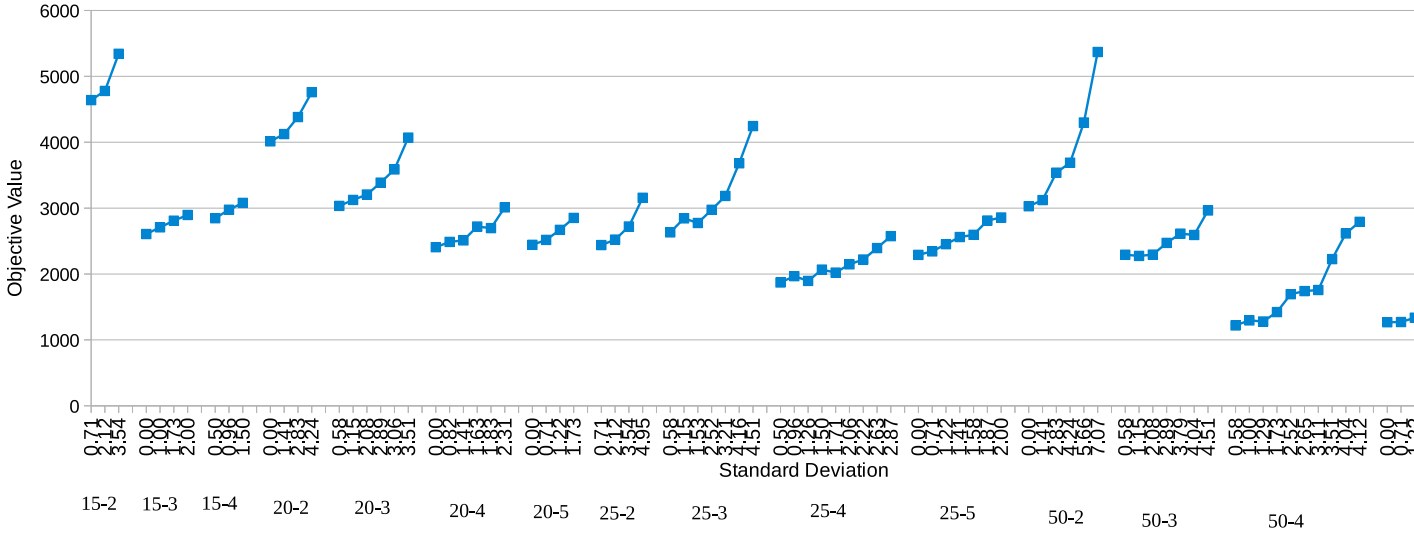

Figure 13: Average of the best found solutions over all competitor algorithms over various standard deviations of runway capacity combinations

7.6. Comparison on the Real Instances

In this section, we test the algorithms on the real-world instances taken from Milan airport. As can be seen in Table 9, the proposed EPS-SBH has the best results among the algorithms as it found 9 out of 12 of best solutions whereas EPS-AATCSR, ILS-AATCSR, ILS-SBH, SA-AATCSR and SA-SBH, respectively found 5, 2, 1, 0, 0 best solutions. In addition, this algorithm has the lowest average ERR value with 2.98. To analyze the results more precisely, a 95% confidence interval plot is performed that can be observed in Figure 14. The EPS-SBH has the best performance and the difference is statistically significant except with ILS-SBH.

7.7. Comparison on ALP Instances

ALP is a subset of ASP as it takes only landing operations into account. In this paper, our focus is on ASP where both landing and takeoffs are considered. However, the ALP has attracted much greater research interest than than ATP, and many researchers have studied ALP. The ILS algorithm (Sabar and Kendall, 2015) is one of the latest and best performing algorithm in the ALP literature. So, besides the ASP, in this paper, we also evaluate the three algorithms (SA, ILS, and EPS) on ALP instances. The ALP instances are obtained by replacing all takeoff operations with landing operations in the 75 ASP instances generated by Farhadi (2014). The results are given in Table 10 and a 95% confidence interval plot in Figure 15. As we see from these, EPS-SBH gives the best performance and significantly better than the other algorithms except EPS-AATCSR.

Table 9: Performance of EPS, ILS, and SA using AATCSR and SBH for ASP on Milan airport instances

Problem Instances	EPS		ILS		SA	
	AATCSR	SBH	AATCSR	SBH	AATCSR	SBH
	ERR (%)	ERR (%)	ERR (%)	ERR (%)	ERR (%)	ERR (%)
FPT01	**0.00**	3.33	6.67	3.33	34.67	6.67
FPT02	3.03	**0.00**	3.03	3.03	52.73	9.09
FPT03	14.29	**0.00**	14.29	14.29	61.90	14.29
FPT04	15.38	**0.00**	26.92	7.69	15.38	15.38
FPT05	8.33	**0.00**	25.00	4.17	37.50	12.50
FPT06	125.00	**0.00**	135.00	87.50	127.50	12.50
FPT07	12.50	**0.00**	12.50	**0.00**	91.25	18.75
FPT08	**0.00**	**0.00**	83.33	33.33	100.00	75.00
FPT09	**0.00**	5.56	16.67	27.78	41.11	38.89
FPT10	**0.00**	11.60	**0.00**	78.57	135.71	135.71
FPT11	114.29	**15.30**	114.29	114.29	255.71	157.14
FPT12	**0.00**	**0.00**	**0.00**	60.00	160.00	80.00
Average	24.40	**2.98**	36.47	36.16	92.79	47.99

Bold: minimum error

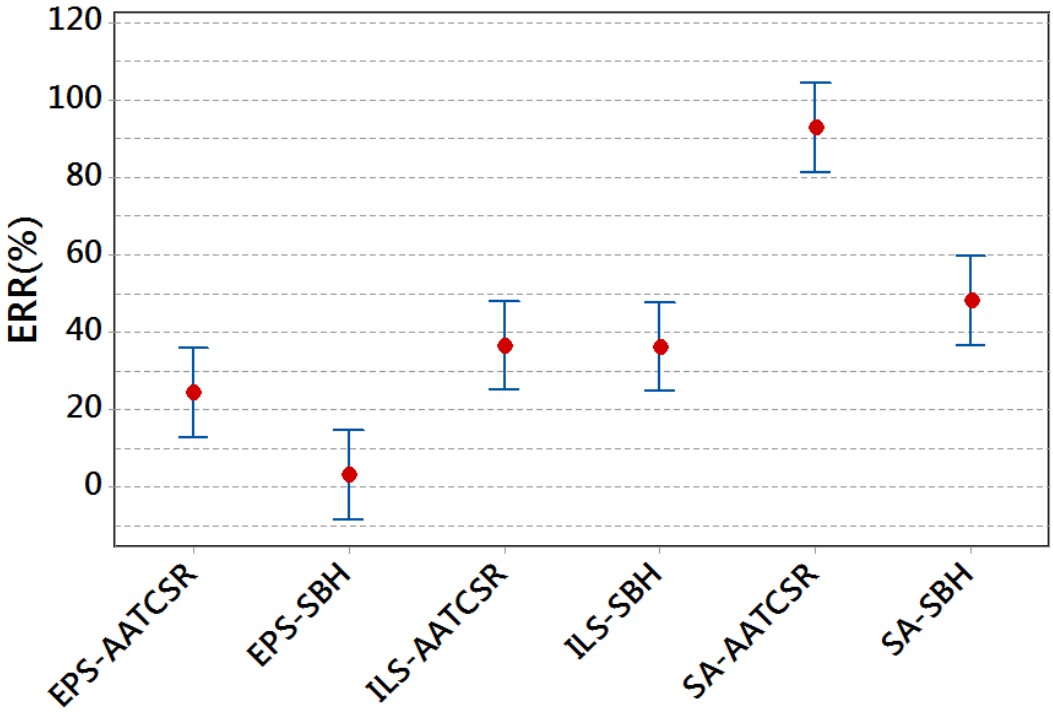

Figure 14: 95% confidence interval plot for ASP on Milan airport instances

8. Overall Discussion

In this paper, we have proposed two new constructive heuristics and a new constraint guided local search algorithm for ASP.

The two proposed constructive heuristics are evaluated against the best known constructive heuristic AATCSR (Hancerliogullari et al., 2013). Table 4 shows that the proposed heuristics

Table 10: Performance of the algorithms on ALP instances

Problem Instances	EPS		ILS		SA	
	AATCSR	SBH	AATCSR	SBH	AATCSR	SBH
$N, M, \#$	ERR (%)	ERR (%)	ERR (%)	ERR (%)	ERR (%)	ERR (%)
15, 2, 5	3.20	**0.90**	2.85	1.32	7.68	3.95
15, 3, 5	6.12	**0.00**	9.35	9.72	8.12	10.88
15, 4, 5	5.00	**0.00**	10.35	8.47	12.01	9.68
20, 2, 5	**4.02**	6.10	4.56	5.10	12.46	13.82
20, 3, 5	6.59	**2.04**	8.94	12.50	18.71	16.63
20, 4, 5	4.71	**0.00**	14.36	9.23	13.01	10.45
20, 5, 5	8.35	**3.89**	7.49	11.60	19.37	14.78
25, 2, 5	5.26	10.19	**1.10**	17.95	27.62	18.81
25, 3, 5	0.92	**0.83**	11.00	7.79	21.06	21.57
25, 4, 5	6.09	**0.74**	21.51	18.14	28.10	24.20
25, 5, 5	3.45	**0.00**	19.09	17.88	26.93	20.33
50, 2, 5	4.15	9.51	**2.80**	7.56	37.74	21.70
50, 3, 5	**1.39**	3.01	25.78	13.10	25.93	33.22
50, 4, 5	10.54	**9.50**	35.03	24.98	41.10	26.04
50, 5, 5	16.65	**1.20**	39.91	29.79	57.59	34.59
average	5.76	**3.19**	14.27	13.01	23.83	18.71

Number of instances for a given N, M pair Bold: minimum error

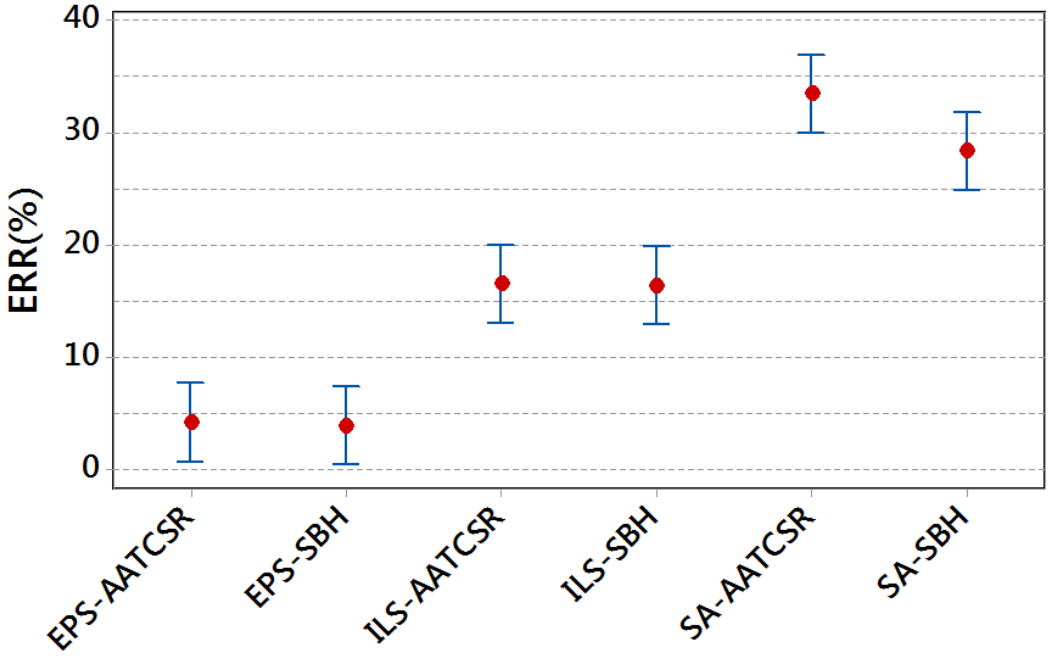

Figure 15: 95% confidence interval plot for ALP

obtained ARPDs 8.9% and 6.7% while AATCSR obtained 59.9%. AATCSR constructs the solution simply by appending each next aircraft based on a precomputed index without performing any search. In contrast, the two proposed heuristics perform search on the already constructed partial solutions using the objective function to find the best positions for the next aircraft. The guided search in the two proposed heuristics clearly makes the difference between their performance and

that of AATCSR, which does not have any search.

The proposed local search algorithm embeds constraint-guided strategies in its exploration and perturbation phases. Taking the time constraint into account, an adjacent swap operator is used in the exploration phase instead of using a typical intra-swap operator. The adjacent swap exchanges two aircraft that in successive positions in a runway while the intra-swap exchanges any two aircraft in a runway. Taking the objective function into account, runways that have the most objective value are selected for the operators to be applied on. The runway selection allows the search to fix the most problematic part of the current solution. window, moving an aircraft to a position that is far from its current position might not be effective and reasonable. The latter one swaps two aircraft that are in two different runways. In this operator, instead of the typical random runway selection, we insert the greediness into it by selecting one of the runway greedily, the one with highest objective function created. This allows us to fix the most problematic part of the current solution. Results in Figures 7, 8, and 9 demonstrate that the proposed constraint-guided operators and greedy runway selections are significantly better than the typical random counterparts. Table 7 shows that our constraint guided local search significantly outperforms SA (Hancerliogullari et al., 2013) and ILS (Sabar and Kendall, 2015) as they obtain 2.15%, 22.52%, and 35.52% ARPDs respectively on the instances from Farhadi (2014). Note that the SA and ILS algorithm do not uses any constraint guided strategies and ddepend on random or exhaustive selections. Table 9 shows similar performances on instances from Furini et al. (2012) while Table 15 shows on ALP instances.

In this paper, we also have done another investigation as algorithms are evaluated on different runway capacity combinations. This experiment helps up see the performance of the algorithms on different runway combination scenarios and to find the best runway capacities. Table 8 shows that in all problem sizes, the lowest standard deviations help find the best solutions. Moreover, the proposed algorithm outperformes the other algorithm significantly in all runway capacity combinations.

9. Conclusion

We have studied multiple runway aircraft sequencing problems that involve both landing and takeoff operations. Given pairwise separation times, and time windows for all aircraft, we minimise a weighted tardiness of the schedule to be produced. Existing methods in a typical way use heuristics and metaheuristics that in general lack problem specific knowledge. In this work we have demonstrated that structural knowledge obtained by looking at the constraints and the objective functions could be exploited to improve both constructive search methods and local search methods. Such knowledge could not only be used in solution evaluation, but also be used in each phase of a search method e.g. in neighbourhood generation, in perturbation steps. Our motivation comes from the constraint optimisation paradigm in artificial intelligence We have performed a thorough evaluation of the constraint optimisation strategies that we propose both using craft instances and instances from real airports. We show that our proposed strategies are effective and thus the resultant search algorithms significantly outperform existing state-of-the-art aircraft sequencing algorithms.

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*, pages 71–90.

ACI (2012). *Airports Council International*. `http://www.aci.aero/`.

Adler, N., Liebert, V., and Yazhemsky, E. (2013). Benchmarking airports from a managerial perspective. *Omega*, 41(2):442–458.

Al-Salem, A., Farhadi, F., Kharbeche, M., and Ghoniem, A. (2012). Multiple-runway aircraft sequencing problems using mixed-integer programming. In *IIE Annual Conference. Proceedings*, page 1. Institute of Industrial Engineers-Publisher.

Artiouchine, K., Baptiste, P., and Dürr, C. (2008). Runway sequencing with holding patterns. *European Journal of Operational Research*, 189(3):1254–1266.

Atkin, J. A., Burke, E. K., Greenwood, J. S., and Reeson, D. (2007). Hybrid metaheuristics to aid runway scheduling at london heathrow airport. *Transportation Science*, 41(1):90–106.

Balakrishnan, H. and Chandran, B. G. (2010). Algorithms for scheduling runway operations under constrained position shifting. *Operations Research*, 58(6):1650–1665.

Balakrishnan, H. and Chandran, B. G. (2014). Optimal large-scale air traffic flow management.

Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., Peterson, E., Sherry, L., Trani, A., and Zou, B. (2010). Total delay impact study: Institute of transportation studies. *University of California, Berkeley*.

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

Belhaiza, S., Hansen, P., and Laporte, G. (2014). A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research*, 52:269–281.

Bencheikh, G., Boukachour, J., Alaoui, A. E. H., and Khoukhi, F. (2009). Hybrid method for aircraft landing scheduling based on a job shop formulation. *International Journal of Computer Science and Network Security*, 9(8):78–88.

Benlic, U., Brownlee, A. E., and Burke, E. K. (2016). Heuristic search for the coupled runway sequencing and taxiway routing problem. *Transportation Research Part C: Emerging Technologies*, 71:333–355.

Bennell, J. A., Mesgarpour, M., and Potts, C. N. (2011). Airport runway scheduling. *4OR: A Quarterly Journal of Operations Research*, 9(2):115–138.

Bennell, J. A., Mesgarpour, M., and Potts, C. N. (2013). Airport runway scheduling. *Annals of Operations Research*, 204(1):249–270.

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

Bianco, L., DellOlmo, P., and Giordani, S. (1997). Scheduling models and algorithms for tma traffic management.

Bianco, L., Rinaldi, G., and Sassano, A. (1987). A combinatorial optimization approach to aircraft sequencing problem. In *Flow Control of Congested Networks*, pages 323–339. Springer.

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

BNE (2017). *Brisbane Airport Corporation*. `http://www.bne.com.au/`.

Briskorn, D. and Stolletz, R. (2014). Aircraft landing problems with aircraft classes. *Journal of Scheduling*, 17(1):31–45.

BTS (2017). *Bureau of Transportation Statistics*. `ttps://www.bts.gov/`.

Castelli, L., Pesenti, R., and Ranieri, A. (2011). The design of a market mechanism to allocate air traffic flow management slots. *Transportation research part C: Emerging technologies*, 19(5):931–943.

Ciesielski, V. and Scerri, P. (1997). An anytime algorithm for scheduling of aircraft landing times using genetic algorithms. *Australian Journal of Intelligent Information Processing Systems*, 4(3/4):206–213.

D'Ariano, A., Pacciarelli, D., Pistelli, M., and Pranzo, M. (2015). Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area. *Networks*, 65(3):212–227.

D'Ariano, A., Pistelli, M., and Pacciarelli, D. (2012). Aircraft retiming and rerouting in vicinity of airports. *IET Intelligent Transport Systems*, 6(4):433–443.

De Maere, G. and Atkin, J. A. (2015). Pruning rules for optimal runway sequencing with airline preferences. *Lecture Notes in Management Science*, 7:76–82.

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

Eurocontrol (2017). *Eurocontrol - Driving excellence in ATM performance*. `http://www.eurocontrol.int/`.

FAA (2015). *American federal aviation administration safety alert for operators (safo).* https://www.faa.gov/other_visit/aviation_industry/airline_operators/airline_safety/safo/.

Fahle, T., Feldmann, R., Götz, S., Grothklags, S., and Monien, B. (2003). The aircraft sequencing problem. In *Computer science in perspective*, pages 152–166. Springer.

Farhadi, F. (2014). *Runway Operations Management: Models, Enhancements, and Decomposition Techniques.* PhD thesis, University of Massachusetts - Amherst.

Frair, L. (1984). Airport noise modelling and aircraft scheduling so as to minimize community annoyance. *Applied Mathematical Modelling*, 8(4):271–281.

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

Furini, F., Persiani, C. A., and Toth, P. (2012). Aircraft sequencing problems via a rolling horizon algorithm. In *International Symposium on Combinatorial Optimization*, pages 273–284. Springer.

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

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

Ghoniem, A., Sherali, H. D., and Baik, H. (2014). Enhanced models for a mixed arrival-departure aircraft sequencing problem. *INFORMS Journal on Computing*, 26(3):514–530.

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

Hancerliogullari, G., Rabadi, G., Al-Salem, A. H., and Kharbeche, M. (2013). Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem. *Journal of Air Transport Management*, 32:39–48.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467.

Heathrow (2017). *runway alternation — noise — heathrow.* https://www.heathrow.com/noise/heathrow-operations/runway-alternation.

Heblij, S. J. and Wijnen, R. A. (2008). Development of a runway allocation optimisation model for airport strategic planning. *Transportation Planning and Technology*, 31(2):201–214.

Hu, X.-B. and Di Paolo, E. (2008). Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling. *IEEE Transactions on Intelligent Transportation Systems*, 9(2):301–310.

Idris, H., Delcaire, B., Anagnostakis, I., Hall, W., Pujet, N., Feron, E., Hansman, R., Clarke, J.-P., and Odoni, A. (1998). Identification of flow constraint and control points in departure operations at airport systems. In *Guidance, Navigation, and Control Conference and Exhibit*, page 4291.

Janssen, S. A., Centen, M. R., Vos, H., and van Kamp, I. (2014). The effect of the number of aircraft noise events on sleep quality. *Applied Acoustics*, 84:9–16.

Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1982). Recent developments in deterministic sequencing and scheduling: a survey. In *Deterministic and stochastic scheduling*, pages 35–73. Springer.

Lieder, A. and Stolletz, R. (2016). Scheduling aircraft take-offs and landings on interdependent and heterogeneous runways. *Transportation research part E: logistics and transportation review*, 88:167–188.

Mehta, V., Reynolds, T., Ishutkina, M., Joachim, D., Glina, Y., Troxel, S., Taylor, B., and Evans, J. (2013). Airport surface traffic management decision support: Perspectives based on tower flight data manager prototype. Technical report.

Montoya, J., Rathinam, S., and Wood, Z. (2014). Multiobjective departure runway scheduling using dynamic programming. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):399–413.

Moser, I. and Hendtlass, T. (2007). Solving dynamic single-runway aircraft landing problems with extremal optimisation. In *Computational Intelligence in Scheduling, 2007. SCIS'07. IEEE Symposium on*, pages 206–211. IEEE.

Murça, M. C. R. and Müller, C. (2015). Control-based optimization approach for aircraft scheduling in a terminal area with alternative arrival routes. *Transportation research part E: logistics and transportation review*, 73:96–113.

Newton, M. H., Pham, D. N., Sattar, A., and Maher, M. (2011). Kangaroo: An efficient constraint-based local search system using lazy propagation. In *International Conference on Principles and Practice of Constraint Programming*, pages 645–659. Springer.

Osaba, E., Yang, X.-S., Diaz, F., Lopez-Garcia, P., and Carballedo, R. (2016). An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. *Engineering Applications of Artificial Intelligence*, 48:59–71.

Ozkurt, N., Sari, D., Akdag, A., Kutukoglu, M., and Gurarslan, A. (2014). Modeling of noise pollution and estimated human exposure around istanbul atatürk airport in turkey. *Science of the Total Environment*, 482:486–492.

Pinol, H. and Beasley, J. E. (2006). Scatter search and bionomic algorithms for the aircraft landing problem. *European Journal of Operational Research*, 171(2):439–462.

Ravidas, A., Rathinam, S., and Wood, Z. (2013). An optimal algorithm for a two runway scheduling problem. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering*, 227(7):1122–1129.

Rodríguez-Díaz, A., Adenso-Díaz, B., and González-Torre, P. L. (2017a). Minimizing deviation from scheduled times in a single mixed-operation runway. *Computers & Operations Research*, 78:193–202.

Rodríguez-Díaz, A., Adenso-Díaz, B., and González-Torre, P. L. (2017b). A review of the impact of noise restrictions at airports. *Transportation Research Part D: Transport and Environment*, 50:144–153.

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. *Omega*, 56:88–98.

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

Samà, M., DAriano, A., DAriano, P., and Pacciarelli, D. (2014). Optimal aircraft scheduling and routing at a terminal control area during disturbances. *Transportation Research Part C: Emerging Technologies*, 47:61–85.

Samà, M., DAriano, A., DAriano, P., and Pacciarelli, D. (2017). Scheduling models for optimal aircraft traffic control at busy airports: tardiness, priorities, equity and violations considerations. *Omega*, 67:81–98.

SESAR (2017). *SESAR Joint Undertaking — High performing aviation for Europe*. `http://www.sesarju.eu/`.

Sherali, H., Ghoniem, A., Baik, H., and Trani, A. (2010). A combined arrival-departure aircraft sequencing problem. *Manuscript, Grado Department of Industrial and Systems Engineering (0118). Virginia Polytechnic Institute and State University*, 250.

Soykan, B. (2016). *A hybrid Tabu/Scatter Search algorithm for simulation-based optimization of multi-objective runway operations scheduling*. PhD thesis, Old Dominion University.

Soykan, B. and Rabadi, G. (2016). A tabu search algorithm for the multiple runway aircraft scheduling problem. In *Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling*, pages 165–186. Springer.

Stiverson, P. W. (2010). *A study of heuristic approaches for runway scheduling for the Dallas-Fort Worth Airport*. PhD thesis, Texas A & M University.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European journal of Operational research*, 47(1):65–74.

Trivizas, D. A. (1998). Optimal scheduling with maximum position shift (mps) constraints: A runway scheduling application. *The Journal of Navigation*, 51(2):250–266.

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

Wang, S. (2009). Solving aircraft-sequencing problem based on bee evolutionary genetic algorithm and clustering method. In *Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on*, pages 157–161. IEEE.

Wen, M., Larsen, J., and Clausen, J. (2005). An exact algorithm for aircraft landing problem. Technical report.