Submitted to *Transportation Science*
manuscript (Please, provide the mansucript number!)

Pruning Rules for Optimal Runway Sequencing

Geert De Maere, Jason A.D. Atkin
ASAP research group, School of Computer Science, University of Nottingham, NG81BB Nottingham, UK
{gdm,jaa}@cs.nott.ac.uk

Edmund K. Burke
Queen Mary University of London, Office of the Principal, Queens Building, Mile End Road, London, E14NS, UK
vp-se@qmul.ac.uk

This paper investigates runway sequencing for real world scenarios at one of the world's busiest airports, London Heathrow. Several pruning principles are introduced that enable significant reductions of the problem's average complexity, without compromising the optimality of the resulting sequences, nor compromising the modelling of important real world constraints and objectives. The pruning principles are generic and can be applied in a variety of heuristic, meta-heuristic or exact algorithms. They could also be applied to different runway configurations, as well as to different variants of the machine scheduling problem with sequence dependent setup times, the generic variant of the runway sequencing problem in this paper. They have been integrated into a dynamic program for runway sequencing, which has been shown to be able to generate optimal sequences for large scale problems at an extremely low computational cost, whilst considering complex non-linear and non-convex objective functions that offer significant flexibility to model real world preferences and real world constraints. The results shown here counter the proliferation of papers that claim that runway sequencing problems are too complex to solve exactly and therefore attempt to solve them heuristically.

Key words: Dynamic programming, Runway Sequencing, Machine Scheduling, Sequence dependent setup times
History:

1. Introduction

The recent and predicted future growth in air transport (Eurocontrol 2013) has already increased the pressure on airport resources around the world, and will continue to do so. This is especially true in the case of runways at highly congested airports that already operate at or close to their maximum capacity. Runway capacity often limits the overall airport capacity, thus the efficient use of this scarce resource is particularly important; failure to do so can significantly increase delays, aircraft emissions and costs for airlines. Adding extra runway capacity (i.e. new runways) is expensive, requires long term planning and may not be possible at many airports due to space restrictions. However, improved use of existing runways may be achieved by intelligently scheduling runway operations by re-sequencing aircraft. This is complex to achieve and requires highly sophisticated algorithmic approaches to be embedded within complex decision support systems to assist runway operators. Such algorithms must be fast enough to allow their use in highly dynamic environments.

Since different separations have to be maintained between aircraft of different types (see Section 2), the order in which aircraft use the runway will affect the overall runway throughput and the delays for each aircraft. The problem of determining this order is called the runway sequencing problem. It aims at finding a feasible sequence that meets all constraints and has a satisfactory or optimal value for some given objective function(s). Many constraints apply to the runway sequencing problem, such as ensuring that safe separations are always maintained between aircraft, ensuring that aircraft are not scheduled to use the runway before they can get there, and meeting any landing/take-off deadlines which may apply to aircraft. There will usually also be a number of conflicting objectives (Atkin et al. 2010, Atkin 2013, Bennell et al. 2011), such as to maximise the runway utilisation, to reduce the average delay per aircraft, and to ensure some level of fairness between the delays for different aircraft. Full details of the problem are provided in Section 2.

If aircraft are considered as jobs and the runway as a machine, the underlying runway sequencing problem is a variant of the machine scheduling problem with sequence dependent setup times, which is known to be NP-hard (Pinedo 2002). For this reason, a fast optimal algorithm for the general formulation is unlikely to be attainable. However, the characteristics of the runway sequencing problem result in sub-structures within the data and separation rules which can be leveraged to improve the tractability of real world instances. For example, the separations which are required between aircraft are not arbitrary, but follow specified rules depending upon the aircrafts' weight classes, speed groups and departure routes. This results in a separation table which is highly structured, the structure of which can be utilised to simplify the problem. We note that, in the case of departures, downstream constraints, e.g., to ensure in-flight separations or control congestion for downstream sectors, are usually applied on the runway. The departure sequencing problem therefore has more, and more complex, constraints than the arrival sequencing problem, for which only the weight classes of the aircraft influence the separations. I.e. separation rules for departures usually do not have such simple structures in them that can be exploited for simplifying the problem. For this reason, the focus of this paper is upon the departure sequencing problems in complex real world environments. Experiments have shown that the approach is also applicable for arrival problems, and that these are actually much easier for it to solve, as will be observed in Section 4.2.

A number of exact approaches for the runway sequencing problem have been introduced previously, and are discussed in more detail below. Heuristic approaches were introduced by, for example, Atkin et al. (2007) and Bianco et al. (1999). We refer the reader to Bennell et al. (2011) for an extensive survey of previous approaches.

Psaraftis (1980) utilised the characteristics of the problem to design an approach which grouped identical aircraft into a number of queues, one per aircraft type, and exploits the fact that a known precedence order exists within the queues in terms of total processing cost. The proposed dynamic program to solve the problem of interleaving queues is polynomial as a function of the number of aircraft n and exponential as a function of the number of aircraft types N ($O(N^2(n+1)^N)$. Such an approach is practical for arrival sequencing, where there may be up to six or seven queues (N), but is impractical for take-off sequencing or mixed mode sequencing (simultaneous arrivals and departures) since many more queues are required in these cases (due to the more complex separations, up to 33 queues for the problem instances considered in this paper). Psaraftis further enhanced his approach by utilising constrained position shifts, introduced by Dear (1976). Constrained position shifts restrict an aircraft's maximum positional shift relative to its position in the initial sequence, usually in first come first served order. This not only reduces the number of aircraft which have to be considered for each position in the sequence, but also enforces equity by preventing individual aircraft from being advanced or delayed disproportionally.

Constrained position shifts were also applied in the dynamic program introduced by Balakrishnan and Chandran (2010). Their approach has a complexity that is polynomial as a function of the number of aircraft n and exponential as a function of the constrained position shift k ($O(n(2k+1)^{(2k+2)})$). The authors also presented an extension of the approach to allow the optimisation of more complex objective functions, albeit at an increased computational complexity.

Whilst constrained position shifts can be effective in many cases of arrival sequencing, in mixed mode operations, delays may differ widely between arrivals and departures, thus overall maximum position shift constraints are impractical. Even within departure operations alone, the differing delays which accumulate across different departure routes, and the requirements to meet time window constraints can require large positional shifts (i.e. large values for k) for the good runway sequences, thereby challenging the tractability and practicality of approaches based on them. We refer the reader to Atkin et al. (2007) and Atkin et al. (2010) for a more detailed discussion of why large positional delays are sometimes beneficial rather than harmful.

If the objectives can be (at least piecewise) linearised, the problem can potentially be solved using a MILP (Mixed Integer Linear Programming) solver such as CPLEX. Beasley et al. (2000) and Ernst et al. (1999) applied such an approach for the arrival scheduling problem with hard landing time windows. Beasley et al. (2000) introduced a mixed integer 0-1 formulation for the static, mixed or segregated, single or multiple runway sequencing problem. The approach exploits the presence of disjoint intervals due to relatively narrow hard time windows for arrivals (caused by speed and fuel limitations), applying a similar sort of simplification as for constrained position shifts, but utilising landing time rather than landing position. The approach allows the modelling of precedence constraints, complex separation matrices and complex piecewise linear and non-linear cost functions through time discretization and linearisation. Additional constraints are added to strengthen the formulation and improve its tractability.

In summary, a number of approaches have been developed in the past to simplify the runway sequencing problem, utilising the characteristics of the problem to do so. However, the assumptions which underlie these approaches fail to hold for departure or mixed mode sequencing, where large position shifts can be necessary for high quality results and time-windows are usually large (or open-ended) and may overlap with many other windows (preventing them from being used to simplify the problem). In addition, real world departure sequencing problem instances often require the consideration of complex objective functions that model trade-offs between multiple individual real world preferences (including delay, equity of delay, and time window compliance), further increasing the challenging nature of this problem.

This paper introduces a number of practical approaches for reducing the computation required for the general runway sequencing problem, expressed in terms of pruning rules for the search tree. Their efficacy in a dynamic program is illustrated on a number of complex real world take-off sequencing problems. A complex non-linear, non-convex and discontinuous objective function which was introduced in Atkin et al. (2007) and is based upon real controller preferences is utilised. Despite the challenging nature of the objective function (in particular for traditional MIP based approaches) and the conflicting objectives that it models, the pruning rules which are introduced here are shown to work well and to enable an algorithm to find optimal solutions extremely quickly - fast enough to be of practical use for real time runway scheduling. The effects of each of these rules is evaluated in terms of their effectiveness in reducing both the number of states in the problem and the solution time.

Pruning rules have received considerable attention in the literature on machine scheduling and help to improve tractability (Allahverdi et al. 1999, 2008). However, the majority of these approaches do not consider sequence dependent setup times, nor complex non-convex, non-linear, or discontinuous objective functions such as the one considered here. Earlier work on dominance rules that did consider sequence dependent setup times includes Ragatz (1993) and Bianco et al. (1999). Ragatz (1993) introduces a branch and bound algorithm to optimise total tardiness and prunes the search tree when local improvements can be achieved through a pairwise interchange of jobs without increasing the future cost. Similar approaches that exploit local improvement strategies were introduced by Luo and C. Chu (2007) for the maximum tardiness problem, Sourd (2005) for the earliness-tardiness problem, and by Luo et al. (2005) and Luo and Chu (2006). The branch and bound approach presented by Sewell et al. (2012) maintains a set of non-dominated solutions during the exploration of the search tree, and uses the set to establish dominance relationships for the current branch.

In contrast to approaches which reduce the search space by limiting the movement of aircraft within the sequence, the pruning of the search space introduced here exploits (in most cases) characteristics of the objective function to infer that the current sequence, or any future sequences based on it (by appending aircraft to it), is sub-optimal. Whilst the characteristics are investigated in the context of the objective function considered here, it can be easily shown that many of them transfer to other objective functions that are commonly considered in the literature. The advantage of exploiting characteristics of the objective function in the pruning rules is that partial sub-sequences which show known poor characteristics can be pruned much earlier, even before the dominating partial sequences have been generated. In addition, our pruning rules often have a much lower complexity when compared to some of the the pruning rules based on local improvements used in the approaches for machine scheduling listed above, and they are therefore usually more effective from a computational point of view.

The pruning rules which we introduce in this paper could also be of use in other approaches to this problem, having value beyond the specific dynamic programming approach which has been developed. Moreover, whilst the results that we present in this paper are for departure sequencing, the approach can also be applied to arrivals and mixed mode operations on a single runway (since both use the same, or simpler, constraints than the ones used here). Furthermore, when multiple runways are considered, runway allocations are usually pre-determined based on the departure route (to de-conflict), the aircraft size, or the position of the aircraft. In this case, the approach can easily be extended to support multiple runways, since the problem can be modelled as a single runway with a separation matrix that includes inter-runway separations if the runways are interacting, or it can be decomposed per runway if they are not interacting. Finally, the survey on machine scheduling carried out by Panwalkar and Iskander (1977) reports that 70% of schedulers state that setup times are sequence dependent in about 25% of the cases, and that the exact setup times depend on the degree of similarity between jobs, and hence are well structured. Given that the runway sequencing problem considered here is cast as a single machine scheduling problem with sequence dependent setup times, and considering the observation that many real world instances of such problems are well structured, our approach is expected to be applicable to a wide variety of similar problems, and could therefore have a significant impact on a large number of real world applications.

The outline of this paper is as follows: The runway sequencing problem is detailed in Section 2, where the constraints and objectives are explained in more detail, along with an explanation of the real world problem and the reasons for the structure within the data. The various pruning rules are then introduced in Section 3, where their complexity is discussed and a proof is provided for each, showing that the application will not result in a loss of optimality. For the purpose of illustrating their efficacy, these rules were implemented within a dynamic programming algorithm and this is explained in Section 3.7. The results of applying this algorithm to the take-off sequencing datasets for Heathrow (from Atkin et al. (2012)) are then given in Section 4, where the problems which would occur if a constrained position shift approach was used are also shown. Finally, some conclusions are drawn and the wider implications of this work are discussed in Section 5.

2. Problem description and model

Given a set of aircraft S, with (asymmetric) minimum separations δ_{ij} between any ordered pair of aircraft i and j (where i precedes j), the runway sequencing problem consists of finding a sequence of landings and take-offs, s, such that an optimal (or acceptable, for heuristic methods) value is achieved for some given objective function(s), subject to the satisfaction of all hard constraints.

2.1. Constraints

A feasible sequence must meet *minimum runway separations*, *hard time windows* (if applicable), and *earliest take-off times.* Any sequence that violates these constraints is not feasible in practice, and can be eliminated from the solution space.

2.1.1. Separation Constraints For the departure instances considered here, the minimum runway separations are determined by the aircraft's weight classes, speed groups, and their standard instrument departure routes (SIDs). An aircraft's weight class determines the severity of the wake turbulence it causes, the time that is required for this to dissipate, and its senstivity to wake turbulence caused by other aircraft. Larger aircraft generate, in general, more turbulence, to which smaller aircraft are more sensitive. Consequently, a larger weight class separation is required when a large aircraft is followed by a small aircraft, than when a small aircraft is followed by a large aircraft (i.e. the separations are asymmetric). In a similar fashion, larger speed group separations may be required when a slower aircraft is followed by a faster aircraft on the same route. This is necessary to prevent the following aircraft from catching up before their routes diverge. Minimum departure route separations are influenced by the climb and the relative bearing of the route, as well as congestion in downstream airspace sectors. The latter may require an increased separation upon take-off, to space out traffic and prevent the overloading of en-route sectors and controllers.

The minimum separation that must be maintained at the runway between the take-off time of two (departing) aircraft is equal to the maximum of their weight, speed, and SID separation. This results in a well structured separation matrix. For instance, the required separation for a fast and small aircraft is usually no less than the separation for a slow and large aircraft if they follow the same aircraft on the same route. However, the resulting separation matrix does not necessarily obey the triangle inequality. I.e., given three aircraft i, j, and k using the runway in the order i, then j, then k with the respective required separations between them denoted by δ_{ij}, δ_{jk}, and δ_{ik}, then $\delta_{ij} + \delta_{jk} \geq \delta_{ik}$ does not necessarily hold. The take-off time of one aircraft (e.g. k) can therefore be influenced by multiple-preceding aircraft (e.g., i and j).

2.1.2. Time Windows Let aircraft i be subject to a hard time window (that must be adhered to) that is defined by its start time et_i and end time lt_i, then its take-off time, t_i, must be within this window. I.e., $et_i \leq t_i \leq lt_i$ must hold. If an aircraft is not subject to a hard time window, it can be considered to be subject to a very large time window with start time et_i equal to $-M$ and end time lt_i equal to $+M$, with M denoting a very large constant (large enough to not interact with the aircraft times).

In addition to a hard time window, an aircraft which is taking off can be subject to a Calculated Take-Off Time (CTOT) or *slot*. A CTOT is a 15-minute time window during which the aircraft should take-off. Let the start and end time of the CTOT window for aircraft i be denoted by ec_i and lc_i respectively. An aircraft cannot take-off before ec_i and may have to be delayed to meet the start of its window. It preferably takes-off before its end, lc_i. Although their use is strongly discouraged and penalised, a limited number of 5 minute (300 seconds) CTOT extensions have been available and could be used for aircraft that would otherwise narrowly miss their CTOT. The start time of a CTOT window is therefore modelled as a hard constraint and the end time is modelled as a heavily penalised soft constraint or objective.

2.1.3. Earliest Take-Off Time Assuming that the earliest time an aircraft i can join the queue of aircraft waiting at the runway for take-off is b_i (which we name the "base time") and that the minimum time to reach the start of the queue and line up with the runway is c_i seconds, the earliest time aircraft i can be sequenced, irrespective of any other aircraft, is called the release time r_i and can be calculated as the maximum of $b_i + c_i$ and the start times of any hard or CTOT windows (et_i and ec_i). This is shown in Equation 1.

$$r_i = \max(b_i + c_i, et_i, ec_i) \tag{1}$$

Assuming that each aircraft will be sequenced as early as possible (which is a valid assumption at busy airports), the time t_i for aircraft i is equal to the maximum of r_i and $t_x + \delta_{xi}$ for all $x \in s_i$, where x denotes an aircraft in the partial sequence s_i of aircraft which take-off before i and t_x its take-off time. This is defined by Equation 2, in which r_i can be substituted by Equation 1.

$$t_i = \max(r_i, \max_{x \in s_i} \; t_x + \delta_{xi}) \tag{2}$$

2.2. Objectives

The objective function, $F(s)$, considered in our approach is defined by Equation 3 and models runway utilisation (quantified by the take-off time of the last aircraft in the partial or final sequence that contains all aircraft in the set, i.e. the makespan), total (non-linear) delay, and CTOT compliance. The runway utilisation is determined by the take-off time of the last aircraft in the sequence s and is equal to $\max_{x \in s} \; t_x$. Apart from its meaning as an objective (since it reflects runway utilisation), makespan is also utilised for the evaluation of partial sequences in the pruning rules and in the dynamic programming algorithm introduced in Section 3, since it affects future take-off times.

The objective function for delay and CTOT compliance is defined by the second component in Equation 3.

$$F(s) = (\max_{i \in s} \ t_i, \sum_{i \in s} (W_1(t_i - b_i)^{\alpha} + W_2 C(t_i, lc_i))) \tag{3}$$

$$C(t_i, lc_i) = \begin{cases} 0 & if \quad t_i \leq lc_i \\ \omega_1(t_i - lc_i) + \omega_2 & if \quad lc_i < t_i \leq lc_i + 300 \\ \omega_3(t_i - lc_i) + \omega_4 & if \quad t_i > (lc_i + 300) \end{cases} \tag{4}$$

The delay cost for an aircraft i is calculated as a function of the difference between its base time b_i and its take-off time t_i, and measured as $W_1(t_i - b_i)^{\alpha}$, where W_1 and α are constants ($\alpha \geq 1$) which can be set to appropriate values to model controller preferences (Atkin et al. 2010). Larger values of α penalise larger delays more severely and encourage a more equitable distribution of delay.

The cost for CTOT violations in Equation 3 is given by $W_2 C(t_i, lc_i)$, in which W_2 denotes a constant. $C(t_i, lc_i)$ is a non-convex discontinuous piecewise linear function that is defined by Equation 4, in which $\omega_1, \omega_2, \omega_3, \omega_4$ represent constants. The different segments of $C(t_i, lc_i)$ reflect the different costs associated with an aircraft taking off within its CTOT window ($ec_i \leq t_i \leq lc_i$), narrowly missing its departure window but leaving no more than 300 seconds late ($lc_i < t_i \leq lc_i + 300$), or missing its CTOT window completely ($lc_i + 300 < t_i$). Given the increasing degree of severity of missing a CTOT and hitting an extension, and missing CTOT and its extension completely, it is usual that $\omega_1 << \omega_3$ and $\omega_2 << \omega_4$. This results in a jump in cost in the objective function that recognises that small time window extensions are sometimes possible for departures but should be avoided, whereas missing an extension is extremely bad. We note that the trade-off between delay cost and slot compliance in Equation 3 can be influenced by setting their weight factors (W_1 and W_2) appropriately. A summary of the notation is provided in Table 1.

The objective function described above is based upon the function used in Atkin et al. (2007) which was defined in collaboration between those authors and the runway controllers at London Heathrow. It is currently used in one of the live decision support tools at London Heathrow. The tuning of this objective function to controller preferences was considered in Atkin et al. (2010). Its properties in the context of the approach introduced here are analysed in more detail in Section 3.2.

Symbol	*Definition*
W_1, W_2	Constant penalty weights for the different objective function components
ω_1, ω_2, ω_3, ω_4	Constant penalties for CTOT compliance
t_i	Take-off time of aircraft i
b_i	Base time for delay calculations, defined as the time the aircraft enters the runway queue for departures, or the local airspace for arrivals
r_i	Release date (earliest take-off time) of aircraft i
et_i	Earliest time of the hard time window for aircraft i
lt_i	Latest time of the hard time window for aircraft i
ec_i	Earliest time of the CTOT window for aircraft i
lc_i	Latest time of the CTOT window for aircraft i
α	Power index to balance delay vs. equity of delay ($\alpha \geq 1$)
δ_{ij}	The minimum required separation between aircraft i and j
n	The number of aircraft in the set
N	The number of aircraft types

Table 1 Summary of the notation

3. Pruning rules and solution method

This section introduces six pruning principles that can significantly reduce the solution time for real world runway sequencing problems, without losing optimality. They also apply to other machine scheduling problems with sequence dependent setup times that have objectives and structures similar to the runway sequencing problem. The pruning rules are explained in this section along with proofs that solution optimality is not lost by using them. A discussion of the applicability and performance of each rule to runway sequencing problems and other setups in which they hold is also given. Finally, the section ends with a description of the dynamic programming method which was developed to solve the runway sequencing problems and is used in Section 4.

Our pruning principles are:

1. A multi-objective extension of the ability to infer a *complete order within sets of separation identical aircraft* (Psaraftis 1980) for the non-additive objective function considered here
2. The ability to infer *conditional orders between sets of separation identical aircraft*
3. The ability to infer *conditional orders between sets of non-separation identical aircraft*
4. The use of *insertion dominance* which prunes sequences with intrinsically bad characteristics
5. The use of *dominance combined with lower bounding*
6. The identification of more *generic dominance rules* between partial sequences (i.e. to which other aircraft still have to be added to the end) that cover non-identical sets of aircraft

The first principle results in the generation of a number of distinct sets of aircraft. The remaining principles result in a tight coupling between those aircraft sets and enable new dominance relations to be inferred between partial sequences. All principles can be applied to prune partial sequences and are therefore particularly useful in algorithms that generate sequences by adding one aircraft at a time, such as the dynamic program used here (described in Section 3.7) or, e.g., branch and bound algorithms. However, they are also useful in other algorithms that are working on complete sequences (containing the entire set of aircraft), e.g. to verify whether a particular change to the sequence will compromise its optimality.

The validity of each of the six core principles above in the context of runway sequencing is discussed in the remainder of this section. The importance of our proofs lies in the fact that they not only prove the feasibility of our approach, but also generalise its applicability to any comparable sets of linear and/or higher order objectives for which $\alpha \geq 1$, $\omega_1 \geq \omega_3$, and $\omega_2 \geq \omega_4$. Their integration into a dynamic program for solving the departure sequencing problem is discussed in Section 3.7, however they could also be applied to augment many other approaches previously introduced in the literature.

3.1. Definitions

For the following proofs, let i and j denote two aircraft that are *separation identical*, i.e. the mutual separations for aircraft i and j with respect to all other aircraft x in the set S (which includes i and j) are the same ($\delta_{ix} = \delta_{jx}, \delta_{xi} = \delta_{xj}\ \forall x \in S$). Let k denote an aircraft that is not separation identical to i, i.e. $\exists x \in S : \delta_{ix} \neq \delta_{kx}$ or $\delta_{xi} \neq \delta_{xk}$. Aircraft k is said to be "*more difficult*" to sequence than aircraft i with respect to a set of aircraft S if the mutual separations between k and any aircraft $x \in S$ are no less than the respective separations between i and x, and strictly greater for at least one aircraft $x \in S$ (i.e. $\delta_{kx} \geq \delta_{ix}, \delta_{xk} \geq \delta_{xi}\ \forall x \in S$ and $\exists x \in S : \delta_{kx} > \delta_{ix}$ or $\delta_{xk} > \delta_{xi}$).

Let the base times for i, j, and k be denoted by b_i, b_j, b_k, respectively, the start times of the hard and CTOT windows be denoted by et_i, et_j, et_k and ec_i, ec_j, ec_k, respectively, and the ends of hard and CTOT windows be denoted by lt_i, lt_j, lt_k and lc_i, lc_j, lc_k, respectively. Finally, let t_x denote the take-off time of aircraft x in sequence s and t'_x denote the take-off time of aircraft x in sequence s'.

3.2. Complete orders within sets of separation identical aircraft

A complete order exists between aircraft i and j if the objective value(s) and feasibility of any arbitrary sequence s including i and j cannot, under any circumstances, be improved by reversing the order of i and j in s. If such complete orders exist, the sequencing problem can be simplified to one of interleaving ordered sets of aircraft, always sequencing the first available aircraft from the respective sets. The existence of such complete orders between separation identical aircraft was shown by Psaraftis (1980) for the optimisation of processing cost, enabling a reduction in the complexity of the problem from factorial as a function of the number of aircraft n (i.e. $n!$) to exponential as a function of the number of aircraft types N, and equal to $O(N^2(n+1)^N)$.

In a multi-objective context, a complete order may be inferred upon a set of aircraft if the complete orders for each of the individual constraints and objectives are consistent within the set. The formal proofs below show that this is the case for the makespan and delay objectives, even with hard time window constraints if the base times (b_i), release dates (r_i), and the end times of hard time windows (lt_i) of the individual aircraft are in order. However, this is not the case for the cost incurred by CTOT windows.

3.2.1. Initial observations We first present some initial observations which can be used to simplify later proofs.

LEMMA 1. *Given two sub-sequences s and s' with identical aircraft in the same order, differing only in the take-off times (e.g. due to different aircraft preceeding s and s'), if $t_x \leq t'_x$ for all aircraft x in s and s', the delay (or CTOT) cost for each individual aircraft in s will be no worse than its delay (or CTOT) cost in s', and the total delay (or CTOT) cost summed over all aircraft in s will be no worse than the total delay (or CTOT) cost summed over all aircraft in s'.*

Proof: Delay (or CTOT) costs monotonically increase with time, hence the pairwise cost relationship holds between corresponding aircraft. The relationship for the total cost summed over all aircraft in s then follows.

□

LEMMA 2. *Given two sub-sequences s and s' with identical aircraft in the same order, differing only in the take-off times, if $t_x \leq t'_x$ for all aircraft x in s and s', the violation of an aircraft's hard time window in s will be no worse than its violation in s', and hence if sequence s is infeasible, so will be sequence s'.*

Proof: Since the aircraft order in s and s' is identical, and since x cannot be sequenced before et_x and can always be delayed to meet et_x, $t_x \geq et_x$ is trivial. Since $t_x \leq t'_x$, if $t'_x \leq lt_x$, then $t_x \leq t'_x \leq lt_x$, hence any aircraft x in s cannot violate the time window in s (i.e. $t_x > lt_x$) if x does not in s' (i.e. $t'_x > lt_x$).

□

LEMMA 3. *Given two aircraft sets A and $A \cup x$, the delay (or CTOT) cost for a sequence s based on A is no less than the delay (or CTOT) cost for a sequence s' of $A \cup x$ with identical aircraft order for the aircraft in A and with x inserted at any arbitrary position, both with or without including the cost for x (without considering the cost of the aircraft remaining to be added).*

Proof: Inserting an additional aircraft in any sequence cannot decrease the times for subsequent aircraft. The delay (or CTOT) cost is monotonically increasing and the delay (or CTOT) cost for x is non-negative (Equation 3).

□

LEMMA 4. *Let s and s' denote two sequences based on the sets A and $A \cup x$, respectively, for which the order of the aircraft in A is the same in s and s', and with x inserted at any arbitrary position in s' (i.e. not appended). The violation of hard time windows for s is no less than for s', both with or without considering the violation for x, and without considering the cost of the aircraft remaining to be added.*

Proof: Inserting an additional aircraft in any sequence cannot decrease the times for subsequent aircraft. It then follows that violations of hard time windows cannot decrease (Lemma 2).

□

3.2.2. Makespan

THEOREM 1. *An objective to minimise makespan can be considered to induce a complete order upon two separation identical aircraft i and j if $r_i \leq r_j$.*

Proof: Let s denote a partial sequence containing i and j (in that order), and let s' denote a partial sequence with identical order except that i and j are reversed. Let the times allocated to i and j be denoted by t_i and t_j for s, and by t'_i and t'_j for s'. We will prove that if $r_i \leq r_j$, then any sequence with prefix s will be no worse than the equivalent sequence with prefix s', and hence a complete order may be inferred between i and j for makespan. The proof has four components. Firstly we show that $t_i \leq t'_j$. Secondly, we show through induction that the corresponding times for the aircraft between i and j in s (j and i in s') are no later in s than s'. Thirdly, we show that $t_j \leq t'_i$. Fourthly, we show that the inductive proof in the second part therefore also holds for any further aircraft which could be added to the end of s and s', completing the proof.

Part 1: Let s_i and s'_j denote the identical partial sequences of aircraft which are sequenced before i in s and before j in s'. Given the definitions of s and s', $s_i = s'_j$. From Equation 2 we know $t_i = max(r_i,\ t_x + \delta_{xi}\ \forall x \in s_i)$ and $t'_j = max(r_j,\ t_x + \delta_{xj}\ \forall x \in s_i)$. Since i and j are separation identical and $r_i \leq r_j$, then $t_i \leq t'_j$.

Part 2: Let y denote any aircraft between i and j in s (between j and i in s') such that t_y and t'_y denote the times of y in s and s', respectively. Let s_y denote the sequence of aircraft which are before y in s and let s'_y denote the sequence of aircraft which are before y in s'. From Equation 2 we know $t_y = max(r_y, t_x + \delta_{xy}\ \forall x \in s_y)$ and $t'_y = max(r_y, t'_x + \delta_{xy}\ \forall x \in s'_y)$. By consideration of corresponding terms between the two equations, $t_y \leq t'_y$ if $t_x \leq t'_x$ for all x prior to y. $t_x = t'_x$ for all $x \in s_i$, since the aircraft are identical, and $t_i \leq t'_j$ (from part 1). Thus, by induction $t_y \leq t'_y$ for all y between i and j in s.

Part 3: Let s_j denote the partial sequence of aircraft before j in s and let s'_i denote the sequence of aircraft before i in s'. From Equation 2, $t_j = max(r_j,\ t_x + \delta_{xj}\ \forall x \in s_j)$ and $t'_i = max(r_i,\ t'_x + \delta_{xi}\ \forall x \in s'_i)$. From part 2, we know that $t'_{x'} \geq t_x$ for all x and x' at corresponding positions in s_j. Since j is sequenced before i in s', $t'_i \geq r_j$, and thus $t'_i \geq t_j$.

Part 4: The inductive proof from part 2 therefore also applies to t_j and t'_i, and, thus, to all subsequent aircraft, including the ones which will be added to the end of the sequence.

□

Lemma 5. *Given the definitions of s, s', t_i, t_j, t'_i, t'_j in the proof of Theorem 1, $t_i \leq t'_j \leq t_j \leq t'_i$ holds.*

Proof: The sequence of aircraft prior to j in s' is a sub-sequence of the sequence of aircraft prior to j in s, thus $t'_j \leq t_j$. From Theorem 1, $t_i \leq t'_j$ and $t_j \leq t'_i$.

□

Lemma 6. *Given the definitions in Theorem 1, $t_x \leq t'_{x'}$ for all aircraft x and x' in corresponding positions in s and s', respectively.*

Proof: This is a direct consequence of Theorem 1.

□

LEMMA 7. *If an aircraft x is appended to both of the sequences s and s' defined in Theorem 1 with times t_x and t'_x respectively, then $t_x \leq t'_x$.*

Proof: The inductive proof of Part 4 also applies to aircraft x, and to any subsequent aircraft.

□

3.2.3. Delay

THEOREM 2. *An objective to minimise the cost for delay in Equation 3 can be considered to induce a complete order upon two separation identical aircraft i and j where $b_i \leq b_j$ and $r_i \leq r_j$.*

Proof: Assume the same definitions for s, s', t_i, t_j, t'_i, t'_j as in Theorem 1. $t_x \leq t'_{x'}$ for all aircraft x and x' in corresponding positions in s and s' respectively (Lemma 6) and delay costs monotonically increase (Lemma 1). An objective to minimise delay can therefore induce a complete order upon i and j if Inequality 5 holds:

$$W_1(t_i - b_i)^\alpha + W_1(t_j - b_j)^\alpha \leq W_1(t'_j - b_j)^\alpha + W_1(t'_i - b_i)^\alpha \tag{5}$$

Since the conditions for Lemma 5 hold, we know $t_i \leq t'_j \leq t_j \leq t'_i$, so we can define $x_1, x_2, x_3 \geq 0$ such that $t'_j = t_i + x_1$, $t_j = t'_j + x_2 = t_i + x_1 + x_2$, $t'_i = t_j + x_3 = t_i + x_1 + x_2 + x_3$. Inequality 6 is then equivalent to Inequality 5.

$$(t_i - b_i)^\alpha + (t_i + x_1 + x_2 - b_j)^\alpha \leq (t_i + x_1 - b_j)^\alpha + (t_i + x_1 + x_2 + x_3 - b_i)^\alpha \tag{6}$$

Let $x_2 = 0$, then Inequality 6 becomes Inequality 7 or 8, which holds for all $x_1, x_3 \geq 0$.

$$(t_i - b_i)^\alpha + (t_i + x_1 - b_j)^\alpha \leq (t_i + x_1 - b_j)^\alpha + (t_i + x_1 + x_3 - b_i)^\alpha \tag{7}$$

$$(t_i - b_i)^\alpha \leq (t_i + x_1 + x_3 - b_i)^\alpha \tag{8}$$

Thus, Inequality 6 holds for $x_2 = 0$. As x_2 is increased $(t_i + x_1 + x_2 + x_3 - b_i)^\alpha$ will increase faster than $(t_i + x_1 + x_2 - b_j)^\alpha$, since $b_i \leq b_j$ and $x_3 \geq 0$. Inequalities 5 and 6 therefore hold for all $x_1, x_2, x_3 \geq 0$, thus the cost of s can be no greater than the cost of s', so there is never a benefit from sequencing j before i.

□

3.2.4. Time windows

THEOREM 3. *Hard time windows can be considered to induce a complete order upon two separation identical aircraft i and j where $r_i \leq r_j$ and $lt_i \leq lt_j$*

Proof: Assume the same definitions of s, s', t_i, t_j, t'_i, t'_j as in Theorem 1. Since an aircraft can always be delayed to meet the start of its window, $et_x \leq t_x$ is trivial. Since $t_x \leq t'_{x'}$ for all aircraft x and x' at corresponding positions in s and s' (Lemma 6), the time window violation of all aircraft other than i and j is no worse in s than in s' (Lemma 2). If i misses its hard time window in s (i.e. $t_i > lt_i$), irrespective of j, i will also miss its time window in s', since $t'_i \geq t_i$, thus $t'_i > lt_i$. If j misses its time window in s (i.e. $t_j > lt_j$), then i will miss its time window in s', since $t'_i \geq t_j$ and $lt_i \leq lt_j$. Since $t_x \leq t'_x$ (Lemma 7) for any arbitrary aircraft x added to both s and s', the time window violation for x is no worse in the case of s. Hence, a complete order can be inferred between i and j for time window violations.

□

LEMMA 8. *A complete order can be inferred within a set of separation identical aircraft with respect to makespan, delay, and hard time window compliance if the base times (b_x), release dates (r_x), and the end times of hard time windows (lt_x) are in the same order for all aircraft x in the set.*

Proof: The necessary conditions for Theorem 1 (the release dates are in order), Theorem 2 (the base times and release dates are in order), and Theorem 3 (the release dates and end times are in order) are satisfied for all ordered pairs of aircraft in the set, hence a complete order can be inferred.

□

In contrast to Lemma 8, complete orders cannot be inferred within separation identical sets when CTOT windows are considered due to the piecewise linear, discontinuous and non-convex objective function that models their cost. I.e., the better order for two separation identical aircraft i and j ($b_i \leq b_j$,$r_i \leq r_j$, $lc_i \leq lc_j$) depends upon the times t_i and t_j. For example, let us assume that j is restricted by a CTOT window but i is not. Let s denote a partial sequence in which i is sequenced at time $t_i \geq r_j$ (i.e. after j becomes available for sequencing), and j is sequenced at time $t_j \geq lc_j$ (i.e., j misses its time window). Since $t_i \geq r_j$, and i and j are separation identical, the aircraft could be swapped with no modification of times to i, j, or other aircraft. Even though the swap may potentially increase the delay cost (since $b_i \leq b_j$, $r_i \leq r_j$) the reduction in the CTOT violation cost from scheduling j earlier could more than offset this. I.e., the total cost could be reduced. Conversely, if $r_j = t_j$ (i.e., j could not be sequenced any earlier), there is no benefit from sequencing j before i, since this would not reduce j's CTOT violation cost, so the total cost for i and j (see §3.2.3) would not improve and could even increase.

If both i and j are subject to a CTOT, then the lower cost order will depend upon the relationship between t_i, t_j, ec_i, ec_j, lc_i, and lc_j. Given the CTOTs for i and j and the possible times at which i and j can be scheduled, say t_1 and t_2, with $t_1 < t_2$, if both aircraft can meet their time windows with i scheduled before j (i.e., $t_i = t_1 \leq lc_i$ and $t_j = t_2 \leq lc_j$), then i should precede j for reasons of delay. However, if $lc_i \leq t_1 < lc_j \leq t_2$ then swapping the aircraft so that $t_j = t_1$ and $t_i = t_2$ would mean that only i misses its time window. This could result in a lower total CTOT violation cost which could more than offset the increased delay cost.

Performance: An efficient algorithm would implement complete orders by generating and ordering the separation identical sets in a pre-processing step, i.e. before the actual sequencing is done. This can be done through pairwise comparison of aircraft and their separations with the aircraft in the set S. The solution method can then interleave the ordered sets by selecting the first available aircraft in each of the sets and avoid consideration of later aircraft. If the solution method is exact, optimality of the resulting sequences will not be compromised since a complete order exists within the sets. It was shown by Psaraftis (1980) that interleaving ordered sets of aircraft reduces the worst case complexity from $n!$ to $O(N^2(n+1)^N)$, with N denoting the number of sets and n denoting the number of aircraft. I.e., complete orders reduce the computational complexity of the algorithm and require no additional computation during its execution.

The efficacy of using complete orders is highly influenced by the complexity and structure of the separation matrix, and the aircraft mix that operates at the airport in practice. In practice, the separation matrices for runway sequencing problems have a structure which enables complete orders to be exploited well. However, in extreme cases, e.g., where all aircraft are subject to a CTOT, when the aircraft mix is highly diverse, or when no separation identical aircraft are present, no complete orders can be inferred. Even in this case, however, the pruning rules introduced below can help to improve tractability.

3.3. Conditional orders

3.3.1. Conditional orders between sets of separation identical aircraft When partial sequences are compared, it will often be possible to infer some information about take-off times (e.g. lower bounds) and the aircraft order required to obtain an optimal sequence taking into account the characteristics of the given objective function, even if the exact take-off times are not yet known. This section looks at how an algorithm could use this information to prune partial sequences much earlier, i.e. before the sequences are generated and before the exact times are known.

THEOREM 4. *A conditional order can be inferred between two separation identical aircraft i and j ($r_i \le r_j$, $lt_i \le lt_j$), such that i should precede j, when t_i, t_j, t'_i, t'_j are such that Inequality 9 holds.*

$$\begin{aligned} W_1(t_i - b_i)^\alpha + W_2C(t_i, lc_i) + W_1(t_j - b_j)^\alpha + W_2C(t_j, lc_j) \le \\ W_1(t'_i - b_i)^\alpha + W_2C(t'_i, lc_i) + W_1(t'_j - b_j)^\alpha + W_2C(t'_j, lc_j) \end{aligned} \tag{9}$$

Proof: Let us assume the definitions of s, s', t_i, t_j, t'_i, t'_j in Theorem 1. Since $r_i \le r_j$ and $lt_i \le lt_j$, a complete order can be inferred between i and j with respect to makespan and hard time window violations. Hence, $t_x \le t'_x$ for all aircraft other than i and j at corresponding positions in s and s' (Lemma 6), or at corresponding positions in sequences obtained by adding a subsequence containing the same aircraft in the same order to s and s' (Lemma 7). It then follows that sequencing j before i could not decrease the makespan, the delay cost, the CTOT violation cost, and the violation of hard time windows for any of these aircraft (Lemmas 1 and 2). Hence, an order can be inferred between i and j such that i should precede j if Inequality 9 holds, and thus the cost for sequencing i before j is lower than the cost of sequencing j before i.

$\square$

THEOREM 5. *If t_j and t'_i are not yet known (e.g. when incrementally building up the sequence), a conditional order can still be inferred between i and j if, in addition to the conditions outlined in Theorem 4, $b_i \le b_j$, $lc_i \le lc_j$ and Inequality 10 hold.*

$$\begin{aligned} W_1(t_i - b_i)^\alpha + W_2C(t_i, lc_i) - W_1(t'_j - b_j)^\alpha - W_2C(t'_j, lc_j) \le \\ W_1(t'_i - b_i)^\alpha + W_2C(t'_i, lc_i) - W_1(t'_i - b_j)^\alpha - W_2C(t'_i, lc_j) \end{aligned} \tag{10}$$

Proof: If $r_i \le r_j$, $lt_i \le lt_j$, and $b_i \le b_j$, a complete order can be inferred between i and j for makespan, delay, and hard time window violations. Since delay costs are monotonically increasing and $t_j \le t'_i$ (Lemma 5), $W_1(t'_i - b_j)^\alpha \ge W_1(t_j - b_j)^\alpha$, and thus $W_1(t'_i - b_i)^\alpha - W_1(t_j - b_j)^\alpha \ge W_1(t'_i - b_i)^\alpha - W_1(t'_i - b_j)^\alpha$. Since CTOT violation costs are monotonically increasing for $\omega_1 \le \omega_3$ and $\omega_2 \le \omega_4$ (Equation 4) and $t_j \le t'_i$ (Lemma 5), $C(t'_i, lc_j) \ge C(t_j, lc_j)$, and thus $C(t'_i, lc_i) - C(t_j, lc_j) \ge C(t'_i, lc_i) - C(t'_i, lc_j)$. Meeting Inequality 10 is therefore sufficient for meeting Inequality 9, and a conditional order can be inferred between i and j as soon as Inequality 10 is met.

$\square$

If $b_i \le b_j$, the minimum value for $W_1(t'_i - b_i)^\alpha - W_1(t'_i - b_j)^\alpha$ occurs at minimal t'_i. The minimum value of $C(t'_i, lc_i) - C(t'_i, lc_j)$ occurs either at minimal t'_i or around the discontinuities in Equation 4, and can therefore be calculated easily once an earliest time for t'_i is known, even before the exact value for t'_i is known. Time t'_i can be no earlier than the latest take-off time for an aircraft which is already in the partial sequence, and will increase as more aircraft are added, thereby further tightening Inequality 10.

A special case arises if $t_i \le lc_i$ ($lc_i \le lc_j$), i.e. if i can meet its time window. In this case, Inequality 9 reduces to Inequality 11. A complete order exists for delay if $r_i \le r_j$ and $b_i \le b_j$, and thus $W_1(t_i - b_i)^\alpha + W_1(t_j - b_j)^\alpha \le W_1(t'_i - b_i)^\alpha + W_1(t'_j - b_j)^\alpha$. From Equation 4, $C(t_j, lc_j) \le C(t'_i, lc_i)$ for $lc_i \le lc_j$, $t_j \le t'_i$ (Lemma 5), and $C(t'_j, lc_j) \ge 0$. Thus, Inequality 11 must hold and a conditional order can be inferred between i and j if $r_i \le r_j$, $b_i \le b_j$, $lt_i \le lt_j$, $lc_i \le lc_j$ and $t_i \le lc_i$.

$$\begin{aligned} &W_1(t_i - b_i)^\alpha + W_1(t_j - b_j)^\alpha + W_2 C(t_j, lc_j) \le \\ &\qquad W_1(t'_i - b_i)^\alpha + W_2 C(t'_i, lc_i) + W_1(t'_j - b_j)^\alpha + W_2 C(t'_j, lc_j) \end{aligned} \tag{11}$$

If $r_i \le r_j$, $b_i \le b_j$, and $lt_i \le lt_j$ and aircraft i and j are not subject to a CTOT window, their cost is equal to 0, and Theorem 5 reduces to Lemma 8, in which case a complete order exists between i and j. Finally, if aircraft j does not have a CTOT window, Inequality 9 reduces to Inequality 12, which is always satisfied (since a complete order exists for delay, the CTOT window cost is monotonically increasing, and $t_i \le t'_i$, Lemma 5). I.e. a complete order exists between i and j in this case.

$$\begin{aligned} &W_1(t_i - b_i)^\alpha + W_2 C(t_i, lc_i) + W_1(t_j - b_j)^\alpha \le \\ &\qquad W_1(t'_i - b_i)^\alpha + W_2 C(t'_i, lc_i) + W_1(t'_j - b_j)^\alpha \end{aligned} \tag{12}$$

3.3.2. Conditional orders between non-separation identical aircraft

Theorem 6. *A conditional order can be inferred between two non-separation identical aircraft i and k (i more difficult to sequence than k, $r_i \le r_k$, $lt_i \le lt_k$) such that i should precede k if the increased separations for i compared to k do not impose an additional delay for any subsequent aircraft in the current partial sequence, or any aircraft remaining to be added to the current sequence when t_i, t_k, t'_i, t'_k are such that Inequality 13 holds.*

$$\begin{aligned} &W_1(t_i - b_i)^\alpha + W_2 C(t_i, lc_i) + W_1(t_k - b_k)^\alpha + W_2 C(t_k, lc_k) \le \\ &\qquad W_1(t'_i - b_i)^\alpha + W_2 C(t'_i, lc_i) + W_1(t'_k - b_k)^\alpha + W_2 C(t'_k, lc_k) \end{aligned} \tag{13}$$

Proof: Let s denote a partial sequence containing i and k (in that order), and let s' denote a partial sequence with identical order except that i and k are reversed. Let the set of aircraft remaining to be added to s and s' be denoted by R. If no additional delays are incurred by the aircraft subsequent to i in s or by any aircraft in R (relative to s'), then $t_x \le t'_x$ for all aircraft other than i and k. The makespan, delay cost, violation of hard time windows, and CTOT violation cost for these aircraft is thus no worse in s than in s' (Theorem 1 and Lemmas 1 and 2). Since $t_i \leq t'_i$ (the sequence of aircraft before i in s is a subsequence of the aircraft before i in s') and $t_k \leq t'_i$ (the increased separations for i compared to k do not impose additional delay), the observations for hard time windows in Theorem 3 remain valid. If Inequality 13 holds, the delay cost and the cost for CTOT violations for i and k is no worse in s than in s' and a conditional order can be inferred for i and k.

□

In a similar fashion as in §3.3.1, a conditional order can still be inferred between i and k even if the exact values of t_k and t'_i are not yet known. This is the case if the current increase in delay and time window cost for scheduling i rather than k is less than any future decrease in delay and time window cost when k and i are later added, as shown by Inequality 14 (a rearrangement of Inequality 13). We note that with respect to the cost for CTOT violations in Inequality 14, the observations from §3.3.1 remain valid.

$$\begin{aligned} W_1(t_i - b_i)^\alpha + W_2C(t_i, lc_i) - W_1(t'_k - b_k)^\alpha - W_2C(t'_k, lc_k) \leq \\ W_1(t'_i - b_i)^\alpha + W_2C(t'_i, lc_i) - W_1(t_k - b_k)^\alpha - W_2C(t_k, lc_k) \end{aligned} \quad (14)$$

Performance: To infer conditional orders when incrementally building up a sequence, the conditions in Theorems 4, 5 and 6 must be validated between the newly added aircraft and both the aircraft preceding it in the sequence and the aircraft remaining to be added to the sequence. The complexity of validating conditional orders is therefore linear as a function of the number of aircraft in S, denoted by $|S|$, since any aircraft is either added to the sequence or remaining to be added. If complete orders are present in S, and hence a number of ordered separation identical sets has been defined, conditional orders have to be evaluated only for the first remaining aircraft in each of the N sets. In addition, they only have to be evaluated for aircraft that are either separation identical or are more difficult to sequence, and for which replacing them with j does not impose an additional delay on subsequent aircraft in the sequence (if $r_j >> t_x$, aircraft j is likely to impose an additional delay). The actual number of comparisons is therefore likely to be significantly fewer in practice. This makes the implementation of conditional orders very efficient from a computational point of view.

Extension: We note that conditional orders can be generalised to any arbitrary objective function/constraints, or any number of aircraft, as long as their exact times in the sequence are known. Indeed, if a local improvement can be obtained without increasing the future cost, the partial sequence, or any sequence based on the unimproved order, is not optimal. This is also the case if the exact times of t_j and t'_i are not yet known, as long as it can be establised for the given objective function that the current increase in cost for sequencing j before i is less than any future decrease.

3.4. Insertion dominance

THEOREM 7. *If an aircraft x can be inserted into a partial sequence s (i.e. not appended) without delaying any of the subsequent and remaining aircraft, the sequence s can be pruned without compromising optimality.*

Proof: Since x can be inserted into s without delaying any of the subsequent and remaining aircraft, the makespan, delay cost, cost for CTOT violations, and the violation of hard time windows for these other aircraft does not increase (Theorem 1 and Lemmas 1 and 2). If x is scheduled after s, its time can be no earlier than if it was scheduled within s, thus the makespan, delay cost, CTOT violation cost (which are monotonically increasing), and the violation of hard time windows for x can also be no less (Theorem 1 and Lemmas 1 and 2). Therefore, the sequence based on s and containing x can be no worse than the sequence based on s to which x is appended later.

□

Performance: To evaluate insertion dominance, it is necessary to verify whether any of the remaining aircraft $x \in R$ can be inserted into s without additional delay to the subsequent aircraft in s, or to the aircraft in $R \setminus x$. In practice, since complete orders exist between the aircraft in the same separation identical set, it is sufficient to validate this dominance rule only for the first remaining aircraft in each of the sets, and only for the positions in s where the maximum separation for x with any arbitrary aircraft exceeds the makespan of s augmented with the minimum separation (i.e., it can still influence future take-off times). The complexity of evaluating insertion dominance is therefore linear as a function of the number of separation identical sets N, and linear as a function of the number of positions that need to be considered for insertion dominance as determined by the maximum separation (this is 2 for the problem instances considered here).

The efficacy of insertion dominance is influenced by three factors: the distribution of the release dates r_i; the accumulated delay; and the occurrence of violations of the triangle inequality in the separation rules. If all release dates are equal, or it is a high delay situation, an aircraft x's release date, r_x, is less likely to delay its take-off time. I.e., t_x is likely to be constrained by the separation requirements only. Hence, the sequence may not contain idle time, so insertion dominance may not apply. However, if the separation rules violate the triangle inequality, i.e. $\delta_{ij} \geq \delta_{ix} + \delta_{xj}$ (with i preceding j in s), it may be possible to insert x between i and j without causing any additional delay to other aircraft, and hence without increasing their cost. In this case, s can be pruned without compromising optimality. In practice, the efficacy of insertion dominance is determined by a complex interaction between these three key factors (release dates, delay, and separation rules). We therefore report empirical results on its efficacy in Section 4.

Extension: We note that insertion dominance is valid for any monotonically increasing objective function. It can be easily and efficiently extended to "conditional insertion dominance" for an arbitrary objective function, provided that it can be easily verified that the cost for inserting an aircraft is less than the future cost for adding the aircraft later.

3.5. Dominance with lower bounding

The presence of sequence dependent separations that violate the triangle inequality (e.g. for departure sequencing, mixed mode operations, or for multiple runway scenarios) means that the take-off time(s) and objective value(s) of future aircraft added to a given sequence s can be influenced by one or more preceding aircraft, typically the most recently sequenced ones. The set of aircraft in s that influence the times of future aircraft is called the *"separation influencing set"* and consists of all aircraft $x \in s$ for which the separation constraints $t_x + \delta_{xy}$ may be binding upon the take-off time t_y of any arbitrary aircraft y in the set of aircraft R remaining to be added to s. The set of other aircraft in s, i.e. the ones that are not binding upon the take-off time of any aircraft $y \in R$ is called the *"non-separation influencing" set.*

Since the separation influencing set can affect the take-off time of future aircraft, and hence their objective value(s), two sequences s and s' are comparable only if their separation influencing sets are the same if standard dominance rules are used. I.e. sequence s is no worse than sequence s' if $F(s) \leq F(s')$ and $t_x \leq t'_x$ for all aircraft x in the separation identical sets. This problem characteristic greatly increases the complexity of the problem by increasing the number of non-comparable sequences by a factor of $m!$, where m is the number of separation influencing aircraft. I.e., the requirement for separation identical sets to be the same can significantly reduce the efficacy of pruning rules.

The requirements on the separation identical sets of s and s' can be relaxed by integrating "look-ahead" or lower bounding strategies to consider the effects of aircraft in the separation influencing sets on the set of aircraft, R, that remain to be added to the partial sequence s. This enables the inference of dominance relations between otherwise incomparable sequences, and significantly increases the number of sequences that can be compared.

Theorem 8. *Given partial sequences s and s' that contain the same set of aircraft, and a set of aircraft R which have not yet been added, any sequence based on s is no worse than a sequence based on s' if s is feasible, $F(s) \leq F(s')$ and $\max_{x\in s}(t_x + \delta_{xy}, r_y) \leq \max_{x\in s'}(t'_x + \delta_{xy}, r_y)$ $\forall y \in R$*

Proof: Let z be a sequence consisting of sub-sequence s followed by any (sub-)set of aircraft in R and let z' be the sequence consisting of the sub-sequence s' followed by the same (sub-)set of aircraft from R in the same order. Let t_y and t'_y denote the times for $y \in R$ in sequences z and z', respectively. Then $t_y \leq t'_y$ $\forall y \in R$, since $\max_{x\in s}(t_x + \delta_{xy}, r_y) \leq \max_{x\in s'}(t'_x + \delta_{xy}, r_y)$ $\forall y \in R$. It follows that the makespan, the delay cost, the cost for CTOT violations, and the violation of hard time windows for z based on s is no worse than for z' based on s' (Theorem 1, Lemmas 1 and 2), so the sequence s', and any sequence based on it, can be pruned without compromising optimality.

□

Performance: To evaluate dominance with lower bounding, it is sufficient to identify all comparable sequences (i.e. sequences that contain the same set of aircraft but do not neccesarily have the same separation influencing set) and validate the conditions in Theorem 8. If the sequences are ordered and grouped by their aircraft set, the sets of comparable sequences can be located using binary search. I.e. the worst case complexity is given by $O(log_2 N)$, with N denoting the number of unique aircraft sets in this case. The worst case value of N is determined by the number of unique subsets that can be selected from S, and is given by $N = \frac{|S|!}{|s|!(|S|-|s|)!}$, with $|s|$ denoting the number of aircraft in sequence s. The value of N reaches a maximum for $|s| = \frac{|S|}{2}$. However, the other pruning rules introduced in this paper will significantly reduce the number of unique sets in practice (i.e., the value of N). Hence, the average complexity can be expected to be significantly less than the worst case complexity, making the implementation of dominance with lower bounding computationally very efficient.

Extension: We note that dominance with lower bounding applies to any arbitrary cost function, even if the aircraft do not have to be scheduled as early as possible. Given $t_y \leq t'_y \ \forall y \in R$, the objective value of y in z will never exceed its value in z', since y in z can always be delayed to t'_y (if this were to improve the objective value) but y in z' cannot be advanced to t_y.

3.6. Dominance between non-identical sets

3.6.1. Dominance considering subsets

Theorem 9. *Let s be a partial sequence containing the aircraft from set S and let s' be a partial sequence containing the aircraft from set S', where $S' \subset S$. Let R denote the set of aircraft which have not yet been sequenced in s and R' denote the set of aircraft which have not yet been sequenced in s'. If s is feasible, $F(s) \leq F(s')$ and $\max_{x\in s}(t_x + \delta_{xy}, r_y) \leq \max_{x\in s'}(t'_x + \delta_{xy}, r_y) \ \forall y \in R$, then the sequence s' and any sequence based on it can be pruned without compromising optimality.*

Proof: Given the definition of s, s', S, S', R and R', $R \subset R'$ in Theorem 8, the sequence obtained by adding the aircraft from R after s can not be worse than the one obtained by adding the same aircraft from R' (in the same order) after s'. Inserting the additional aircraft from $R' \setminus R$ cannot reduce the makespan, the delay cost, the cost for CTOT violations, and violation of hard time windows (Theorem 1 and Lemmas 3 and 4), regardless of their positions. Thus, the sequence based on s and R cannot be worse than the sequence based on s' and R'. The sequence s', and any sequences based it, can therefore be pruned without compromising optimality.

□

We note that this dominance relationship can be further tightened by considering a lower bound L for the total cost of the aircraft in $R' \setminus R$, calculated from their earliest take-off times given that they must follow s', and consider whether $F(s) \leq F(s') + L$ rather than $F(s) \leq F(s')$.

Performance: Similarly to dominance with lower bounding, dominance between subsets requires the retrieval of the set of all sequences containing a given set of aircraft, which can be done in $O(log_2\ N)$. This has to be repeated for all subsets of the aircraft in s that one would want to consider. Hence, the complexity of validating dominance between subsets is a linear function of the number of subsets that are considered, and logarithmic as a function of the number of unique aircraft sets.

Extension: We note that dominance with subsets is applicable for any arbitrary non-negative cost function.

3.6.2. Dominance considering non-identical sets

THEOREM 10. *Let s and s' be arbitrary partial sequences containing the aircraft from set $S \cup i$ and $S \cup k$, respectively (i more difficult to sequence than k, $r_i \leq r_k$, $b_i \leq b_k$, $lc_i \leq lc_k$, $lt_i \leq lt_k$). Let $R \cup k$ and $R \cup i$ denote the sets of aircraft which have not yet been sequenced in s and s', respectively. If s is feasible, $\max_{x \in s}(t_x + \delta_{xy}, r_y) \leq \max_{x \in s'}(t'_x + \delta_{xy}, r_y)\ \forall y \in R$, $\max_{x \in s}(t_x + \delta_{xk}, r_k) \leq \max_{x \in s'}(t'_x + \delta_{xi}, r_i)$, and Inequality 15 holds, then s' (and any sequence based on it) can be pruned from the solution space without compromising optimality.*

$$F(s) - F(s') \leq W_1(t - b_i)^{\alpha} + W_2 C(t, lc_i) - W_1(t - b_k)^{\alpha} - W_2 C(t, lc_k)\ \forall t \geq \max_{x \in s'}(t_x + \delta_{xi}, r_i) \quad (15)$$

Proof: Let z be a sequence consisting of partial sequence s, followed by any sequence of aircraft from $R \cup k$. Let z' be the corresponding sequence consisting of partial sequence s', followed by the same sequence of aircraft from $R \cup i$, replacing aircraft k by aircraft i. Given that $\max_{x \in s}(t_x + \delta_{xk}, r_k) \leq \max_{x \in s'}(t'_x + \delta_{xi}, r_i)$ (i.e., if $\delta_{ik} > \delta_{ki}$, it has no influence) and $r_k \leq t'_i$ (k preceeds i in s'), then $t_k \leq t'_i$. Given that $lt_i \leq lt_k$, $t_x \leq t'_x\ \forall x \in R$ and that s is feasible, it then follows that the violation of hard time windows for z no worse than z'. In addition, given that $t_x \leq t'_x\ \forall x \in R$, the cost incurred by any aircraft $x \in R$ is no worse in z than in z'. Also, since $t_k \leq t'_i$ and $t'_i \leq t$ ($t \geq \max_{x \in s'}(t_x + \delta_{xi}, r_i)$), $t_k \leq t$ and $W_1(t - b_k)^{\alpha} + W_2 C(t, lc_k) \geq W_1(t_k - b_k)^{\alpha} + W_2 C(t_k, lc_k)$. Hence, satisfying Inequality 15 is sufficient for satisfying Inequality 16 and implies that the current difference in cost between s and s' is less than any future difference between z and z', and that for any sequence z' there is a sequence z which is no worse. Pruning s' (or any sequence based on it) from the solution space will therefore not compromise optimality. Theorem 10 is thereby a generalisation of Theorem 6, for which the aircraft in s and s' can take any arbitrary order. From a computational perspective however, Theorem 6 allows for much faster implementation, since it only applies to one single sequence, and does not require the retrieval of the set of sequences containing the aircraft in s'.

$$F(s) - F(s') \leq W_1(t'_i - b_i)^{\alpha} + W_2 C(t'_i, lc_i) - W_1(t_k - b_k)^{\alpha} - W_2 C(t_k, lc_k) \quad (16)$$

□

3.7. Dynamic program

3.7.1. Algorithm Outline The pruning rules introduced above were integrated in a dynamic program (DP) that incrementally builds up a sequence by adding one aircraft to the partial sequence at every stage. Our code was implemented following the template provided in Algorithm 1. Each state in stage n represents a partial sequence containing n aircraft that have already been scheduled for take-off. A state is defined by the set of non-separation influencing aircraft, the set of separation influencing aircraft and their take-off times, the objective values (Equation 3), and any constraint violations. States are expanded in a similar way to other dynamic programming approaches previously introduced in the literature (see Section 1), however, any state at any stage that violates any of the complete or conditional orders defined above is pruned here. Dominance with lower bounding, dominance between subsets, and dominance between non-identical sets is applied when comparing states against each other. The additional pruning and improved dominance rules (beyond the normal dynamic programming approach of implicitly pruning sub-optimal paths to achieving the states in the current stage) resolves the state space problem. Each state in the final stage of our DP therefore represents a Pareto-optimal runway sequence consisting of n aircraft. Results are shown in the next section for the application of this method to real departure problems at Heathrow, including an analysis of the contribution of each rule in terms of the reduction in the size of the state space and the runtime of the algorithm. An example of applying these rules to a recent arrival sequencing problem is also provided, to show their effectiveness, and the results contrasted with those from earlier work.

3.7.2. Discussion In contrast to some previous methods where only one aircraft is considered to influence the separations of later aircraft, the dynamic program and the pruning rules that it uses explicitly considers multiple separation influencing aircraft. This implies that the approach is applicable to problem instances where the triangle inequality does not hold (e.g. departures and mixed mode operations) or to multi-runway scenarios where it is necessary to take, at least, the last aircraft on every runway into account. The fact that considering multiple separation influencing aircraft may result in a combinatorial explosion of the state space is addressed by introducing the pruning rules and using more flexible dominance rules.

Algorithm 1 Outline of our pruned dynamic program

```
1: Initialise $previousStateSpace$ with a single state with no aircraft
2: Initialise $currentStateSpace$ to be empty
3: while aircraft remain to be added do
4:   for each state $s$ in $previousStateSpace$ do
5:     for each ordered set of separation identical aircraft $S_i$ (§3.2) do
6:       $a$ = first aircraft in $S_i$ that is not in $s$, $null$ if none are left
7:       if $a$ != null then
8:         if appending $a$ to $s$ will not violate insertion dominance (§3.4) then
9:           if appending $a$ to $s$ will not violate conditional orders (§3.3.1) then
10:            if appending $a$ to $s$ will not violate conditional non-identical orders (§3.3.2) then
11:              expand $s$ by adding $a$, resulting in $sNew$
12:              add $sNew$ to $currentStateSpace$
13:              check for dominance with look-ahead in $currentStateSpace$ (§3.5)
14:            end if
15:          end if
16:        end if
17:      end if
18:    end for
19:  end for
20:  check for subset dominance between $previousStateSpace$ and $currentStateSpace$ (§3.6.1)
21:  check for dominance between non-identical sets in $currentStateSpace$ (§3.6.2)
22:  $previousStateSpace = currentStateSpace$
23:  Initialise $currentStateSpace$ to be empty
24: end while
```

4. Results

4.1. Problem Instances

The performance of our pruned dynamic program is illustrated here using complex real world problem instances, covering three days of departure operations at London Heathrow Airport. The instances were first introduced in Atkin et al. (2012) for the pushback time allocation problem. Their characteristics are summarised in Table 2, which shows the number of sets of separation identical aircraft and the number of aircraft with CTOTs. Each instance contains 55 aircraft, of which the first 5 are assumed to be fixed to give a take-off history.

Day I			Day II			Day III		
Id.	*Sets*	*CTOTs*	*Id.*	*Sets*	*CTOTs*	*Id.*	*Sets*	*CTOTs*
1	26	17	13	23	13	25	33	23
2	17	8	14	27	18	26	24	17
3	14	6	15	25	17	27	30	20
4	16	8	16	24	16	28	26	18
5	21	13	17	25	17	29	29	21
6	20	11	18	20	9	30	23	12
7	13	5	19	12	5	31	24	16
8	17	4	20	15	5	32	29	19
9	20	8	21	8	1	33	18	9
10	12	4	22	9	1	34	15	5
11	13	1	23	10	1	35	12	3
12	9	0	24	9	0	36	15	5

Table 2 Problem instances

The terminal manoeuvering area around London Heathrow is highly complex. It has ten different standard instrument departure routes in normal use at any time. In addition, up to three different speed classes and five different weight classes have to be considered, resulting in 180 different aircraft types (corresponding to N above) and 32400 possible combinations. This results in a large and complex separation matrix in which triangle inequalities are violated (the separation of subsequent aircraft is influenced by the two most recent ones in general, and more on occasion). A detailed description of the separation requirements at Heathrow Airport is provided in Atkin (2008). It is also obvious from Table 2 that there are more than the usual 3 to 6 different separation classes (as discussed in the arrival scheduling literature) to consider.

We start our analysis with a comparison between the runtimes obtained by our approach with those obtained for the approaches introduced by Psaraftis (1980) and Balakrishnan and Chandran (2010). We complete this consideration by applying the pruning rules to an alternative problem which was previously introduced in the literature and illustrate their effectiveness. The focus of the remainder of our analysis is threefold: to evaluate the effectiveness of the principles introduced in §3 that underpin our dynamic program; to see the impact of optimising multiple objectives as opposed to a single objective; to see the potential real world improvements that could be obtained from runway sequencing without constraining an aircraft's maximum positional shift relative to the initial first come first served sequence (i.e., modelling equity as an objective through the non-linear delay penalty only), as opposed to runway sequencing in which the maximum positional shift relative to the aircraft's initial position is constrained (i.e., modelling equity as a hard constraint, in addition to the non-linear delay penalty).

Our pruned DP was implemented in Java and all experiments reported in this section were carried out on a single core of an Intel(R) Core(TM)i7 CPU 950@3.70GHz desktop PC. Our code was allocated 16GB of RAM and executed on Sun's Java(TM) Runtime environment (Version 6), on a Windows 7, 64 bit, Enterprise Edition platform. The following parameter settings were used: $W_1 = 100$, $W_2 = 10$, $\omega_1 = 10$ $\omega_2 = 10000$, $\omega_3 = 100$, $\omega_4 = 1000000$, $\alpha = 1.5$, which were taken from Atkin (2008).

4.2. Comparison with previous approaches

As discussed in Section 1, constrained position shifts have been utilised in the past to reduce the problem complexity by limiting the number of positions that each aircraft can take. The problems of such an approach are illustrated in this section.

The average, maximum, and total runtime for different values of the constrained position shift and for different previous approaches are listed in Table 3. For instance, the third line in Table 3 for constrained position shift 3 represents the results that were obtained when the maximum number of positions that an aircraft could deviate from its position in the initial sequence was restricted to ±3. All code was implemented in a comparable way and common components were shared between the implementations to make the comparison as fair as possible.

The results show that the computational cost for the approaches introduced by Psaraftis (1980) and Balakrishnan and Chandran (2010) increases rapidly as a function of the constrained position shift. This is in line with the expectations based on their theoretical complexity and with the results in the publications themselves, since they were not designed for such complex problems, relying on having a low number of separation groups. In the case of Balakrishnan and Chandran's approach, the runs for a constrained position shift of 10 were terminated when our implementation failed to solve the first instance within 12 hours. This was also the case for Psaraftis' approach when a constrained position shift of 55 was applied (which effectively means that any aircraft could take up any position in the sequence).

	Psaraftis (1980)			Balakrishnan and Chandran (2010)			Pruned DP		
CPS	Avg.	Max.	Total	Avg.	Max.	Total	Avg.	Max.	Total
1	0.001	0.020	0.020	0.003	0.020	0.100	0.004	0.020	0.160
2	0.003	0.020	0.100	0.005	0.030	0.190	0.001	0.020	0.020
3	0.012	0.030	0.440	0.028	0.060	0.990	0.006	0.030	0.230
4	0.039	0.110	1.420	0.150	0.360	5.410	0.013	0.050	0.460
5	0.165	0.690	5.940	0.950	2.820	34.190	0.029	0.080	1.060
6	0.556	1.750	20.010	5.061	9.920	182.180	0.067	0.170	2.410
7	1.931	7.250	69.510	28.288	60.140	1018.380	0.153	0.420	5.500
8	6.333	30.280	227.980	154.940	364.700	5577.830	0.325	1.010	11.690
9	20.555	137.130	739.980	862.696	1575.160	31057.060	0.668	1.870	24.050
10	68.287	588.210	2458.320				1.379	4.600	49.660
55							1.288	8.860	46.350

Table 3 Computational cost in seconds for different dynamic programs, summed for all 36 data instances.

Table 4 lists the results for our approach on the benchmark instances for Milan Airport provided by Furini et al. (2012), containing 60 aircraft each. These instances are freely available from http://www.or.deis.unibo.it/research.html. Furini et al. (2012) do not consider CTOT costs and makespan, and apply a weighted linear delay penalty per aircraft based on its size and fuel consumption. I.e., the value of α in Equation 3 is set to 1 (linear delay cost), the value of W_1 is aircraft dependent and replaced by $W_{1,i}$, and the value of W_2 is set to 0 (which significantly simplifies the problem).

Furini's instances were considerably easier to solve than the instances considered for Heathrow Airport. The results in Table 4 were obtained by enabling complete orders, conditional orders, dominance with lower bounding, and insertion dominance. These rules were able to prune the state space sufficiently without the additional computational burden of checking the other pruning rules, and without the need to add additional complexity to the algorithm. The total runtime to solve all problem instances to optimality was 64 milliseconds, or on average 5.3ms per problem instance. This equates to a speedup by a factor of 37170 relative to the runtimes reported in Furini et al. (2012) (which were an average of 197 seconds for heuristically obtained solutions, although we note that the hardware configuration used by Furini et al. (2012) is different from the hardware configuration used to generate the results in this paper). The results reported above for the problem instances from Heathrow and Milan counter the common belief that many real world runway sequencing problems are too complex to solve exactly, despite the difficulty that many algorithms have had in the past.

Dataset	Start Time	End Time	Objective Value
FPT01	14:55:00	17:32:00	265
FPT02	15:30:00	18:00:00	293
FPT03	15:47:00	18:27:00	255
FPT04	16:14:00	18:47:00	268
FPT05	16:35:00	19:36:00	249
FPT06	14:00:00	16:47:00	167
FPT07	14:32:00	17:08:00	198
FPT08	14:55:00	17:37:00	167
FPT09	15:25:00	18:10:00	183
FPT10	15:55:00	18:45:00	211
FPT11	16:24:00	19:34:00	229
FPT12	16:45:00	20:17:00	207

Table 4 Optimal results for the benchmark instances introduced by Furini et al. (2012).

4.3. Pruning

The results in Table 5 illustrate the efficiency of the pruning rules, by showing the effects of iteratively disabling each of the rules in our full implementation. The number of states that were generated and the runtimes for the full implementation are listed in the second column of Table 5. The numbers listed in the other columns represent the ratio of the number of states generated in the modified implementation versus the number of states in the full implementation. The numbers in parentheses in the other columns represent the ratio of the computation time required by the modified implementation versus the computation time required by the full implementation. The last two rows of Table 5 list the total number of states and the total runtime for our full implementation in the second column, and the aggregated ratios for the total number of states and the total runtime in the other columns. The aim is to determine the benefits of adding each of the pruning rules even when all of the other rules are already present, i.e. what this rule contributes that other rules do not.

		Orders			Dominance rules			
Id.	Full Implementation	Complete	Conditional Identical	Conditional Non-Identical	Insertion	Lower Bounding	Subset	Non-Identical
1	513.00 (0.05)	1.00 (2.80)	1.74 (2.20)	1.00 (1.60)	5.27 (5.00)	6.00 (4.40)	4.95 (2.80)	1.00 (1.20)
2	522.00 (0.05)	1.00 (1.80)	2.18 (1.00)	1.00 (0.60)	3.79 (1.20)	6.70 (2.20)	4.58 (1.60)	1.00 (0.60)
3	25439.00 (0.64)	1.00 (4.41)	1.66 (1.66)	2.25 (2.41)	2.49 (3.22)	17.81 (22.77)	8.26 (5.22)	1.34 (1.31)
4	5621.00 (0.13)	1.00 (3.69)	1.58 (1.54)	1.11 (1.23)	2.68 (3.15)	36.13 (53.62)	11.79 (8.92)	0.95 (1.08)
5	178904.00 (6.97)	1.00 (2.52)	6.12 (11.74)	1.28 (1.27)	1.91 (3.10)	46.76 (86.57)	9.47 (10.14)	1.08 (1.01)
6	243800.00 (8.39)	1.00 (2.64)	9.89 (23.43)	1.00 (0.97)	1.84 (2.98)	49.50 (104.11)	12.95 (18.00)	1.00 (0.98)
7	3461.00 (0.06)	1.00 (6.33)	1.42 (1.83)	1.05 (1.33)	2.59 (3.67)	16.88 (26.83)	5.74 (4.67)	1.00 (1.00)
8	1317.00 (0.05)	1.00 (3.40)	1.01 (0.60)	1.16 (1.00)	4.74 (6.20)	12.49 (9.60)	3.01 (1.20)	1.00 (1.00)
9	442.00 (0.02)	1.00 (2.50)	1.00 (0.00)	1.20 (1.50)	8.08 (8.00)	5.23 (4.00)	28.11 (11.00)	1.00 (1.00)
10	238.00 (0.02)	1.00 (1.00)	1.01 (0.00)	1.07 (0.00)	3.79 (1.00)	4.34 (1.00)	1.35 (0.00)	1.00 (0.00)
11	11254.00 (0.26)	1.00 (4.54)	1.02 (1.00)	1.55 (1.69)	3.71 (6.08)	10.19 (15.88)	5.21 (3.62)	1.03 (1.04)
12	169.00 (0.02)	1.00 (0.00)	1.00 (0.00)	1.01 (0.00)	4.18 (0.00)	4.92 (0.00)	1.66 (0.00)	1.00 (0.00)
13	333.00 (0.00)	1.00 (-)	1.23 (-)	1.12 (-)	4.32 (-)	4.91 (-)	1.41 (-)	1.00 (-)
14	289.00 (0.02)	1.00 (1.00)	1.15 (0.00)	1.01 (0.00)	4.15 (1.50)	3.47 (1.50)	3.34 (1.50)	1.00 (1.00)
15	33864.00 (1.22)	1.00 (2.16)	4.29 (4.48)	1.04 (1.03)	2.72 (4.09)	24.80 (36.30)	8.58 (6.24)	0.70 (0.70)
16	2306.00 (0.08)	1.00 (2.50)	3.45 (3.38)	1.00 (1.00)	3.94 (4.25)	16.67 (18.50)	85.72 (52.13)	1.00 (1.00)
17	421.00 (0.03)	1.00 (1.67)	1.19 (0.67)	1.07 (0.67)	4.55 (2.00)	5.24 (3.00)	5.95 (3.00)	1.02 (0.67)
18	1126.00 (0.03)	1.00 (4.00)	2.51 (3.67)	1.11 (1.67)	4.69 (7.67)	12.60 (13.67)	24.44 (12.33)	1.00 (1.67)
19	22306.00 (0.40)	1.00 (5.58)	4.52 (5.08)	4.06 (4.55)	5.99 (7.53)	59.78 (118.55)	9.94 (8.43)	2.31 (2.35)
20	216842.00 (6.15)	1.00 (2.74)	1.94 (2.20)	1.00 (0.97)	6.82 (16.53)	34.90 (93.20)	13.63 (15.87)	1.00 (0.98)
21	1600.00 (0.00)	1.00 (-)	1.01 (-)	1.23 (-)	2.02 (-)	53.47 (-)	7.62 (-)	1.03 (-)
22	3385.00 (0.05)	1.00 (5.60)	1.00 (1.00)	1.00 (0.60)	2.23 (2.20)	18.83 (26.80)	2.89 (1.20)	1.00 (1.00)
23	23559.00 (0.45)	1.00 (4.58)	1.00 (0.98)	1.00 (1.00)	2.21 (2.91)	91.58 (227.93)	17.51 (13.33)	1.00 (0.98)
24	339.00 (0.02)	1.00 (1.50)	1.00 (1.00)	1.06 (0.00)	3.89 (0.00)	4.10 (0.00)	1.36 (0.00)	1.00 (1.00)
25	379.00 (0.00)	1.00 (-)	1.77 (-)	1.04 (-)	3.71 (-)	3.67 (-)	1.98 (-)	1.00 (-)
26	216.00 (0.00)	1.00 (-)	1.23 (-)	1.00 (-)	4.47 (-)	3.43 (-)	1.63 (-)	1.00 (-)
27	1722.00 (0.05)	1.00 (1.80)	3.14 (3.20)	1.00 (1.00)	4.33 (7.20)	81.44 (122.60)	43.21 (22.40)	1.00 (1.00)
28	1786.00 (0.08)	1.00 (2.13)	1.21 (1.00)	1.01 (0.75)	25.27 (28.25)	4.62 (4.25)	7.37 (4.13)	1.00 (1.00)
29	32192.00 (1.65)	1.00 (2.07)	17.99 (16.78)	1.09 (1.00)	4.27 (5.33)	25.25 (29.16)	70.67 (53.41)	1.00 (0.95)
30	53252.00 (1.42)	1.00 (2.80)	2.73 (3.14)	1.01 (0.96)	1.75 (2.47)	12.38 (23.99)	21.26 (17.89)	1.00 (0.96)
31	72095.00 (3.26)	1.00 (2.45)	6.28 (6.71)	1.22 (1.17)	2.69 (3.81)	26.87 (37.83)	7.46 (4.62)	1.05 (0.99)
32	85494.00 (3.74)	1.00 (1.91)	57.67 (185.70)	1.00 (0.97)	2.29 (3.63)	47.29 (80.80)	43.92 (50.76)	1.00 (0.96)
33	67914.00 (1.64)	1.00 (3.23)	16.19 (40.95)	1.00 (0.96)	3.30 (4.68)	65.70 (177.04)	21.73 (23.88)	1.00 (0.96)
34	294738.00 (8.86)	1.00 (3.28)	3.76 (5.02)	1.08 (1.04)	1.62 (2.22)	17.99 (36.47)	4.45 (3.49)	1.01 (0.98)
35	25116.00 (0.52)	1.00 (3.63)	1.23 (1.23)	1.00 (0.92)	2.58 (3.92)	102.94 (238.04)	24.25 (20.56)	1.00 (0.92)
36	605.00 (0.02)	1.00 (2.50)	1.80 (1.00)	1.02 (1.00)	3.02 (1.50)	4.26 (2.50)	2.84 (0.00)	1.00 (0.00)
Ratios		1.00 (2.78)	8.94 (25.10)	1.14 (1.10)	2.95 (5.05)	37.74 (76.15)	14.54 (16.14)	1.03 (0.99)
Totals	1413559.00 (46.35)	1413559.00 (128.77)	12641532.00 (1163.31)	1615264.00 (51.01)	4167486.00 (233.96)	53351127.00 (3529.50)	20546938.00 (748.02)	1462068.00 (45.84)

Table 5 Number of states (and final runtime, in seconds) for full implementation of the algorithm, compared to the ratio of the number of states (runtime) without the pruning rules. The two last lines lists the ratio and absolute values for the number of states (runtime) across all 36 instances. The last column lists the number of states and the runtime for our full implementation.

It can be observed from Table 5 that the most efficient principles are dominance with lower bounding, subset dominance, and conditional orders between identical aircraft. These reduced the number of states (or runtimes) by a factor of 37.74 (76.15), 14.54 (16.14), and 8.94 (25.10) respectively. They are followed by insertion dominance, conditional orders between non-identical aircraft, and dominance between non-identical sets, which resulted in a factor of 2.95, 1.14, and 1.03 times fewer states being generated, and speed-ups of a factor of 5.05, 1.10, and 0.99, respectively. The table also shows that conditional orders between identical aircraft covers complete orders, as explained in §3.3.1: exactly the same number of states are generated if the respective rule is disabled. However, the inclusion of complete orders still results in a speed up by a factor of 2.78 by reducing the computational burden. The relative ratio of the state space reduction versus the reduction in runtime illustrates that some principles are more "costly" to implement. However, apart from dominance between non-identical sets, the significant reduction in the number of states always outweighed the additional computational cost of adding the pruning rule to the implementation. Our full implementation is able to generate optimal results in an average runtime of 1.29 seconds per dataset, a maximum runtime of 8.86 seconds, and a total runtime of 46.35 seconds across all 36 instances. We note that no parallelisation of our code was used, however preliminary experiments indicate that further reductions in runtime are possible from parallelisation.

The pruning rules evaluated above exploit three key characteristics that are present in real world instances:

- Aircraft arrive over time and cannot depart before they are ready (insertion dominance, dominance with lower bounding)
- Sets of identical aircraft are present (complete orders, conditional orders between identical aircraft)
- The separations are structured (conditional orders between non-identical aircraft and dominance between non-identical sets)

These characteristics are expected to be present in the majority of the real world runway sequencing problems, since they are inherent to the core nature of the problem. In the worst case scenario, where every aircraft belongs to a different weight class, and/or speed class, and/or follows a different SID, or all aircraft have a slot, no complete orders can be inferred, and the pruning exploiting conditional orders between identical and non-identical aircraft will become more important. Similarly, if all aircraft are ready at the same time, insertion dominance and dominance with lower bounding will become less efficient, but the other pruning rules will still apply. However, none of these scenarios are likely to occur in practice.

For instance, insertion dominance and dominance with lower bounding exploit the fact that aircraft "depart over time", and are not all ready at the same time (which is likely to be always the case in a real world instances). If all aircraft are subject to a slot (which is unlikely to happen in practice), complete orders can not be inferred. However, in this case, conditional orders could be inferred in many cases, as discussed in special cases for Theorem 4 in §3.3.

4.4. Objectives

Table 6 lists the average, maximum and total computational cost (in seconds) of our pruned dynamic program across all 36 problem instances for different constrained position shifts (CPS) and different combinations of objectives. If only makespan is optimised, and no CTOTs are modelled, our approach solves the unconstrained problem (i.e. without constrained position shifts) in 0.233 seconds on average, with a maximum runtime of 1.592 seconds, and a total runtime of 8.398 seconds for all 36 instances. These times increase to 0.342, 2.278, and 12.316 seconds, respectively, if CTOTs are modelled as a constraint (i.e. the CTOT start time may delay take-off), but only makespan is optimised (i.e., CTOT violations and delay are not penalised in the objective function, 3^{rd} column). These values gradually increase when CTOT violations or delay are added as an objective (4^{th} and 5^{th} column, respectively). When makespan, CTOT violations, and delay are all included as objectives, the respective values rise to 1.288, 8.860, 46.350 seconds. These results illustrate the increasing challenges when considering multiple objectives. We note that the computation times for CPS 10 are slightly larger than those for the unconstrained problem. This is due to the fact that the pruning rules need to account for constrained position shifts, which reduces their efficiency/applicability in some cases.

CPS	*No CTOTs*			*CTOTs*											
	Makespan			*Makespan*			*Makespan, TW*			*Makespan, Delay*			*Makespan, TW, Delay*		
1	0.000	0.015	0.030	0.001	0.016	0.062	0.000	0.016	0.016	0.002	0.016	0.079	0.003	0.016	0.124
2	0.003	0.031	0.109	0.000	0.015	0.015	0.000	0.016	0.031	0.000	0.016	0.031	0.000	0.016	0.016
3	0.002	0.016	0.094	0.002	0.016	0.077	0.001	0.016	0.062	0.003	0.016	0.110	0.005	0.031	0.188
4	0.003	0.016	0.111	0.007	0.031	0.252	0.008	0.032	0.313	0.006	0.016	0.235	0.010	0.046	0.374
5	0.010	0.031	0.389	0.013	0.031	0.499	0.022	0.078	0.827	0.019	0.047	0.703	0.027	0.078	0.998
6	0.022	0.062	0.798	0.033	0.079	1.217	0.052	0.140	1.890	0.042	0.109	1.513	0.066	0.172	2.388
7	0.045	0.140	1.638	0.070	0.172	2.545	0.114	0.312	4.122	0.090	0.203	3.265	0.152	0.421	5.495
8	0.088	0.312	3.169	0.139	0.374	5.012	0.247	0.764	8.899	0.179	0.437	6.464	0.324	1.014	11.695
9	0.160	0.686	5.762	0.266	0.857	9.584	0.520	1.545	18.731	0.337	0.920	12.160	0.668	1.873	24.054
10	0.290	1.389	10.462	0.489	1.918	17.605	1.090	3.807	39.249	0.608	2.029	21.916	1.379	4.603	49.673
55	0.233	1.592	8.398	0.342	2.278	12.316	0.732	4.900	26.356	0.620	4.477	22.331	1.288	8.860	46.350

Table 6 Computational cost in seconds (average, maximum, total) as a function of the constrained position shift for different objective function configurations.

4.5. Impact

The results in Table 7 illustrate the real world benefits that could be obtained for makespan (runway utilisation), delay, and CTOT compliance when solving the departure sequencing problem without constrained position shifts. The values in Table 7 show the increase in makespan and delay (in minutes), and the absolute increase in the number of CTOT violations and their extensions for solutions obtained with a constrained position shift, relative to unconstrained solutions. For example, the results illustrate that for a constrained position shift of 3 and with the given preferences between delay, equity of delay and CTOT compliance, i.e. the values of W_1, W_2, α, ω_1, ω_2, ω_3, ω_4 listed in §4.1, the total increase in makespan across all 36 instances is 157 minutes, 7103 additional delay minutes are incurred, 46 additional CTOTs are missed, and 21 additional violations of CTOT extensions are accumulated. Even when the constrained position shift is equal to 10, the increase in makespan, delay, and CTOT violations is still significant. Thus, the results show that imposing equity as a hard constraint through constrained position shifts rather than as an objective has a detrimental effect on the quality of the resulting sequences for the problem instances considered here.

CPS	ΔMakespan	ΔDelay	ΔTW Violations	ΔTW Extension Violations	Avg. Shift	Max. Shift
0	718	23847	185	135	0.00	0
1	373	14218	120	68	0.57	1
2	232	9884	77	38	1.05	2
3	157	7103	46	21	1.42	3
4	112	5413	31	16	1.69	4
5	92	4236	19	11	1.92	5
6	71	3323	14	10	1.99	6
7	64	2703	9	7	2.10	7
8	48	2208	6	6	2.15	8
9	44	1848	6	6	2.18	9
10	35	1514	4	4	2.27	10

Table 7 Incremental difference between optimal unconstrained solutions and with a constrained position shift, summed over all instances

5. Conclusions

This paper introduced a number of highly effective pruning rules for optimal unconstrained runway sequencing (i.e. without imposing constrained position shifts). They were shown to make otherwise intractable runway sequencing problems with complex separation constraints, time-windows, and multiple non-linear objectives tractable. In addition, they can be applied to different types of runway sequencing problems, including segregated and mixed mode operations on a single runway or multiple runways and pre-determined runway allocation. The pruning rules presented in this paper are expected to be particularly beneficial in these scenarios.

The importance of the work in the context of airport operations is underlined by the results that were reported for real world sequencing problems from London Heathrow. They show that significant improvements in runway utilisation, delay, equity of delay, and slot compliance can be obtained, whilst maintaining a careful balance between these objectives. The importance of this work is also illustrated by the fact that we are currently building upon it in our work on multi-runway sequencing with en-route dependencies (considering the interactions between multiple departure and/or arrival routes) and our work in integrated airport operations (in particular the integration of ground movement and runway sequencing). Within this context we would like to refer to our recent work on Target Start-At Time allocation (TSAT) at London Heathrow Atkin et al. (2012). The TSAT generator is used to allocate pushback times to aircraft and to negotiate optimal calculated times of take-off (CTOTs or slots). This enables the operations at the airport to drive the network rather than the other way round. Runway sequencing forms an indispensable element of the TSAT system and we hope to be able to incorporate those ideas into that system.

The broader impact of the work reported here is illustrated by the fact that our pruning rules are applicable to different exact and heuristic approaches. In addition, they are applicable to different types of machine scheduling problems with sequence dependent setup-times. They can therefore be expected to have a significant impact within the broader field of airport operations and machine scheduling.

Acknowledgments

Part of this work was carried out under under EPSRC grants EP/F033214/1, EP/H000968/1, and EP/H004424/2. The authors would like to thank National Air Traffic Services for their continued support. We would also like to thank the editor in chief, the associate editor and anonymous reviewers for their detailed and constructive feedback.

References

Allahverdi, A., J.N.D. Gupta, T. Aldowaisan. 1999. A review of scheduling research involving setup considerations. *Omega* **27**(2) 219 – 239.

Allahverdi, A., C.T. Ng, T.C.E. Cheng, M.Y. Kovalyov. 2008. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* **187**(3) 985 – 1032.

Atkin, J.A.D. 2008. Online decision support for take-off runway scheduling at London Heathrow airport. Ph.D. thesis, School of Computer Science, University of Nottingham, Nottingham, UK.

Atkin, J.A.D. 2013. Airport airside optimisation problems. A.S. Etaner-Uyar, E. Ozcan, N. Urquhard, eds., *Automated Scheduling and Planning, From Theory to Practice*, chap. 1. Studies in Computational Intelligence Volume 505, Springer, 1 – 37.

Atkin, J.A.D., E.K. Burke, J.S. Greenwood. 2010. TSAT allocation at London Heathrow: the relationship between slot compliance, throughput and equity. *Public Transport* **2** 173 – 198.

Atkin, J.A.D., E.K. Burke, J.S. Greenwood, D. Reeson. 2007. Hybrid metaheuristics to aid runway scheduling at London Heathrow airport. *Transportation Science* **41** 90 – 106.

Atkin, J.A.D., G. De Maere, E.K. Burke, J.S. Greenwood. 2012. Addressing the pushback time allocation problem at Heathrow airport. *Transportation Science* **47**(4) 584 – 602.

Balakrishnan, H., B. Chandran. 2010. Algorithms for scheduling runway operations under constrained position shifting. *Operations Research* **58** 1650 – 1665.

Beasley, J.E., M. Krishnamoorthy, Y.M. Sharaiha, D. Abramson. 2000. Scheduling aircraft landings - the static case. *Transportation Science* **34** 180 – 197.

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

Bianco, L., P. Dell'Olmo, S. Giordani. 1999. Minimizing total completion time subject to release dates and sequence-dependent processing times. *Annals of Operations Research* **86** 393 – 415.

Dear, R.G. 1976. The dynamic scheduling of aircraft in the near terminal area. Tech. rep., Flight Transportation Laboratory, MIT, Cambridge, Massachusets, U.S.

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

Eurocontrol. 2013. Challenges of growth 2013. Tech. rep., Eurocontrol.

Furini, F., C. Persiani, P. Toth. 2012. Aircraft sequencing problems via a rolling horizon algorithm. A. Mahjoub, V. Markakis, I. Milis, V. Paschos, eds., *Combinatorial Optimization*, *Lecture Notes in Computer Science*, vol. 7422. Springer Berlin / Heidelberg, 273–284.

Luo, X., C. Chu. 2007. A branch-and-bound algorithm of the single machine schedule with sequence-dependent setup times for minimizing maximum tardiness. *European Journal of Operational Research* **180**(1) 68 – 81. doi:http://dx.doi.org/10.1016/j.ejor.2005.06.069.

Luo, X., F. Chu. 2006. A branch and bound algorithm of the single machine schedule with sequence dependent setup times for minimizing total tardiness. *Applied Mathematics and Computation* **183**(1) 575 – 588.

Luo, X., X. Liu, C. Wang, Z. Liu. 2005. Dominance rules for single machine schedule with sequence dependent setup and due date. *Journal of Control Theory and Applications* **3**(4) 364 – 370. doi: 10.1007/s11768-005-0025-2.

Panwalkar, S.S., W. Iskander. 1977. A survey of scheduling rules. *Operations research* **25** 45–61.

Pinedo, M. 2002. *Scheduling. Theory, Algorithms and Systems*. 2nd ed. Prentice-Hall inc., Upper Saddle, River, New Jersey.

Psaraftis, H.N. 1980. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research* **28** 1347 – 1359.

Ragatz, G.L. 1993. A branch and bound method for minimum tardiness sequencing on a single processor with sequence dependent setup times. *Proceedings of the 24th Annual Meeting of the Decision Sciences Institute*. 1375 – 1377.

Sewell, E.C., J.J. Sauppe, D.R. Morrison, S.H. Jacobson, G.K. Kao. 2012. A BB&R algorithm for minimizing total tardiness on a single machine with sequence dependent setup times. *Journal of Global Optimization* **54**(4) 791 – 812. doi:10.1007/s10898-011-9793-z.

Sourd, F. 2005. Earliness – tardiness scheduling with setup considerations. *Computers & Operations Research* **32**(7) 1849 – 1865. doi:http://dx.doi.org/10.1016/j.cor.2003.12.002.