# Improved rolling horizon approaches to the aircraft sequencing problem

**Article** *in* Journal of Scheduling · January 2015

DOI: 10.1007/s10951-014-0415-8

CITATIONS
16

READS
100

**4 authors**, including:

Fabio Furini
Paris Dauphine University
**40** PUBLICATIONS **219** CITATIONS
SEE PROFILE

Martin Philip Kidd
Technical University of Denmark
**12** PUBLICATIONS **298** CITATIONS
SEE PROFILE

Paolo Toth
University of Bologna
**14** PUBLICATIONS **601** CITATIONS
SEE PROFILE

# Improved rolling horizon approaches to the aircraft sequencing problem

Fabio Furini

*LAMSADE, Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris, France*
*fabio.furini@dauphine.fr*

Martin Philip Kidd

*DEI, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy*
*martin.kidd@unibo.it*

Carlo Alfredo Persiani

*ENAV S.p.A, Italian Agency for Air Navigation Services, Via Salaria 716 Roma, Italy*
*carlo.persiani@enav.it*

Paolo Toth

*DEI, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy*
*paolo.toth@unibo.it*

**Abstract**

In a scenario characterized by a continuous growth of air transportation demand, the runways of large airports serve hundreds of aircraft every day. Aircraft sequencing is a challenging problem that aims to increase runway capacity in order to reduce delays as well as the workload of air traffic controllers. In many cases, the air traffic controllers solve the problem by using the simple "First-Come-First-Serve" (FCFS) rule. In this paper we present a rolling horizon approach which partitions a sequence of aircraft into chunks and solves the Aircraft Sequencing Problem (ASP) individually for each of these chunks. Some rules for deciding how to partition a given aircraft sequence are proposed and their effects on solution quality investigated. Moreover, two Mixed Integer Linear Programming (MILP) models for the ASP are reviewed in order to formalize the problem, and a tabu search heuristic is proposed for finding solutions to the ASP in a short computation time. Finally, we develop an IRHA which, by using different chunking rules, is able to find solutions significantly improving on the FCFS rule for real world air traffic instances from Milano Linate Airport.

*Keywords:* Aircraft Sequencing Problem, MILP, Rolling Horizon Algorithms

## 1. Introduction

Many researchers identify airports as the bottlenecks of the air transportation system, where a relevant share of the delays is often generated by an inefficient management of the runway capacities. The need for increasing efficiency and capacity while reducing delays persuade many researchers to investigate runway scheduling problems. In this paper we address the *Aircraft Sequencing Problem* (ASP) which is the problem of optimally scheduling the departures and the arrivals operations of airport runways. The ASP can be formally defined as follows. On input we are given the set $I = \{1, 2, \ldots, n\}$ of aircraft (including both arrivals and departures) that will use the runway during a predefined time window . For each aircraft $i \in I$, let $t_I^i$ be the scheduled arrival/departure time, $d_i$ the maximum acceptable delay and $w_i$ the cost associated with one time unit of delay between the scheduled and actual arrival/departure times. For each ordered pair of aircraft $(i_1, i_2) \in I \times I$ a minimum separation of $t_S^{i_1 i_2}$ time units must be maintained between the actual arrival/departure times of $i_1$ and $i_2$. The goal of the problem is to find a sequence of the given aircraft, together with the actual arrival/departure times $z_i$ for each $i \in I$, so as to minimize the global weighted delay of the aircraft with respect to their scheduled times, *i.e.* to minimize the cost function:

$$\sum_{i \in I} w_i (z_i - t_I^i),$$

subject to the constraints mentioned above. All the input data are positive integers and all information is known a priori, *i.e.* we are considering the static case of the ASP.

The ASP is an NP-hard problem. For further details see [7] where it is shown that the ASP can be viewed as either a job shop scheduling problem with release times and sequence-dependent processing times (but zero setup times), or a job shop scheduling problem with release times and sequence-dependent setup times (but zero processing times).

A number of different objectives and constraints may be taken into account for the ASP, as discussed by Bennel *et al.* [10]. This may be influenced by various stakeholders such as air-traffic controllers, airports, arlines, government, etc., and may include objectives such as minimizing the average delay or minimizing the average makespan. There are also various ways of restricting the amount of delay an aircraft may experience by including additional constraints. In this paper we consider the variant of the problem studied in [7].

The most important additional delay constraint that may also be included is so-called *constrained position shifting* (CPS), where each aircraft in the sequence may only deviate by a certain amount of positions, called the *maximum position shift* (MPS), from its position in the *First-Come-First-Serve* (FCFS)

order (in which aircraft are ordered according to their scheduled arrival/departure times). For this version of the ASP it has recently been shown that the problem can be efficiently solved for small values of the MPS by using dynamic programming [6, 16]. However, to the best of our knowledge current exact approaches in the literature for the version of the ASP where CPS is not considered are not capable of solving long sequences of aircraft for online applications.

In many cases, the air traffic controllers solve the problem by using the simple FCFS rule. However, a large margin of improvement is possible with the use of optimization techniques. In this paper, we present a heuristic rolling horizon approach which partitions a sequence of aircraft into chunks and solves the ASP individually for each of these chunks. This algorithm is capable of solving large sequences of aircraft in a short computing time, and is therefore ideal for online applications. Moreover, CPS is implicitly taken into account by restricting the position of an aircraft within a chunk in the rolling horizon algorithm, while the chunk sizes we consider are too long for the dynamic programming to be used efficiently without explicitly imposing small MPS values.

### *1.1. Aeronautical background*

The aircraft operations close to large airports are generally managed by three Air Traffic Control Authorities that provide Air Navigation Services for the controlled airspace: the Approach Radar Control, the Ground Control and the Tower Control. Since the runway can be used by one aircraft at a time, the Approach (or Terminal Area, depending on local configuration) Radar Control is responsible for sequencing the inbound traffic on the Instrumental Landing System (ILS) which is a radio path that the pilots lock and follow in order to perform precision landings. The Ground Control is responsible for sequencing the outbound traffic up to the runway holding points which are the positions located at a safe distance from the runway where the pilots wait for their departure clearance. The Tower Control is responsible for integrating these two flows.

The air traffic controllers are also responsible for ensuring that a minimum separation between the arrival/departure of one aircraft and the arrival/departure of another aircraft is maintained, where different notions of separation may be defined. Two of the most applied notions are "radar separation," which is a longitudinal spacing of 5NM (3NM in congested area) and a vertical spacing of 1000ft, and "wake turbulence separation," which is a time spacing that depends on the dimensions and weight of the aircraft. The International Civil Aviation Organization (ICAO) defines four categories of wake turbulence, namely Light (L), Medium (M), Heavy (H) and Super (which includes only the Airbus A380). Furthermore, several factors are considered in the application of separation between aircraft movements, including the

inbound and outbound routes of the aircraft, the aircraft types and performances, the weather conditions and the wake turbulence category.

It is possible to define the runway occupancy slot for both inbound and outbound traffic using the Estimated Landing Time (ELT) and the Estimated Take-off Time (ETOT) respectively. The ELT may be calculated as the sum of two contributions, namely the Expected Approach Time (EAT) and the final approach flight time. The EAT for an aircraft may be considered as the starting time of the final stage of the approach, and it can be approximated using the time at which the aircraft locks the ILS. Once an aircraft has started its final approach, changes in the landing sequence cannot be made without demanding a large workload from the controllers, and thus they may be considered only as contingency operations. Delay actions are realistic only before this phase. The aircraft speeds during the final approach phase are similar and can be approximated with a constant value that represents the final approach flight time.

Once boarding operations have been terminated, the pilots of a departing aircraft request startup clearance from the Ground Controller. If the delay foreseen at the holding point is acceptable, the startup is approved and the aircraft is instructed to taxi (or possibly to perform a previous push back that is an exit operation from the parking stand). The average taxi times from the various parking areas of an airport are reported in the Aeronautical Information Publications. The ETOT may be calculated as the sum of two contributions, namely the Estimated off Block Time (EOBT), which is the scheduled time given by the companies at which a flight will be able to start to taxi, and the taxi time as previously reported. Delay actions for a departure are possible before the engines start up (at the parking). After that moment delay actions are not efficient from a fuel consumption perspective. The standard taxi time (10 minutes) is an approximation due to the fact that we don't know the exact parking position of each flight. Moreover, the taxi time also depends on other factors such as constraints into the manoeuvring area (e.g. taxiway closed or restrictions due to aircraft size or work in progress) that we do not take into account. For these reasons we assume a standard taxi time for all the departures. The possibility of refining the taxi time makes it possible to have more precise ETOT, but it does not change the solution approach from a logical point of view since our algorithms take as input the ETOT and the ELT (as scheduled time of runway utilization).

It is easy to see that performing delay actions on departures is a simpler, cheaper and safer task when compared to arrivals. It is not realistic that a holding minute on the ground has the same cost of a minute of flight, *i.e.* an arriving aircraft cannot be delayed as much as a departing one. On this basis, regulations on the departure flows are currently in operation in several countries. The Central Flow Management Unit of Eurocontrol (the European authority for the air navigation services) assigns a Calculated Take-off time (CTOT) to the departures in order to respect the en-route and airport capacities, and to deal with ATC

sectors closure. Despite this, delay actions in the form of "long vectoring" (*i.e.* the route extension by using radar instructions in terms of headings), holding procedures or speed control are tactically used to sequence inbound traffic. The problem of sequencing the arrivals and the departures remains a tactical problem for the air traffic controllers.

In this paper we assume that for each flight we have as input data the type of aircraft, the wake turbulence category and an associated delay cost per time unit. Depending on the type of movement, *i.e.* if a flight is a departure or an arrival, we have a scheduled time related to a specific simulation day. The delay cost per time unit may be obtained by considering the average number of seats on an aircraft together with its fuel consumption. Arrivals and departures have different delay costs per time unit, in other words the effect of one minute of delay depends on whether the aircraft operates as an arrival or departure. The separations enforced between aircraft movements in this paper are time separation constraints obtained from ICAO regulations. Modeling separations between aircraft movements in a realistic way is more complicated because legislation makes it possible for controllers to reduce the minimum separation (during normal weather conditions) if they have an aircraft in sight. In order to overcome this problem, we consider the case of normal weather conditions without departures from intermediate positions, and we assume that one minute is sufficient for the inbound aircraft to vacate the runway and for the departing aircraft to free the runway.

### *1.2. Literature review*

In the literature on runway scheduling problems, the sequencing of arrivals (*Aircraft Landing Problem*, ALP) and departures (*Aircraft Take-off Problem*, ATP) are often considered as separate problems. Only a few authors explain how to extend their approaches to a combination of arrivals and departures, *i.e.* how to tackle the ASP.

Bennel *et al.* [10] recently presented a complete overview of the techniques used for the landing and take-off scheduling problems. These problems may be further subdivided into static and dynamic problems. In the static case it is assumed that complete information on the set of aircraft is known, while in the dynamic case the solution is revised each time new aircraft are introduced into the system.

Beasley *et al.* [7] proposed an interesting mathematical formulation for the static case, where the static ALP is formulated and solved as a mixed integer zero-one program. Abela *et al.* [1] proposed one of the first Mixed Integer Linear Programming (MILP) formulations of the dynamic ALP, and developed a branch-and-bound algorithm and a genetic algorithm for solving the problem. Furthermore, Beasley *et al.* [8] investigated the dynamic case as an application of the displacement problem. Another interesting

formulation of the ALP was presented by Beasley and Pinol [9], where two objective functions, one linear and one nonlinear, were used, and where a scatter search algorithm and a bionomic algorithm have been developed.

For the ASP with CPS, Balakrishnan and Chandran [6] present dynamic programming approaches for a number of different variants of the objective function, also providing complexity analyses of their algorithms. Further work on dynamic programming for the ASP with CPS has been conducted by Furini *et. al.* [16], who investigated several methods for speeding up the dynamic programming algorithm by using state space reduction techniques.

Various heuristic approaches for solving the ALP have also been proposed. Among metaheuristics, genetic algorithms are the most popular, and one of the first applications was proposed by Stevens [23]. Ciesielski and Scerri [14] also developed a genetic algorithm aimed at solving the ALP in real-time, and experiments were performed on landing data from Sydney airport (during busy days). A genetic algorithm has been also proposed by Cheng *et al.* [13] for the ALP with multiple runways. The effectiveness of memetic algorithms has been illustrated by several researchers in aviation field. An interesting example is the application of a memetic algorithm for multi-objective robustness improvement in airline schedules by Burke *et al.* [12].

Atkin *et al.* [4] investigated real air traffic data concerning London Heathrow airport, and compared three different algorithms for the ATP, namely a tabu search, a simulated annealing and a descent algorithm. Further developments on such algorithms have since been proposed by the same authors in [5] , [2] and [3].

An interesting model that considers both arrivals and departures on the same or different runways was proposed by Ernst *et al.* [15]. Each arriving aircraft has a desired arrival or departure time and the objective function assigns a penalty to an aircraft arriving or departing before or after this time. A branch-and-bound method and a local search heuristic are developed in order to solve this problem. The combination of arrival and departure sequences has also been considered by Persiani and Bagassi [21]. The real case of a busy, single runway airport was considered and a particle swarm optimization algorithm developed in order to integrate arrival and departure flows.

Preliminary research on solving the ASP using a rolling horizon approach has been presented in [17]. A new MILP formulation of the problem has been proposed and used to solve the ASP in a rolling horizon framework, where the length of the horizon was defined by a fixed number of aircraft.

### *1.3. Paper contributions*

In this paper we consider the static case of the problem of sequencing both arrivals and departures on a single runway. The contributions of this paper may be summarised as follows: (1) a fast heuristic that effectively solves small subsequences (chunks) of aircraft is proposed; (2) a number of criteria for partitioning a sequence of aircraft into subsequences of aircraft (chunking rules) are studied; (3) it is shown that by using in an iterative way different chunking rules applied to various sequences of the given aircraft, improved results are obtained in comparison to those obtained by solving the problem without chunking or by using just a single chunking rule applied to the FCFS sequence; (4) it is shown that the proposed algorithm is suitable for use in real-time; (5) the proposed algorithm, although a heuristic, is able to find optimal solutions to real-world instances within a short time frame.

## 2. A rolling horizon approach

In this section we present a rolling horizon approach specifically developed for online applications where the ASP has to be solved for long sequences of aircraft. The main idea of the approach is to consider subsets of a large set of aircraft, called *chunks*, which are small enough to be solved in real time when given as input to the ASP, and then to reconstruct a global solution from the partial solutions of the chunks. The rolling horizon approach represents a valid solution method to the problem of sequencing aircraft for several reasons. Indeed, (i) it is similar to an operative approach; (ii) the computation time is suitable for real time applications; (iii) it is able to improve the FCFS solution significantly. It is also important to note that the rolling horizon approach can be used to solve both the static and the dynamic cases of the ASP, where the dynamic case consists of a dynamic updating of the input data and of the rolling horizon time window.

In the next three subsections we discuss the rolling horizon algorithm, the rules used to decide how to partition a sequence of aircraft into chunks, and an iterative algorithm which updates the sequence of aircraft in each iteration by applying the rolling horizon algorithm.

### *2.1. The rolling horizon algorithm*

The rolling horizon algorithm is given by Algorithm 1. It takes as input an initial sequence $s = (s_1, s_2, \ldots, s_n)$ of $n$ distinct aircraft such that $s_i \in I$ for all $i \in \{1, 2, \ldots, n\}$, and an increasing sequence of $c$ distinct integers $(\gamma_1, \gamma_2, \ldots, \gamma_c)$ called *chunk points*, where $\gamma_c = n$. The chunk points are generated according to a specified *chunking rule* as will be discussed below, and are used to construct $c$ chunks $J_k \subset I$ for $k = 1, \ldots, c$ corresponding to a partition of the $n$ aircraft.

In the rolling horizon algorithm, additional initial boundary constraints are taken into account in order to ensure that no aircraft in any chunk is assigned a time earlier than that of the last aircraft in the previous chunk, plus the necessary minimum separation. Given a chunk $J_k$ and the last aircraft $i^*$ of the previous chunk $(k-1)$, we define an initial time $t^{i^*}_{J_k}$ and a list of separation values $\bar{t}_S = (t_S^{i^* s_{\gamma_{k-1}+1}}, t_S^{i^* s_{\gamma_{k-1}+2}}, \ldots, t_S^{i^* s_{\gamma_k}})$, one for each aircraft $i \in J_k$. Constraints are then imposed to ensure that no aircraft $i \in J_k$ may be assigned a time earlier than $t^{i^*}_{J_k} + t_S^{i^* i}$.

In Steps 4 and 9, the algorithm makes use of a function $\texttt{Solve}(J_k, t^{i^*}_{J_k}, \bar{t}_S)$, which solves the ASP with aircraft in the chunk $J_k$ as input, together with an intial time and a list of separation values as defined above. This function gives as output a sequence $s' = (s'_{\gamma_{k-1}+1}, s'_{\gamma_{k-1}+2}, \ldots, s'_{\gamma_{k-1}+|J_k|})$ (with $\gamma_0 = 0$, $\gamma_{k-1} + |J_k| = \gamma_k$) of the aircraft in $J_k$ and an assigned time $z'_i$ for each aircraft $s'_i$ in $s'$.

The chunk point $\gamma_1$ defines the end of the first chunk $J_1$ in Step 1, and the initial boundary values are set to 0 for the first chunk in Steps 2–3. This chunk is then solved in Step 4. Each subsequent chunk $J_{k+1}$ is then constructed in Step 6 by taking the next list of aircraft as specified by the chunking point $\gamma_{k+1}$. The initial boundary values are set in Steps 7–8 according to the last aircraft in $s'$ (*i.e.* the solution of chunk $J_k$), and the new chunk $J_{k+1}$ is solved in Step 9. When the algorithm terminates, each aircraft $i \in I$ has been assigned a time by the `Solve` function, and this provides a complete solution for the set $I$ which is guaranteed to be feasible due to the boundary separation constraints.

### *2.2. Chunking rules*

In [17] each chunk consists of a fixed number of aircraft. In what follows we present this chunking rule together with two further rules based on the gaps between the scheduled times of consecutive aircraft in the input sequence $s$ and on the distribution of costs around certain potential chunk points in $s$. The objective of a chunking rule is to determine a list of $c$ increasing chunk points $(\gamma_1, \gamma_2, \ldots, \gamma_c)$ that will be used in the rolling horizon algorithm as discussed above.

*Fixed cardinality chunks.* The first chunking rule, $\text{CR}_{\text{fixed}}$, generates chunks according to the chunk points $(\ell, 2\ell, 3\ell, \ldots, \lfloor (n-1)/\ell \rfloor \ell, n)$ for a given integer $\ell < n$. In other words, $\lfloor (n-1)/\ell \rfloor + 1$ chunks are generated each with a cardinality $\ell$, except when $n \pmod{\ell} > 0$ in which case the last chunk has cardinality $n \pmod{\ell}$.

*Time gap chunking.* The second chunking rule, $\text{CR}_{\text{time}}$, is based on a measure that relates to the time gap between two consecutive aircraft in a sequence $s$. This rule attempts to split $s$ at large time gaps in order to create chunks containing aircraft that are scheduled to arrive or depart shortly after one another.

**Algorithm 1:** Rolling horizon algorithm

**Input** : A input set $I$ of aircraft, a sequence $s = (s_1, s_2, \ldots, s_n)$ of the aircraft in $I$, and a list $(\gamma_1, \gamma_2, \ldots, \gamma_c)$ of chunking points as determined by a chunking rule.

**Output**: An assigned runway time $z_i$ for each aircraft $i \in I$.

$J_1 \leftarrow \{s_1, s_2, \ldots, s_{\gamma_1}\}$;

$t_{J_1}^{i^*} = 0$;

**for** $i \in J_1$ **do** $t_S^{i^* i} \leftarrow 0$ ;

$\{(s'_1, s'_2, \ldots, s'_{|J_1|}), (z'_1, z'_2, \ldots, z'_{|J_1|})\} \leftarrow \texttt{Solve}(J_1, t_{J_1}^{i^*}, \bar{t}_S)$;

**for** $k = 1, 2, \ldots, c-1$ **do**

  $J_{k+1} \leftarrow \{s_{\gamma_k+1}, s_{\gamma_k+2}, \ldots, s_{\gamma_{k+1}}\}$;

  $t_{J_{k+1}}^{i^*} \leftarrow z'_{\gamma_k}$;

  **for** $i \in J_{k+1}$ **do** $t_S^{i^* i} \leftarrow t_S^{s'_{\gamma_k} i}$ ;

  $\{(s'_{\gamma_k+1}, s'_{\gamma_k+2}, \ldots, s'_{\gamma_{k+1}}), (z'_{\gamma_k+1}, z'_{\gamma_k+2}, \ldots, z'_{\gamma_{k+1}})\} \leftarrow \texttt{Solve}(J_{k+1}, t_{J_{k+1}}^{i^*}, \bar{t}_S)$;

**end**

The measure used is defined by $\tau_s^{s_i} = t_I^{s_{i+1}} - t_I^{s_i} - t_S^{s_i s_{i+1}}$ for each potential chunk point $s_i$ (with $i = 1, 2, \ldots, n-1$). Hence $\tau_s^{s_i}$ is the maximum delay aircraft $s_i$ may experience before delaying its successor $s_{i+1}$ in $s$. The potential chunking points are then sorted according to the measure $\tau_s^{s_i}$ from the largest one to the smallest one, and are sequentially considered for inclusion into a final list of chunk points. A chunk point is only included if it does not lead to a chunk shorter than a minimum chunk length $\ell_{\min}$ or longer than a maximum chunk length $\ell_{\max}$.

*Weighted gap chunking.* The third chunking rule, $\text{CR}_{\text{weight}}$, is based on a measure that relates to the costs of the aircraft around a potential chunk point. This rule attempts to perform a split at points where the aircraft that will be affected by the split have low weights and can afford to lose some freedom in movement. The measure used is defined by $\omega_s^{s_i} = \max\left(w_{s_i}, w_{s_{i+1}}\right)$ for each potential chunk point $s_i$ (with $i = 1, 2, \ldots, n-1$). In other words, for each pair of consecutive aircraft $s_i$ and $s_{i+1}$ the maximum of two sums are taken, namely the sum of the weight of $s_i$ and the weights of the $p-1$ aircraft preceding it, and the sum of the weight of $s_{i+1}$ and the weights of the $p-1$ aircraft following it. The potential chunking points are now sorted according to the measure $\omega_s^{s_i}$ from the smallest one to the largest one, and are sequentially considered for inclusion into a final list of chunk points. Again, a chunk point is only included if it does not lead to a chunk shorter than a minimum chunk length $\ell_{\min}$ or longer than a maximum chunk length

$\ell_{\max}$.

It is important to note that the last two chunking rules may fail to determine chunks of length between $\ell_{\min}$ and $\ell_{\max}$ in cases where $\ell_{\max} < 2\ell_{\min}$, since during the procedure a chunk of length $\ell$ where $\ell_{\max} < \ell < 2\ell_{\min}$ may be generated, in which case any partition of the chunk will result in at least one chunk smaller than $\ell_{\min}$. In such cases backtracking on the previous chunks may be considered, but in this paper we relax the minimum chunk length restriction and generate two chunks by splitting the infeasible chunk after its $\lfloor \ell/2 \rfloor$-th aircraft. Finally, a *chunking configuration* specifies one of the three chunking rules together with specific values for $\ell$, $\ell_{\min}$ and $\ell_{\max}$, depending on the chunking rule.

It is important to mention that in the literature mainly two other chunking rules are applied in order to derive rolling horizon algorithms. The first one considers an approximation of the future aircraft (*overlapping chunking rules*) and the second one considers instead terminal penalties depending on the end of the current chunk (*terminal penalties chunking rule*). Recently, rolling horizon approaches for aircraft scheduling were studied in [22] where the authors presented an alternative graph formulation and an overlapping chunking rule specifically introduced for the dynamic case where real-time information is provided by the traffic control system. These different chunking rules for rolling horizon approaches have also been applied to different classes of problems, such as machine scheduling problems [19, 24] and others. In [24], for example, a rolling horizon approach is used in which a terminal penalty function is included in the objective function of each chunk, which considers the impact of the local solution on the global objective function. However, our preliminary computational experiments showed that the use of overlapping and terminal penalties chunking rules leads to little or no improvement on the three chunking rules defined at the beginning of this section. Therefore, for the sake of simplicity we only present results for these three chunking rules (namely fixed cardinality chunks, time gap chunking and weighted gap chunking) in the remainder of the paper.

### *2.3. An iterative rolling horizon algorithm*

Two important questions arise when applying the rolling horizon algorithm to a set of aircraft, namely (1) to which sequence of the aircraft the rolling horizon algorithm must be applied (a random sequence, the FCFS sequence, etc.), and (2) which chunking configuration must be used. In order to deal with these questions an iterative procedure was developed which applies the rolling horizon algorithm to different sequences of the aircraft using different chunking configurations.

The *iterative rolling horizon algorithm (IRHA)* is similar to a tabu search procedure, where at each iteration a set of neighbourhood solutions of the current sequence is found using the rolling horizon algo-

rithm, and the objective function values are kept in a tabu list in a first-in-first-out fashion. As input the algorithm takes the FCFS sequence as the initial sequence and an ordered list of chunking configurations $\mathcal{C}$. The neighbourhood of the current sequence is found by applying the rolling horizon algorithm to it using each of the configurations in $\mathcal{C}$ in the order they are given. The first non-tabu neighbour improving the current sequence replaces it in the next iteration, while if no neighbour improves the current sequence, the best non-tabu neighbour is chosen. The IRHA terminates after a specified time or number of iterations, or if all neighbours are tabu.

## 3. Solution of the ASP for small sequences – solving single chunks

In this section we present techniques which can be used to solve individual ASP instances of small length, *i.e.* with a relatively small number of aircraft, in short computing time. This is particularly important for our rolling horizon approach, where a sequence of small ASP instances (chunks) have to be solved. More specifically, we review two MILP formulations from the literature, which can also be used in a heuristic way (by imposing a time limit), and develop a new tabu search heuristic algorithm.

### *3.1. MILP formulations*

The first MILP formulation was presented in [7] and is henceforth referred to as Model A, and the second one was presented in [17] and is referred to as Model B. These two formulations mainly differ in the way the decision variables are defined. In Model A the authors used, for each pair of aircraft, a decision variable which models the relative order between the two aircraft, while in Model B the authors used, for each aircraft, a decision variable which models the position of the aircraft in the sequence. Both formulations, without the addition of any valid inequalities, have a weak continuous (linear programming) relaxation, which attains the straightforward lower bound of value 0. It is therefore not immediately apparent which formulation is computationally more effective in solving the ASP.

*Model A.* The corresponding mathematical formulation is given by (1)–(5), where, for each pair of aircraft $i_1$ and $i_2$, $\delta_{i_1 i_2}$ is a binary variable which attains a value of 1 if aircraft $i_1$ is scheduled before aircraft $i_2$ in the sequence and 0 otherwise.

$$\min \quad \sum_{i \in I} w_i (z_i - t_I^i) \tag{1}$$

$$t_I^i \leq z_i \leq t_I^i + d_i \qquad i \in I \tag{2}$$

$$\delta_{i_1 i_2} + \delta_{i_2 i_1} = 1 \qquad i_1, i_2 \in I, i_1 \neq i_2 \tag{3}$$

$$z_{i_2} \geq z_{i_1} + t_S^{i_1 i_2} - M(1 - \delta_{i_1 i_2}) \quad i_1, i_2 \in I, i_1 \neq i_2 \tag{4}$$

$$\delta_{i_1 i_2} \in \{0, 1\} \qquad\qquad i_1, i_2 \in I, i_1 \neq i_2 \tag{5}$$

The objective function (1) aims at minimizing the total cost due to the weighted delay of the aircraft in the sequence. Constraints (2) ensure that the delay of each aircraft $i$ respects its lower and upper bounds, while constraints (3) ensure that either aircraft $i_1$ arrives/departs before aircraft $i_2$, or vice versa, for any two aircraft $i_1$ and $i_2$. Constraints (4) ensure that the minimum separation constraints between aircraft are satisfied, where $M$ (which denotes a large positive constant) may be replaced by $t_S^{i_1 i_2} + d_{i_1}$ as proposed in [7].

The following inequalities impose an initial condition on the delay by considering the last aircraft $i^*$ assigned before the considered aircraft set:

$$z_i \geq t_I^{i^*} + t_S^{i^* i} \qquad\qquad i \in I. \tag{6}$$

As proposed in [7], constraints (4) may be strengthened by defining the sets

$$S = \{(i_1, i_2) \mid t_I^{i_1} + d_{i_1} < t_I^{i_2},\ i_1, i_2 \in I, i_1 \neq i_2\},$$

$$S' = \{(i_1, i_2) \mid t_I^{i_1} + d_{i_1} + t_S^{i_1 i_2} > t_I^{i_2},\ (i_1, i_2) \in S\}.$$

and

$$T = \{(i_1, i_2) \mid t_I^{i_1} + d_{i_1} \geq t_I^{i_2},\ t_I^{i_2} + d_{i_2} \geq t_I^{i_1},\ i_1, i_2 \in I, i_1 \neq i_2\},$$

The set $S$ of aircraft denotes the set of ordered pairs of aircraft $(i_1, i_2)$ where $i_1$ cannot arrive/depart after $i_2$ without violating the feasible time windows, while $S'$ denotes the set of ordered pairs of aircraft $(i_1, i_2) \in S$ such that $i_2$ may be delayed by $i_1$ due to the separation constraints. The set $T$ denotes the set of ordered pairs of aircraft $(i_1, i_2)$ for which there is the possibility of either aircraft arriving/departing before the other. Constraints (4) may be replaced by constraints (7)–(9) below.

$$\delta_{i_1 i_2} = 1 \qquad\qquad (i_1, i_2) \in S \tag{7}$$

$$z_{i_2} \geq z_{i_1} + t_S^{i_1 i_2} \qquad\qquad (i_1, i_2) \in S' \tag{8}$$

$$z_{i_2} \geq z_{i_1} + t_S^{i_1 i_2} - M(1 - \delta_{i_1 i_2}) \qquad\qquad (i_1, i_2) \in T \tag{9}$$

It is easy to verify that, if $M$ in constraints (4) is large enough (which is the case for the instances that we consider in our computational experiments), then an optimal solution to the continuous relaxation is $\delta_{i_1 i_2} = 0.5$ for all $i_1, i_2 \in I, i_1 \neq i_2$ and $z_i = t_I^i$ for all $i \in I$, which attains the straightforward lower bound of value 0.

*Model B.* The mathematical formulation presented in [17] is given by (10)–(17), where $x_k^i$ is a binary variable which attains a value of 1 if aircraft $i$ is scheduled to arrive/depart in position $k$ in the sequence and 0 otherwise, and where $y_k$ denotes the actual arrival/departure time of the aircraft scheduled to arrive/depart in position $k$.

$$\min \quad \sum_{i\in I} w_i(z_i - t_I^i) \tag{10}$$

$$\sum_{k\in I} x_k^i = 1 \qquad i \in I \tag{11}$$

$$\sum_{i\in I} x_k^i = 1 \qquad k \in I \tag{12}$$

$$z_i \leq t_I^i + d_i \qquad i \in I \tag{13}$$

$$z_i \geq y_k + M(x_k^i - 1) \qquad k \in I, i \in I \tag{14}$$

$$y_k \geq \sum_{i\in I} t_I^i x_k^i \qquad k \in I \tag{15}$$

$$y_k \geq y_{k-1} + t_S^{i_1 i_2}(x_{k-1}^{i_1} + x_k^{i_2} - 1) \qquad k \in I, i_1, i_2 \in I, k \neq 1, i_1 \neq i_2 \tag{16}$$

$$x_k^i \in \{0, 1\} \qquad i \in I, k \in I. \tag{17}$$

As in Model A, the objective function (10) aims at minimizing the total cost due to the weighted delay of the aircraft in the sequence. Constraints (11) and (12) ensure, respectively, that each aircraft is assigned to one position in the sequence and that to each position in the sequence one aircraft is assigned. Constraints (13) ensure that the delay of each aircraft $i$ respects its upper bound, while constraints (14) impose that the arrival/departure time of an aircraft is equal to the the arrival/departure time corresponding to a position if and only if the aircraft is assigned to that specific position. Constraints (15) ensure that no aircraft arrives/departs before its scheduled arrival/departure time, and constraints (16) ensure that the minimum separation constraints between aircraft are satisfied, while together they ensure that $z_i \geq t_I^i$.

The following inequalities impose an initial condition on the delay by considering the last aircraft $i^*$ assigned before the considered aircraft set:

$$y_1 \geq t_S^{i^* i} x_1^i + t_I^{i^*} \qquad i \in I. \tag{18}$$

The formulation may be further strengthened by including the following inequalities:

$$y_k \geq y_{k-1} + \min_{i_1, i_2 \in I, i_1 \neq i_2} t_S^{i_1 i_2} \qquad k \in I, k \neq 1. \tag{19}$$

Constraints (14) are modelled in CPLEX using *indicator constraints* instead of using a big-$M$ value explicitly, and CPLEX removes indicator constraints when calculating the continuous relaxation of a model. It is easy to verify that the continuous relaxation of this model also attains the straightforward lower bound of value 0 when implemented in this way.

### *3.2. Tabu search heuristic*

For the purposes of the tabu search heuristic, a solution is defined by a permutation of the given set of aircraft, and it should be noted that determining the actual arrival/departure times of the aircraft which minimize the total weighted delay, *i.e.* evaluating the solution value, is a trivial problem if the sequence of aircraft is fixed. A neighbour solution of a given feasible solution is defined as the solution obtained by performing either a *swap move*, *i.e.* by exchanging the positions of two aircraft in the given solution, or a *shift move*, *i.e.* by shifting an aircraft to a new position.

During each iteration a tabu-list is updated in a first-in-first-out fashion. If the move performed during the iteration was a swap, then the aircraft that were swapped are recorded, and for a predefined number of iterations (called the tabu *tenure*) these two aircraft may not be swapped again. If the move was a shift, then the resulting subsequence of three consecutive aircraft $(i_1, i_2, i_3)$ is recorded, where $i_2$ is the shifted aircraft. For a predefined number of iterations, when a shift move is applied, $i_2$ may not be shifted to a position inbetween $i_1$ and $i_3$ again. These tabu list structures were chosen for three reasons, namely (1) to avoid the same move being applied again, (2) to allow for a fast comparison between potential moves and moves on the tabu list, and (3) to avoid that the tabu list over-restricts the neighbourhood of a solution.

The neighbour solutions of the current solution in an iteration (initially the FCFS order) is obtained by peforming swaps or shifts not forbidden by the tabu list. We consider two approaches for exploring the neighbourhood. In the first approach, the entire neighbourhood is evaluated and the best non-tabu solution is chosen to replace the current solution in the next iteration. In the second approach, the moves are performed on the aircraft in the order in which they appear in the current sequence, and the first improving solution is immediately accepted and replaces the current solution in the next iteration. The second case simply reduces to the first case in the event that there is no improving solution in the neighbourhood. The search continues until a predefined time-limit or number of iterations is reached, and the best overall solution found is given as output.

## 4. Computational experiments

In this section the results of some computational experiments are presented. We first present a detailed description of the sets of instances we consider, and for what purposes the different sets will be used. Thereafter we present results on using the described heuristics to solve a small chunk in order to determine which heuristic to use in the rolling horizon algorithm. Finally, we present results for the IRHA, both on medium sized instances, for which the optimal solution is known, and on larger instances of aircraft.

All methods were implemented in the C programming language and all computations were performed on an Intel(R) Core(TM)2 Duo CPU clocked at 3.2GHz with 2Gb RAM under the Linux operating system. The MILP models were solved with CPLEX 12.5 [18] with default parameter settings.

### *4.1. Test bed instances*

We consider real air traffic instances from Milano Linate Airport, the largest Italian airport with a single runway for commercial traffic (the second runway is used only by general aviation). The instances correspond to two simulation days, one in August 2011 (from 14:00 to 20:00 UTC) and one in September 2011 (from 14:00 to 21:00 UTC). The former contains 100 scheduled flights (54 arrivals and 46 departures) while the latter contains 121 scheduled flights (68 arrivals and 53 departures). In order to obtain larger and more congested instances, we also merged these two simulation days into one. The 48 flights operating on both days, were not duplicated, leading to a total of 173 scheduled flights (94 arrivals and 79 departures) in the merged set.

The average taxi time varies during the day according to the peak hours, typically from 10 minutes during non congested hours up to 25 minutes in low visibility conditions or during peak hours. We assume an average taxi time of 10 minutes, and that arriving aircraft vacate the runway and proceed directly to their parking stand. In the case of arrivals, the taxi time to the parking does not affect the sequencing problem. Typically the runway services an average of 450 movements per day.

The delay costs per time unit of the aircraft are obtained by considering the number of seats on the aircraft and a medium value of the fuel consumption. In particular, we multiply the hundreds of seats by the ton/hour of fuel, and round the results up to the nearest integer. Table 1 shows the values of the delay costs per time unit used for the most common aircraft.

Table 1: Aircraft costs

| | | | |
|---|---|---|---|
| Airbus A320 | 5 | Fokker F70 | 4 |
| Airbus A319 | 4 | Fokker F100 | 5 |
| Airbus A321 | 6 | Canadian Regional Jet CRJ7 | 4 |
| Mcdonnel Douglas MD80/82 | 6 | Canadian Regional Jet CRJ9 | 5 |
| Boeing B737-600 | 5 | Cessna C550 | 1 |
| Boeing B737-800 | 6 | LearJet L30/50/60 | 2 |
| Boeing B737-500 | 4 | Hawker H25B | 2 |
| British Aerospace BA46 | 4 | Falcon F900 | 2 |

In the following we describe three different sets of instances that were used to obtain the results, all of which are based on the instances of the two simulation days described above.

*Instance Set A.* This set contains twelve (overlapping) instances of 60 aircraft each, where instances 1–5 correspond to the day in August 2011 and instances 6–12 correspond to the day in September 2011. These instances were also considered in [17].

The instances, and also the ICAO separation used between each pair of aircraft, are available online at `www.or.deis.unibo.it/research.html`. For each aircraft, the instances also specify the wake turbulence category, the type of operation (arrival or departure), the Estimated Landing Time and the Estimated Take-off Time (as sum of the Estimated off Block Time and of the average taxi time).

*Instance Set B.* For the purpose of comparing the described heuristics, 500 smaller instances were generated by sampling sequences of consecutive aircraft of lengths ranging between 15 and 24 (50 instances for each sequence length) from the FCFS sequence of the merged set of 173 aircraft. More specifically, the $i$-th (with $i = 1, 2, \ldots, 50$) instance of length $n$ (with $15 \leq n \leq 24$) consisted of aircraft in positions $i, i+1, \ldots, i+n-1$ of the FCFS sequence of the given 173 aircraft.

*Instance Set C.* A similar approach was used for evaluating the performance of the rolling horizon algorithm on larger instances. In this case, the instances consist of longer sequences of aircraft having lengths $70, 90, 110, 130$, and $150$. For each sequence length, 3 instances (each consisting of a sequence of consecutive aircraft) were sampled from the FCFS sequence of the merged set of 173 aircraft, so as to avoid large overlaps between instances. More specifically, the $i$-th (with $i = 1, 2, 3$) instance of length $n \in \{70, 90, 110, 130, 150\}$ consisted of $n$ consecutive aircraft starting from position $(i-1)\lfloor(170-n)/2\rfloor$ of the FCFS sequence of the given 173 aircraft.

### *4.2. Identification of the best way of solving a chunk*

In order to determine which is the most appropriate way for finding a solution for a small chunk, we performed computational experiments on Instance Set B using the tabu search algorithm and the MILP models.

For the tabu search algorithm, results on the peformance of various configurations of the parameters applied to the instances of Instance Set B are presented in Table 2. The table compares the percentage improvement on the FCFS solution when using either swap moves, shift moves or both (in which case swap moves are considered first), either exploring the entire neighbourhood or accepting the first improvement, and either using a tabu tenure of value 20, 50 or 100. For each instance, a time limit of 15 seconds was imposed as a stopping criterion.

Table 2: Comparison of different configurations of the tabu search algorithm

| | *Shift moves* | | | | | | *Swap moves* | | | | | | *Shift & swap moves* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Entire neighbourhood* | | | *First improvement* | | | *Entire neighbourhood* | | | *First improvement* | | | *Entire neighbourhood* | | | *First improvement* | | |
| *n* | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 |
| *15* | 38.52 | 38.53 | 38.55 | 37.72 | **38.60** | **38.60** | **38.60** | 38.56 | 38.09 | **38.60** | 38.49 | 36.79 | **38.60** | **38.60** | **38.60** | **38.60** | **38.60** | **38.60** |
| *16* | 38.22 | 38.29 | 38.29 | 37.49 | 38.32 | **38.35** | **38.35** | 38.33 | 37.79 | **38.35** | 38.27 | 36.60 | **38.35** | **38.35** | **38.35** | **38.35** | **38.35** | **38.35** |
| *17* | 37.97 | 37.98 | 38.06 | 36.50 | 37.98 | **38.12** | **38.12** | 38.10 | 37.54 | **38.12** | 38.02 | 36.37 | **38.12** | **38.12** | **38.12** | **38.12** | **38.12** | **38.12** |
| *18* | 37.64 | 37.65 | 37.81 | 35.17 | 37.74 | **37.89** | **37.89** | 37.83 | 37.30 | **37.89** | 37.80 | 36.26 | **37.89** | **37.89** | **37.89** | **37.89** | **37.89** | **37.89** |
| *19* | 37.34 | 37.35 | **37.70** | 35.36 | 37.16 | **37.70** | **37.70** | 37.53 | 37.13 | 37.68 | 37.55 | 36.14 | **37.70** | **37.70** | **37.70** | **37.70** | **37.70** | **37.70** |
| *20* | 37.19 | 37.20 | 37.49 | 35.25 | 37.09 | 37.62 | **37.63** | 37.57 | 36.87 | 37.61 | 37.49 | 36.06 | 37.62 | 37.62 | **37.63** | **37.63** | **37.63** | **37.63** |
| *21* | 37.21 | 37.20 | 37.41 | 35.14 | 37.15 | 37.68 | 37.68 | 37.61 | 36.92 | 37.69 | 37.47 | 35.99 | **37.71** | **37.71** | **37.71** | 37.70 | 37.70 | **37.71** |
| *22* | 37.44 | 37.51 | 37.63 | 35.06 | 37.54 | 37.94 | **37.96** | 37.85 | 37.02 | 37.94 | 37.69 | 36.18 | 37.93 | 37.93 | **37.96** | 37.93 | **37.96** | **37.96** |
| *23* | 37.90 | 37.96 | 38.06 | 35.59 | 37.90 | 38.32 | **38.35** | 38.24 | 37.50 | **38.35** | 38.04 | 36.60 | **38.35** | **38.35** | **38.35** | **38.35** | **38.35** | **38.35** |
| *24* | 38.37 | 38.38 | 38.47 | 36.40 | 38.25 | **38.80** | **38.80** | 38.67 | 37.93 | **38.80** | 38.56 | 37.18 | 38.78 | 38.78 | **38.80** | 38.78 | 38.78 | **38.80** |
| Avg | 37.78 | 37.80 | 37.95 | 35.97 | 37.77 | 38.10 | **38.11** | 38.03 | 37.41 | 38.10 | 37.94 | 36.42 | 38.10 | 38.10 | **38.11** | **38.11** | **38.11** | **38.11** |

From the table it may be seen that when using shift moves the average improvement consistently increases as the tabu tenure value is increased, while when using swap moves the average improvement consistently decreases as the tabu tenure value is increased. Furthermore, when using shift moves with a tabu tenure of value 100 the average improvement is larger if the first improvement is accepted, while when using swap moves and a tabu tenure of value 20 the average improvement is generally larger if the entire neighbourhood is considered compared to accepting the first improvement. However, when using both moves there is very little variation in the results for different tabu tenure values, and also very little variation when considering the entire neighbourhood compared to accepting the first improvement. Overall, the configuration that finds the best average improvement uses both moves, a tabu tenure of value 100 and accepts the first improvement.

Table 3 gives a comparison of the best tabu search configuration and the two MILP models, all using a time limit of 15 seconds for each instance. The table shows, for each value of $n$ and for each method, the average percentage improvement on the FCFS solution, and the average percentage gap between the solution value found by the method and the best lower bound found by CPLEX in solving Model A within the time limit (this lower bound was always better than the lower bound found by Model B within the time limit). For Model B the time limit was reached in all instances, while, for Model A, Table 3 gives, for each value of $n$, the percentage of instances for which the time limit was reached as well as the average computation time in seconds.

The results clearly show that Model A outperforms Model B with respect to both the required computation time and the quality of the solution values, while the tabu search heuristic obtains higher average improvements compared to Model A. Motivated by these results, the tabu search algorithm was selected as the heuristic used to solve small chunks in the rolling horizon algorithm, using the best configuration of the parameters according to Table 2.

### *4.3. Computational comparison between the chunking rules*

As discussed in Section 2.2, the chunking procedure for the rolling horizon algorithm takes as input the size of the fixed chunks in case of chunking rule $\text{CR}_{\text{fixed}}$ and the minimum and maximum chunk sizes in case of chunking rules $\text{CR}_{\text{time}}$ and $\text{CR}_{\text{weight}}$. For $\text{CR}_{\text{fixed}}$ chunk sizes 15 and 20 were chosen, while for $\text{CR}_{\text{time}}$ and $\text{CR}_{\text{weight}}$ the minimum and maximum chunk size pairs $(10, 20)$ and $(15, 25)$ were chosen. Table 4 shows some features of the chunks generated by the latter two chunking rules for Instance Set C. For each chunking rule the table contains three columns, reporting the smallest chunk length generated on average for an instance ("*Min*"), the average chunk length over all instances ("*Avg*"), and the largest chunk

Table 3: Comparison of heuristics for solving small chunks

| | *Average improvement (%)* | | | *Average gap (%)* | | | *Model A computation time* | |
|---|---|---|---|---|---|---|---|---|
| *n* | *Model A* | *Model B* | *TS* | *Model A* | *Model B* | *TS* | *% Time limits* | *Average time (s)* |
| *15* | **38.60** | 0.81 | **38.60** | **4.99** | 41.30 | **4.99** | 20.00 | 4.22 |
| *16* | 38.34 | 0.94 | **38.35** | 7.39 | 42.43 | **7.37** | 28.00 | 5.41 |
| *17* | 37.97 | 0.00 | **38.12** | 9.57 | 44.00 | **9.44** | 30.00 | 6.08 |
| *18* | 37.67 | 0.00 | **37.89** | 12.11 | 45.18 | **11.92** | 38.00 | 7.38 |
| *19* | 37.49 | 0.00 | **37.70** | 14.35 | 46.28 | **14.17** | 46.00 | 8.64 |
| *20* | 37.30 | 0.00 | **37.63** | 17.55 | 48.06 | **17.30** | 52.00 | 9.57 |
| *21* | 37.43 | 0.00 | **37.71** | 21.43 | 50.39 | **21.25** | 60.00 | 10.82 |
| *22* | 37.54 | 0.00 | **37.96** | 25.07 | 52.68 | **24.76** | 68.00 | 12.00 |
| *23* | 37.70 | 0.00 | **38.35** | 28.23 | 54.78 | **27.80** | 74.00 | 12.98 |
| *24* | 38.12 | 0.00 | **38.80** | 33.83 | 58.40 | **33.36** | 88.00 | 13.87 |
| Avg | 37.82 | 0.17 | **38.11** | 17.45 | 48.35 | **17.24** | 50.40 | 9.10 |

length generated on average for an instance ("*Max*"), where $n$ represents the total number of aircraft.

For each chunking rule, the tabu search algorithm was used to solve the ASP for the different chunks. For each instance and for each chunking rule, a time limit was imposed, which was in turn divided among the different chunks proportional to their lengths.

In Table 5 the three chunking rules, each with the two configurations of the parameters mentioned above and a time limit of 10 minutes, are compared on Instance Set C with respect to the average percentage improvement on the FCFS solution value (of the entire sequence), and the average percentage gap between the solution obtained by each chunking rule and the best lower bound obtained by using Model A with a 5 hour time limit. In the next subsection the results of the IRHA will be compared with the results of this table.

We also compare the same configurations of the chunking rules with respect to Instance Set A and a time limit of 120 seconds. For Instance Set A the optimal solutions were obtained using Model A. In Table 6 the percentage gap between the value of the solution found by a chunking configuration and the corresponding optimal solution value is given. It may be seen from the table that in some cases the optimal solutions are found by applying a certain chunking configuration, but that there is no configuration that consistently finds the best solution. On average, the best results are obtained by the chunking rule $\mathrm{CR}_{\mathrm{time}}$ with minimum and maximum chunk size $10$ and $20$, respectively.

Table 4: Characteristics of the chunks generated by the chunking rules $\mathrm{CR}_{\mathrm{time}}$ and $\mathrm{CR}_{\mathrm{weight}}$

| | $\mathrm{CR}_{\mathrm{time}}$ | | | | | | $CR_{\mathrm{weight}}$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $(10, 20)$ | | | $(15, 25)$ | | | $(10, 20)$ | | | $(15, 25)$ | | |
| *n* | *Min* | *Avg* | *Max* | *Min* | *Avg* | *Max* | *Min* | *Avg* | *Max* | *Min* | *Avg* | *Max* |
| *70* | 10 | 14.00 | 17 | 15 | 17.50 | 22 | 11 | 14.00 | 18 | 15 | 17.50 | 20 |
| *70* | 10 | 14.00 | 19 | 13 | 14.00 | 15 | 10 | 11.67 | 16 | 15 | 17.50 | 21 |
| *70* | 10 | 14.00 | 19 | 15 | 17.50 | 21 | 10 | 14.00 | 17 | 15 | 17.50 | 21 |
| *90* | 11 | 15.00 | 19 | 15 | 18.00 | 25 | 10 | 12.86 | 17 | 13 | 15.00 | 20 |
| *90* | 10 | 12.86 | 19 | 15 | 18.00 | 20 | 10 | 11.25 | 16 | 13 | 18.00 | 22 |
| *90* | 12 | 15.00 | 19 | 13 | 15.00 | 18 | 10 | 15.00 | 19 | 14 | 15.00 | 17 |
| *110* | 13 | 15.71 | 19 | 15 | 15.71 | 17 | 10 | 12.22 | 17 | 13 | 15.71 | 22 |
| *110* | 12 | 15.71 | 19 | 13 | 15.71 | 19 | 10 | 12.22 | 19 | 13 | 15.71 | 20 |
| *110* | 12 | 15.71 | 19 | 13 | 15.71 | 19 | 10 | 15.71 | 20 | 14 | 15.71 | 20 |
| *130* | 11 | 14.44 | 19 | 15 | 16.25 | 20 | 10 | 11.82 | 17 | 13 | 16.25 | 22 |
| *130* | 11 | 14.44 | 19 | 13 | 16.25 | 24 | 10 | 13.00 | 19 | 13 | 16.25 | 25 |
| *130* | 10 | 14.44 | 19 | 13 | 16.25 | 20 | 10 | 13.00 | 19 | 13 | 16.25 | 22 |
| *150* | 12 | 15.00 | 19 | 13 | 15.00 | 17 | 10 | 12.50 | 19 | 13 | 16.67 | 25 |
| *150* | 10 | 13.64 | 19 | 13 | 16.67 | 24 | 10 | 12.50 | 19 | 13 | 16.67 | 25 |
| *150* | 11 | 15.00 | 19 | 13 | 16.67 | 24 | 10 | 13.64 | 19 | 13 | 15.00 | 20 |
| Avg | 11.00 | 14.60 | 18.87 | 13.80 | 16.28 | 20.33 | 10.07 | 13.03 | 18.07 | 13.53 | 16.32 | 21.47 |

Table 5: Comparing parameters for the three chunking rules on Instance Set C

| | *Average improvement (%)* | | | | | | *Average gap (%)* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $CR_{\text{fixed}}$ | | $CR_{\text{time}}$ | | $CR_{\text{weight}}$ | | $CR_{\text{fixed}}$ | | $CR_{\text{time}}$ | | $CR_{\text{weight}}$ | |
| *n* | 15 | 20 | $(10,20)$ | $(15,25)$ | $(10,20)$ | $(15,25)$ | 15 | 20 | $(10,20)$ | $(15,25)$ | $(10,20)$ | $(15,25)$ |
| *70* | 40.03 | 34.71 | 39.09 | 39.69 | 40.21 | **43.47** | 62.61 | 65.66 | 63.19 | 62.82 | 62.50 | **60.33** |
| *70* | 20.62 | 17.53 | 21.66 | 21.48 | 19.32 | **27.29** | 92.15 | 92.45 | 92.05 | 92.07 | 92.28 | **91.44** |
| *70* | 22.64 | 27.00 | 35.64 | **40.57** | 30.79 | 30.79 | 68.79 | 66.93 | 62.49 | **59.38** | 65.12 | 65.12 |
| *90* | 32.14 | 32.23 | **32.77** | 29.16 | 32.14 | 32.14 | 79.77 | 79.74 | **79.58** | 80.62 | 79.77 | 79.77 |
| *90* | 19.46 | **27.87** | 18.03 | 18.67 | 16.84 | 24.47 | 92.17 | **91.25** | 92.30 | 92.24 | 92.41 | 91.65 |
| *90* | 23.41 | 33.99 | 30.30 | 28.80 | **39.50** | 26.33 | 88.11 | 86.21 | 86.94 | 87.21 | **84.95** | 87.64 |
| *110* | 23.84 | **32.19** | 27.05 | 25.44 | 25.91 | 30.04 | 88.56 | **87.15** | 88.05 | 88.31 | 88.24 | 87.54 |
| *110* | 19.90 | 19.10 | 20.64 | 19.56 | 19.46 | **23.00** | 93.29 | 93.35 | 93.22 | 93.32 | 93.32 | **93.02** |
| *110* | 19.23 | **22.65** | 19.33 | 18.98 | 21.13 | 21.13 | 94.85 | **94.62** | 94.85 | 94.87 | 94.73 | 94.73 |
| *130* | 21.35 | **28.22** | 20.69 | 20.26 | 19.43 | 25.46 | 91.78 | **90.99** | 91.85 | 91.89 | 91.98 | 91.33 |
| *130* | 18.05 | **23.71** | 19.97 | 19.77 | 19.04 | 23.60 | 93.69 | **93.22** | 93.54 | 93.55 | 93.61 | 93.23 |
| *130* | 17.58 | **22.62** | 19.34 | 19.00 | 17.86 | 21.33 | 94.36 | **93.99** | 94.24 | 94.26 | 94.34 | 94.09 |
| *150* | 20.71 | 23.98 | 20.50 | 19.61 | 19.32 | **24.14** | 92.96 | 92.65 | 92.98 | 93.05 | 93.08 | **92.64** |
| *150* | 17.09 | 20.23 | 18.08 | 18.96 | 18.14 | **22.35** | 94.24 | 94.02 | 94.17 | 94.11 | 94.17 | **93.85** |
| *150* | 16.87 | **22.59** | 19.90 | 19.59 | 18.64 | 21.26 | 94.43 | **94.02** | 94.22 | 94.25 | 94.31 | 94.12 |
| Avg | 22.20 | 25.91 | 24.20 | 23.97 | 23.85 | **26.45** | 88.12 | 87.75 | 87.58 | 87.46 | 87.65 | **87.37** |

Table 6: Comparing the three chunking rules with respect to instances with known optimal solutions

| | *% gap w.r.t. optimal solution value* | | | | | |
|---|---|---|---|---|---|---|
| | $CR_{\text{fixed}}$ | | $CR_{\text{time}}$ | | $CR_{\text{weight}}$ | |
| *Instance* | 15 | 20 | $(10,20)$ | $(15,25)$ | $(10,20)$ | $(15,25)$ |
| *FPT1* | 0.75 | 13.68 | **0.00** | 2.93 | 9.56 | 3.64 |
| *FPT2* | 11.48 | 5.79 | 7.57 | **0.00** | 10.40 | 0.68 |
| *FPT3* | 15.28 | 12.97 | **0.00** | 9.89 | 11.15 | 9.89 |
| *FPT4* | 2.90 | 6.94 | **0.00** | **0.00** | 2.90 | **0.00** |
| *FPT5* | 25.89 | **3.11** | 7.09 | 28.86 | 22.19 | 17.55 |
| *FPT6* | 1.76 | 10.70 | 1.76 | **0.00** | **0.00** | 1.76 |
| *FPT7* | 1.98 | **0.00** | 1.49 | **0.00** | 7.48 | **0.00** |
| *FPT8* | 8.24 | 14.36 | 1.76 | **0.00** | 1.76 | 11.17 |
| *FPT9* | 12.44 | **0.00** | 1.61 | 1.61 | 1.61 | **0.00** |
| *FPT10* | 2.76 | 10.97 | **1.40** | **1.40** | **1.40** | 4.09 |
| *FPT11* | 8.40 | **0.00** | 1.29 | 2.97 | 1.29 | 12.60 |
| *FPT12* | **0.00** | 2.82 | **0.00** | **0.00** | **0.00** | **0.00** |
| Avg | 7.66 | 6.78 | 2.00 | 3.97 | 5.81 | 5.11 |

Table 7: Results of the IRHA

| $n$ | *Average improvement (%)* | *Average gap (%)* |
|---|---|---|
| *70* | 48.11 | 56.79 |
| *70* | 54.65 | 86.27 |
| *70* | 43.14 | 57.54 |
| *90* | 44.75 | 75.15 |
| *90* | 55.38 | 85.86 |
| *90* | 57.98 | 78.33 |
| *110* | 54.48 | 80.85 |
| *110* | 53.80 | 88.36 |
| *110* | 53.72 | 91.01 |
| *130* | 59.78 | 83.93 |
| *130* | 53.24 | 88.94 |
| *130* | 53.07 | 90.09 |
| *150* | 57.93 | 86.72 |
| *150* | 51.88 | 90.08 |
| *150* | 48.40 | 91.03 |
| Avg | 52.69 | 82.06 |

### *4.4. Results on the IRHA*

For the IHRA the set $\mathcal{C}$ of chunking configurations used are the six configurations compare in Table 5 in the order in which they appear (from left to right) in this table. In Table 7 we show the performance of the IRHA by using Instance Set C with a time limit of 10 minutes. The values reported in Table 7 have the same meanings of those reported in Table 5. Moreover, a time limit of 40 seconds (divided among the different chunks proportionally to their lengths) was imposed for the rolling horizon algorithm to solve a sequence using a chunking configuration in the IRHA. This time limit was chosen so that each iteration of the algorithm (which evaluates at most six neighbour solutions) does not exceed a running time of 4 minutes, allowing for at least two iterations in the worst case.

Comparing the results of Table 7 with the results of Table 5 (for which a time limit of 10 minutes was also imposed), it follows that combining the different chunking rules and applying them to various sequences, instead of using only one chunking configuration applied to the FCFS sequence, has the potential of producing significantly improved results.

It should be noted that when the tabu search algorithm is applied to the entire sequence of aircraft (*i.e.* with no chunking), there are no restrictions imposed on the positions of the aircraft, whereas in the case of the IRHA position restrictions are implicitly imposed by chunking. Imposing no restrictions on the positions of the aircraft will naturally allow the tabu search algorithm to find good solutions in terms of

Table 8: Results of the IRHA with respect to instances with known optimal solutions

| *Instance* | *Model A* | *IRHA* |
|---|---|---|
| *FPT1* | 2005 | 120 |
| *FPT2* | 88701 | 120 |
| *FPT3* | 5394 | 120 |
| *FPT4* | 8072 | 120 |
| *FPT5* | 1543 | 120 |
| *FPT6* | 31 | 120 |
| *FPT7* | 411 | 120 |
| *FPT8* | 142 | 120 |
| *FPT9* | 34 | 120 |
| *FPT10* | 302 | 120 |
| *FPT11* | 1445 | 120 |
| *FPT12* | 9 | 40 |

weighted delay, but on average the position shift values of aircraft will be higher than in the case of the IRHA. The solutions may therefore be not suitable for use in practice due to the lack of fairness in the schedules, which is why the IRHA would be preferable.

Table 8 shows for Instance Set A the computation time (in seconds) required by Model A to find the optimal solution, together with the computation time (in seconds) required by the IRHA to find the optimal solution (once it has been identified by Model A). The optimal solution values coincide since for these instances it was not necessary to shift the aircraft much, as is the case in many real-world applications. It can furhter be seen that the IRHA is able to find optimal solutions within a realistic time frame (note that due to the 40 second time limit mentioned above, all times for the IRHA are multiples of 40). Furthermore, it can be pointed out that, within the same time frame of 120 seconds, there was no chunking configuration that consistently found the optimal solution, while combining the configurations the IRHA was able to find the optimal solution in all cases.

When short computation times are required, it is therefore beneficial to solve the ASP for small chunks and combine the results instead of solving the ASP for the entire sequence of aircraft. The results show that this is a promising method for solving the ASP in real-time.

## 5. Conclusions and further development

The runway is a critical bottleneck for the air transportation system. Aircraft scheduling on the runway is a challenging problem that aims to reduce delays and the workload of the controllers, and to increase runway capacity. In this paper we consider the problem of sequencing both arrivals and departures without

imposing position shifting constraints, review two MILP models from the literature and propose a tabu search heuristic for solving the problem. A rolling horizon approach, which partitions the sequence of aircraft into so-called chunks and solves ASP individually for each of these chunks, has been proposed and tested on real world air traffic data from the Milano Linate Airport. Three different methods of generating chunks were considered, and the results show that combining these chunking rules leads to an improvement with respect to the case where no chunking is performed. The final objective of this paper consisted of the design of a supporting tool for air traffic controllers that is able to suggest an optimised runway utilization sequence. Further developments may focus on other methods for generating chunks, while additional exact methods for solving the individual chunks may also be considered, such as, for example, dynamic programming, provided that constrained shift positioning is imposed and the maximum position shift value is small.

## Acknowledgments

The authors would like to thank the anonymous referees, whose comments greatly improved the presentation of the paper.

## References

[1] Abela, J., Abramson, D., Krishnamoorthy, M., De Silva, A., Mills, G.: Computing optimal schedules for landing aircraft. Proceeding 12th National ASOR Conference, Adelaide, Australia. 71–90 (1993)

[2] Atkin, J.A.D., Burke, E.K., Greenwood, J.S., Reeson, D.: Hybrid metaheuristic to aid runway scheduling at London Heathrow airport. Transportation Science, 41, 90-106, (2007)

[3] Atkin, J.A.D., Burke, E.K., Greenwood, J.S., Reeson, D.: On-line decision support for take-off runway scheduling under uncertain taxi time at London Heathrow airport. Journal of Scheduling, 11, 323-346, (2007)

[4] Atkin, J.A.D., Burke, E.K., Greenwood, J.S., Reeson, D.: A metaheuristic approach to aircraft departure scheduling at London Heathrow airport. Computer-aided Systems in Public Transport, Lecture Notes in Economics and Mathematical Systems, 600, 235–252 (2008)

[5] Atkin, J.A.D., Burke, E.K., Greenwood, J.S., Reeson, D.: An examination of take-off scheduling constraints at London Heathrow airport. Public Transport, 1(3), 169–187, (2009)

[6] Balakrishnan, H. and Chandran, B.G., Algorithms for Scheduling Runway Operations Under Constrained Position Shifting. Operations Research, 58(6), 1650–1665. (2010)

[7] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., Abramson, D.: Scheduling Aircraft landing-the static case. Transportation Science, 34, 180–197 (2000)

[8] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., Abramson, D.: Displacement problem and dynamically scheduling aircraft landings. Journal of the Operational Research Society, 55, 54–64 (2004)

[9] Beasley, J.E., Pinol, H.: Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem. European Journal of Operational Research, 127(2) 439-462, (2006)

[10] Bennel, J.A., Mesgarpour, M., Potts, C.N.: Airport Runway Scheduling. 4OR, 9, 115–138 (2011)

[11] Brentnall, A.R, : Aircraft Arrivals Management. Dissertation Thesis, University of Southhampton. (2006)

[12] Burke E.K., De Causmaecker P., De Maere G., Mulder J., Paelinck M., Vanden Berghe G.: A multi-objective approach for robust airline scheduling. Computers & Operations Research, 37(5), 822–832, (2010)

[13] Cheng, V.H.L., Crawford, L.S., Menon, P.K.: Air traffic control using genetic search techniques. IEEE International Conference on Control Applications. (1999)

[14] Ciesielski, V., Scerri, P.: Real Time Genetic Scheduling of Aircraft Landing Times. Proceeding of the 1998 IEEE International Conference on Evolutionary Computation (ICEC98), IEEE, New York, USA, 360-364, (1998)

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

[16] Furini, F., Kidd, M.P., Persiani, C.A., Toth, P.: State space reduced dynamic programming for the aircraft sequencing problem with constrained position shifting. To appear in Lecture Notes in Computer Science.

[17] Furini, F., Persiani, C.A., Toth, P.: Aircraft sequencing problems via a rolling horizon algorithm. Lecture Notes in Computer Science, 7422, 273–284, (2012)

[18] IBM ILOG, CPLEX Optimizer 12: CPLEX www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[19] Kimms, A., Rolling Planning Horizon. Multi-Level Lot Sizing and Scheduling, Production and Logistics, 239–245, (1997)

[20] Luenberger, R.A.: A travelling-salesman-based approach to aircraft scheduling in the terminal area. Technical Report, NASA, Technical Memorandum 100062. (1998)

[21] Persiani, C.A., Bagassi, S.: Integration of AMAN and DMAN using Particle Swarm Optimization. Electronic Proceedings of the 3rd CEAS AirSpace Conference. (2011)

[22] Samà, M., D'Ariano, A., and Pacciarelli, D.: Rolling Horizon Approach for Aircraft Scheduling in the Terminal Control Area of Busy Airports. Procedia - Social and Behavioral Sciences, 80, 531–552. (2013)

[23] Stevens, G.: An approach to scheduling aircraft landing times using genetic algorithms. Dissertation Thesis, Department of Computer Science, RMIT University, Melbourne, Australia. (1995)

[24] Wang, B., Xi, Y. and Gu, H.: Terminal penalty rolling scheduling based on an initial schedule for single-machine scheduling problem. Computers & Operations Research, 32, 11, 3059–3072. (2005)