OPERATIONS RESEARCH
Vol. 58, No. 6, November–December 2010, pp. 1650–1665
ISSN 0030-364X | EISSN 1526-5463 | 10 | 5806 | 1650
DOI 10.1287/opre.1100.0869
© 2010 INFORMS

# Algorithms for Scheduling Runway Operations Under Constrained Position Shifting

**Hamsa Balakrishnan**
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, hamsa@mit.edu

**Bala G. Chandran**
Analytics Operations Engineering Inc., Boston, Massachusetts 02109, bchandran@nltx.com

The efficient operation of airports, and runways in particular, is critical to the throughput of the air transportation system as a whole. Scheduling arrivals and departures at runways is a complex problem that needs to address diverse and often competing considerations of efficiency, safety, and equity among airlines. One approach to runway scheduling that arises from operational and fairness considerations is that of constrained position shifting (CPS), which requires that an aircraft's position in the optimized sequence not deviate significantly from its position in the first-come-first-served sequence. This paper presents a class of scalable dynamic programming algorithms for runway scheduling under constrained position shifting and other system constraints. The results from a prototype implementation, which is fast enough to be used in real time, are also presented.

*Subject classifications*: transportation: runway scheduling under constrained position shifting; dynamic programming/optimal control: deterministic polynomial-time scheduling algorithms.
*Area of review*: Transportation.
*History*: Received September 2008; revisions received January 2009, July 2009, November 2009; accepted February 2010.

## 1. Introduction

The air transportation system in the United States is a tightly constrained system that is operating at (or close to) capacity at most major airports. In 2005, terminal-area congestion accounted for only 13% of all delays at the 35 busiest airports; that number had risen to 17% in 2008, and is currently at 21% over the first nine months of 2009 (Federal Aviation Administration 2009). The increasing delays, coupled with the expected increase in the demand for air transportation in the future, have motivated several initiatives, both in the United States and in Europe, for the enhancement of terminal-area capacities (Arkind 2004, Boehme 1994). The runway system has been identified as the primary bottleneck in airport capacity, due to various operational constraints on runway operations (Idris et al. 1998). Consequently, even small enhancements to runway throughput can have a significant impact on systemwide delays.

The terminal area is a dynamic and uncertain environment, with constant updates to aircraft states being obtained from surveillance systems and airline reports (Atkins and Brinton 2002). The dynamic nature of the terminal area necessitates the development of scheduling algorithms that are computationally efficient and therefore amenable to replanning when new events occur, such as when a new aircraft enters the center boundary or when data updates are obtained. The challenge lies in simultaneously achieving safety, efficiency, and equity, which are often competing objectives, and doing so in a reasonable amount of time (Böhme 2005, Carr 2004, Anagnostakis et al. 2000). There is broad consensus on how to independently model safety, efficiency, and equity: safety is achieved by maintaining separation between aircraft and by satisfying downstream metering constraints; efficiency is equivalent to achieving high throughput and/or low average delay; and equity is modeled by limiting the deviation from a nominal order or by minimizing variance in delay. However, few solution approaches have been able to simultaneously model all three components and optimally solve the runway scheduling problem in a computationally tractable manner. One reason for this computational hurdle is that most runway-scheduling models are, from a theoretical perspective, inherently hard to solve (Beasley et al. 2000). Consequently, most practical implementations resort to heuristic or approximate approaches that produce "good" solutions in a short time (Böhme 2005, Anagnostakis et al. 2001). The difficulty in solving these scheduling models arises primarily because the solution space allows for the optimal sequence to deviate arbitrarily from the first-come-first-served (FCFS) sequence.

Dear (1976) recognized that in the short term it is unrealistic to allow arbitrary deviations from the FCFS sequence for two reasons: (i) the system affords controllers limited flexibility in reordering aircraft, and (ii) large deviations from a nominal schedule may be unacceptable to airlines from a fairness standpoint. This observation led to the constrained position shifting (CPS) approach for scheduling aircraft, which stipulates that an aircraft may be moved up to a specified maximum number of positions from its FCFS order. For example, if the maximum position shift (MPS) allowed was 2, an aircraft that is in the 8th position in the FCFS sequence can be placed at the 6th, 7th, 8th, 9th, or 10th position in the new sequence. Several researchers in both the United States and Europe have used CPS to model fairness and have worked toward developing fast solution techniques for scheduling within the CPS framework (Psaraftis 1980, Dear and Sherif 1991, Neuman and Erzberger 1991, Trivizas 1998, de Neufville and Odoni 2003, Carr 2004).

Psaraftis (1980) was the first to develop a polynomial-time algorithm for scheduling under CPS. His algorithm exploited the fact that the number of different types of aircraft is typically small (small, large, heavy, etc.) and had a complexity of $O(N^2(n/N+1)^N)$, where $n$ is the number of aircraft and $N$ is the number of distinct aircraft types. This algorithm relied on all aircraft of the same type being identical, which did not accommodate time-window restrictions on aircraft or precedence relationships among aircraft, thus effectively scheduling all aircraft of a certain type in FCFS order. Trivizas (1998) proposed a search-based algorithm with a complexity of $O(n2^k)$, where $n$ is the number of aircraft and $k$ is the maximum shift parameter; however, achieving this complexity required a very sophisticated implementation using up to $2^k$ parallel processors. Further, his model also failed to account for time-window restrictions and precedence constraints. The difficulty of incorporating all operational constraints within a CPS framework even led to a conjecture by Carr (2004) that in general, runway scheduling under CPS had exponential complexity.

This paper presents new algorithms for efficient runway scheduling on a single runway with CPS constraints, while accounting for various operational considerations (including time-window restrictions and precedence constraints, which had not been modeled by previous approaches). For reasons discussed in §7.3, the multiple runway case is beyond the scope of this paper. Our key contribution in this paper is to cast the scheduling problem on a graph, referred to as the CPS network and described in §3, whose size is polynomial in the number of aircraft. The scheduling problems are then solved using dynamic programming on this network.

The core problem we consider is that of maximizing runway throughput (equivalent to minimizing the makespan or the landing time of the last of a given set of aircraft) for arrivals-only or departures-only operations. This scenario is of practical importance because many major airports, such as Atlanta, Dallas/Fort Worth, Denver, and New York LaGuardia, use dedicated arrival and departure runways, especially during periods of heavy demand (Federal Aviation Administration 2004). Our algorithm to solve this problem, presented in §4, has a complexity of $O(n(2k+1)^{(2k+2)})$, where $n$ is the number of aircraft and $k$ is the maximum shift parameter. Thus, the complexity of the algorithm is essentially linear in the number of aircraft because $k$ is typically a small constant (1, 2, or 3). In §5, we extend the algorithm to two other objective functions—namely, minimizing the maximum delay over all aircraft and minimizing the average delay—and show that these can also be solved with a complexity that is polynomial in $n$ and exponential in $k$, although our algorithm for the latter problem is unable to account for time-window constraints.

To handle more complex extensions, we introduce the discrete-time CPS network in §6. This network, whose size is dependent on the number of time periods being considered, allows us to develop pseudopolynomial algorithms to minimize the weighted average of delay given arbitrary aircraft-dependent cost structures. In §7, we present algorithms for the problem of mixed operations (simultaneous arrival and departure scheduling on a single runway). One of the algorithms, which solves a realistic problem of merging several departure queues and an arrival queue, has a complexity that is polynomial in the number of aircraft and exponential in the number of departure queues. Finally, we describe a prototype implementation of our algorithm for minimizing makespan in §8.

This paper is of significance because it presents the first class of algorithms that are able to handle commonly encountered operational constraints and objectives within the CPS framework while being computationally scalable.

## 2. Problem Definition

The runway scheduling problem is to find a sequence and corresponding arrival/departure times that optimize some objective of the schedule (for example, minimize the makespan or minimize a weighted average of aircraft delay), subject to the following constraints.

**1. Fairness: position shift constraints.** Because airlines are major stakeholders in the air transportation system, it is important that an increase in efficiency is not achieved at the expense of an equitable allocation of resources. This could happen if an aircraft that would have had an early arrival or departure in the FCFS sequence is rescheduled to operate last, thereby incurring a disproportionate amount of delay. CPS ensures some degree of fairness because it does not allow the final sequence to deviate significantly from the FCFS order. The maximum number of position shifts allowed is denoted by $k$, and the resultant scenario is referred to as a $k$-CPS scenario. Typically, $k$ for both arrival and departure scheduling is between 1 and 3 (de Neufville and Odoni 2003).

**2. Minimum spacing requirements.** An aircraft operating on a runway faces the risk of instability if it interacts with the wake-vortex of an aircraft landing or taking off before it. To prevent this, the Federal Aviation

Administration (FAA) mandates minimum spacing requirements under instrument approach conditions (IAC) between aircraft operations on a runway, which depend on the on the maximum takeoff weight capacity of the aircraft (Federal Aviation Administration 2006). Although these spacing requirements are specified in terms of distance, they can be converted to time requirements assuming a 5 nmi final approach path (de Neufville and Odoni 2003). Representative values for these separations for three weight classes—small, large, and heavy—are listed in Table 1 (for simplicity, we ignore separation requirements for B757 aircraft). We denote the minimum time required between leading aircraft $a$ and trailing aircraft $b$ by $\delta_{ab}$.

Note that the wake-vortex separation requirements for arrivals-only or departures-only operations satisfy the *triangle inequality*, that is, $\delta_{ac} \leqslant \delta_{ab} + \delta_{bc}$, $\forall\, a, b, c$. In addition, these separation requirements satisfy all higher-order polygon inequalities (for instance, the quadrilateral inequality in which $\delta_{ad} \leqslant \delta_{ab} + \delta_{bc} + \delta_{cd}$, $\forall\, a, b, c, d$). As a result, ensuring that spacing requirements are met between successive aircraft ensures that the spacing requirements are met for all pairs of aircraft. This property will be exploited in subsequent sections when developing algorithms for the arrivals-only or departures-only case. In §6.2, we describe algorithmic modifications that allow us to solve the scheduling problem even when the triangle inequality is violated.

**3. Time-window constraints.** Limits on the levels of delay that can be incurred by an aircraft due to downstream traffic flow management initiatives or constraints on possible maneuvers that can be performed by the aircraft restrict the times at which an aircraft can reach a runway (Carr 2004). These constraints could possibly result in a set of disjoint time intervals in which an aircraft can arrive/depart. For simplicity of notation, we describe the case of a continuous-time interval defined by an earliest and latest time, but our approach is applicable to disjoint intervals as well.

**4. Precedence constraints.** Precedence constraints are pairwise requirements on aircraft that stipulate whether one aircraft must land before another. Sources of such constraints are the airlines themselves, which have precedence constraints due to banking operations or priority flights. In addition, arrivals on the same jet route are constrained to not overtake each other. Precedence constraints can also represent the restricted freedom available to taxiing departures that are not allowed to overtake each other (Carr 2004).

**Table 1.** Minimum separation (in seconds) between operations on the same runway (Lee 2008).

| | Trailing | | | | | |
|---|---|---|---|---|---|---|
| | Arrivals | | | Departures | | |
| Leading | Heavy | Large | Small | Heavy | Large | Small |
| Arrivals | | | | | | |
| Heavy | 96 | 157 | 196 | 75 | 75 | 75 |
| Large | 60 | 69 | 131 | 75 | 75 | 75 |
| Small | 60 | 69 | 82 | 75 | 75 | 75 |
| Departures | | | | | | |
| Heavy | 60 | 60 | 60 | 90 | 120 | 120 |
| Large | 60 | 60 | 60 | 60 | 60 | 60 |
| Small | 60 | 60 | 60 | 60 | 60 | 60 |

## 3. The CPS Network

Our key contribution is to cast the scheduling problem on a directed acyclic graph in which every feasible CPS sequence is represented by a path in the network; the scheduling problem is then solved using dynamic programming.

For simplicity, we assume that the aircraft are labeled $(1, 2, \ldots, n)$ according to their position in the FCFS sequence. The network consists of $n$ *stages* $\{1, \ldots, n\}$, where each stage corresponds to an aircraft position in the final sequence. A node in stage $p$ of the network represents a subsequence of aircraft of length $\min\{2k+1, p\}$. For example, for $n = 6$ and $k = 1$, the nodes in stages $3, \ldots, 6$ represent all possible sequences of length $2k+1 = 3$ ending at that stage. Stage 2 contains a node for every possible aircraft sequence of length 2 ending at position 2, whereas stage 1 contains a node for every possible sequence of length 1 starting at position 1. This network, shown in Figure 1, is obtained by finding all sequence combinations of possible aircraft assignments to each position in the sequence. We refer to the last aircraft in a node's sequence as the *final aircraft* of that node.

We then introduce two nodes—a source and a sink—that represent the beginning and end of the sequencing process, respectively. We add arcs from the source to each node in stage 1 and from each node in stage $n$ to the sink. An arc $(i, j)$ is drawn from a node in stage $p$ to one in stage $p+1$ if the aircraft subsequence of node $j$ can follow that of node $i$, i.e., the first $\min\{2k, p\}$ aircraft of node $j$'s subsequence are the same as the last $\min\{2k, p\}$ aircraft of node $i$'s subsequence. For example, a sequence (1–2–3) in stage 3 can be followed by the sequences (2–3–4) or (2–3–5) in stage 4. This results in a network where every directed path from a node in stage 1 to one in stage $n$ represents a possible $k$-CPS sequence. For example, the path (2) → (2–1) → (2–1–3) → (1–3–4) → (3–4–6) → (4–6–5) represents the sequence 2–1–3–4–6–5.

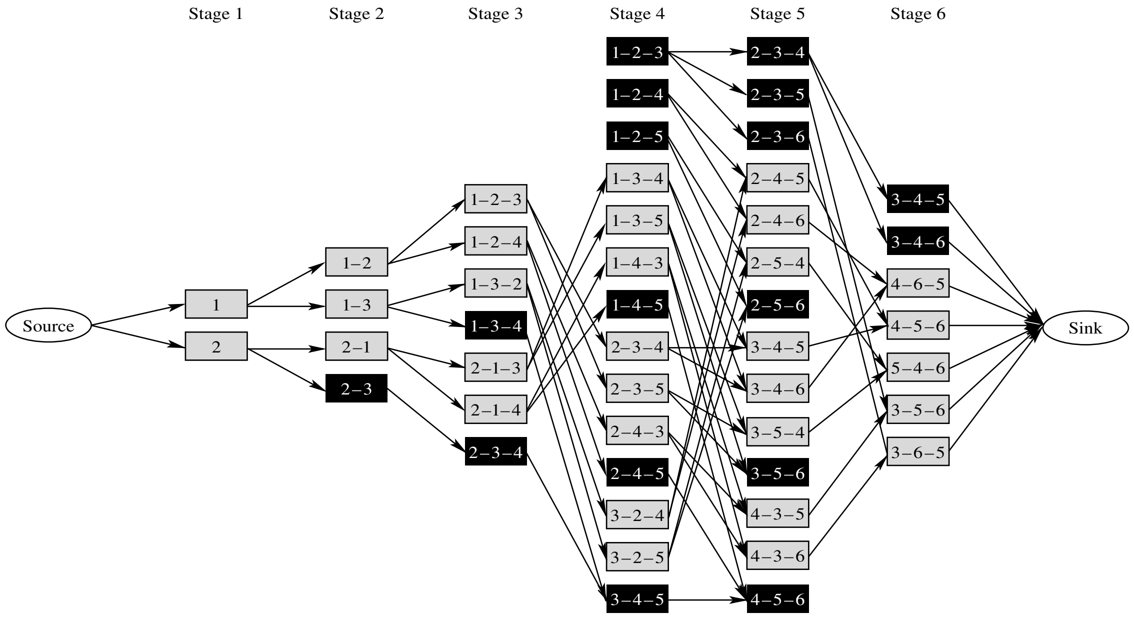
Theorem 1. *A $k$-CPS sequence exists if and only if there exists a corresponding source-sink path in the network.*

Proof. By enumerative construction, every $k$-CPS sequence is a source-sink path in the network. We now prove that every source-sink path in the network is a $k$-CPS sequence.

First, we observe that every source-sink path will consist of exactly $n$ nodes (excluding the source and sink) because each arc in the path moves forward by one stage. Given a path in the network, the corresponding aircraft sequence

**Figure 1.** CPS network for $n = 6$, $k = 1$ generated from possible aircraft assignments shown on top.

| Position | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Possible | 1 | 1 | 2 | 3 | 4 | 5 |
| aircraft | 2 | 2 | 3 | 4 | 5 | 6 |
| assignments | | 3 | 4 | 5 | 6 | |



*Note.* Nodes shaded in black do not belong to any source-sink path and hence can be pruned from the network.

is obtained by taking the final aircraft of a node belonging to stage $p$ and assigning it to position $p$ in the sequence. Thus, position 1 in the sequence is the final aircraft of the first node in the path, position 2 in the sequence is the final aircraft of the second node in the path, and so on. Because there are $n$ aircraft and the path is of length $n$, this procedure will yield a feasible sequence as long as we assign a unique aircraft to each position, i.e., as long as the final aircraft of each node in the path is different.

We now prove this result by contradiction. Suppose there exists a source-sink path in the network containing two nodes with the same final aircraft. Let one of these nodes be in stage $p_1$ and the other be in stage $p_2$ where $p_1 < p_2$. Then, the aircraft appears in position $p_1$ and $p_2$ in the sequence represented by the path. The network is constructed using the fact that any aircraft can appear in at most $2k + 1$ positions, so $p_2$ is within $2k + 1$ positions of $p_1$. However, every subsequence of length $2k + 1$ or less is captured in some node along the path, so there exists a node in the path in whose subsequence the same aircraft appears in more than one position, which is not allowed to occur while generating the network. This contradiction implies that there cannot exist a source-sink path containing two nodes with the same final aircraft.

Therefore, any source-sink path represents a sequence of $n$ distinct aircraft, where each aircraft appears in a position that is one of at most $2k + 1$ possible position assignments for it. □

There are some nodes in the network that cannot be part of any path from the source to the sink (shown shaded in Figure 1) and can hence be eliminated to reduce the network size. The process of pruning the network involves testing whether each node is reachable from both the source and the sink (using a forward and reverse graph search) and eliminating nodes that are not reachable either from the source or the sink.

## 3.1. Incorporating Precedence Constraints

We now describe how precedence constraints can be incorporated into the network. Let $f(a)$ be the position of aircraft $a$ in the final (optimal makespan) sequence and $f(b)$ be the position of aircraft $b$ in the final sequence. For two aircraft $a$ and $b$ such that $a < b$, the precedence constraint can either require that (i) $f(a) < f(b)$ or (ii) $f(a) > f(b)$.

*Case* I: The precedence constraint requires that $f(a) < f(b)$.

Lemma 1. *Suppose there exists a path in the network such that $f(a) > f(b)$. Then, the path contains at least one node such that $b$ appears before $a$ in that node's subsequence.*

Proof. Because each aircraft can shift at most $k$ positions, $f(a) \leqslant a + k$ and $f(b) \geqslant b - k$. Therefore, $f(a) - f(b) \leqslant a - b + 2k \leqslant 2k - 1$ (because $a - b \leqslant -1$ given that $a < b$). Because $f(a)$ and $f(b)$ are within $2k + 1$ of each other and every sequence of length at most $2k + 1$ is contained in some node along the path, the path contains some node whose sequence has $f(a) > f(b)$ and violates the precedence constraint. □

The above lemma shows that removing nodes that violate precedence constraints is not only necessary but also sufficient for eliminating all source-sink paths that violate precedence constraints. This yields the following procedure: in the presence of precedence constraints where $a < b$ and we require $f(a) < f(b)$, remove all nodes that violate the precedence constraints and solve the problem on the resulting network.

*Case* II: The precedence constraint requires that $f(a) > f(b)$.

Because the earliest position that $b$ can land is $b - k$, the earliest time that $a$ can land given that it lands after $b$ is $b - k + 1$. Similarly, the latest position that $a$ can land is $a + k$, meaning that the latest time that $b$ can land is $a + k - 1$. Any node that contains a sequence that violates these two constraints—i.e., has $a$ in a position that is less than $b - k$ or has $b$ in a position greater than $a + k$—should be removed from the network because they cannot belong to a feasible path. We refer to the network obtained after removing such nodes as the *position-constrained* network.

Lemma 2. *Suppose there exists a path in the position-constrained network such that $f(a) < f(b)$. Then, the path contains at least one node such that a appears before b in that node's subsequence.*

Proof. In the position-constrained network, $b - k + 1 \leqslant f(a) \leqslant a + k$ and $b - k \leqslant f(b) \leqslant a + k - 1$. Therefore, $f(b) - f(a) \leqslant a - b + 2k - 2 \leqslant 2k - 3$ (because $a - b \leqslant -1$ given that $a < b$). Because $f(a)$ and $f(b)$ are within $2k + 1$ of each other and every sequence of length at most $2k + 1$ is contained in some node along the path, the path contains some node whose sequence has $f(a) < f(b)$ and violates the precedence constraint. □

This yields the following procedure: in the presence of precedence constraints where $a < b$ and we require $f(a) > f(b)$, we first remove all nodes where the position constraint is violated, giving the position-constrained network. Then, we remove all nodes that violate the precedence constraints and solve the problem on the resulting network.

### 3.2. Asymmetric Shift Constraints

One extension that is easily handled using the described framework is that of asymmetric shift constraints, i.e., the number of allowed forward shifts is different from the number of backward shifts. Let the maximum forward and backward shift parameters be denoted by $k^+$ and $k^-$, respectively. In order to model the asymmetric case, we first create a CPS network with parameter $k = \max\{k^+, k^-\}$. Then, we remove all nodes that violate the position shift constraints and prune the network as before. For example, suppose aircraft 4 in the example in §3 has $k^+ = 1$ and $k^- = 0$. The CPS network is created as before in Figure 1 using $k = 1$. Then, nodes (1–2–4), (1–3–4), (2–1–4), and (2–3–4) are eliminated from stage 3 (because aircraft 4 is not allowed to land in position 3), and the network is pruned as before.

## 4. Minimizing the Makespan

We now present an algorithm to minimize makespan using dynamic programming on the described CPS network. Each arc $(i, j)$ in the network is associated with a distance equal to the minimum separation between the final aircraft of node $i$ and that of node $j$, if they were to take off consecutively and in that order. Arcs that lead into the sink and out of the source have zero distance associated with them. The notation used in our algorithm is listed in Table 2.

### 4.1. Dynamic Programming Recursion

Lemma 3. *The values of $T^*(\cdot)$ can be computed by the following dynamic programming recursion*:

$$T^*(j) = \max\left\{e(j), \min_{i \in P(j):\, T^*(i) \leqslant l(i)} (T^*(i) + \delta_i(j))\right\}. \tag{1}$$

Proof. $T^*(j) \geqslant e(j)$ because a sequence cannot start before the earliest arrival time of the final aircraft in the sequence.

If $T^*(i) > l(i)$ for some predecessor node $i$, it implies that the sequence of node $i$ cannot possibly start within the allowable time window $[e(i), l(i)]$, and hence this node cannot be part of any feasible source-sink path sequence. Therefore, all nodes with $T^*(i) > l(i)$ can be ignored while finding the arrival time of node $i$.

It is not possible for $T^*(j)$ to be strictly less than $T^*(i) + \delta_i(j)$ for all predecessors $i$ because that would violate the separation requirement between the final aircraft of $i$ and $j$. Therefore, $T^*(j) \geqslant T^*(h) + \delta_h(j)$ for at least one of the predecessors $h \in P(j)$: $T^*(h) \leqslant l(h)$. Because $T^*(j) \geqslant T^*(h) + \delta_h(j)$ for at least one feasible predecessor $h$, it is certainly greater than the minimum of $T^*(i) + \delta_i(j)$ over all feasible predecessors $i$.

We have shown so far that $T^*(j) \geqslant e(j)$ and $T^*(j) \geqslant \min_{i \in P(j):\, T^*(i) \leqslant l(i)} (T^*(i) + \delta_i(j))$. To complete the proof, we

**Table 2.** Notation used in the algorithm for merging departure queues with an arrival stream.

| | |
|---|---|
| $T(i)$ | Arrival time of the final aircraft of node $i$. |
| $T^*(i)$ | Arrival time of the final aircraft of node $i$ in an optimal solution. |
| $e(i)$ | Earliest possible arrival time of the final aircraft of node $i$. |
| $l(i)$ | Latest possible arrival time of the final aircraft of node $i$. |
| $\delta_i(j)$ | Minimum separation between the final aircraft of node $i$ (leading) and node $j$ (trailing). |
| $P(i)$ | Set of nodes that are predecessors of node $i$, i.e., there exists an arc from every node in $P(i)$ to node $i$. (Note that $P(\text{source}) = \varnothing$.) |

have to show that at least one of the above inequalities holds as an equality so that

$$T^*(j) = \max\Big\{e(j), \min_{i\in P(j):\ T^*(i)\leqslant l(i)} \big(T^*(i)+\delta_i(j)\big)\Big\}.$$

We now prove the rest by contradiction:

Suppose $T^*(j) > e(j)$ and $T^*(j) > \min_{i\in P(j): T^*(i)\leqslant l(i)} \cdot (T^*(i)+\delta_i(j))$. Then, it is possible to reduce the value of $T^*(j)$ by some sufficiently small quantity while still maintaining a feasible solution, which contradicts the optimality of $T^*(j)$.

Therefore, $T^*(j) \geqslant e(j)$ and $T^*(j) \geqslant \min_{i\in P(j): T^*(i)\leqslant l(i)} \cdot (T^*(i)+\delta_i(j))$ and at least one of the two inequalities holds as an equality. □

The minimum makespan solution is obtained by applying the boundary condition $T^*(\cdot) = e(\cdot)$ for all nodes in stage 1, and successively computing $T^*(\cdot)$ for each node in stage 2 and above by traversing the network from left to right. The minimum makespan is the lowest value of $T^*(\cdot)$ among all nodes in stage $n$. The optimal sequence corresponding to the minimum makespan solution is obtained by keeping track of the predecessor of node $i$ in the sequence as $\arg\min_{i\in P(j): T^*(i)\leqslant l(i)}\{T^*(i)+\delta_i(j)\}$, breaking ties arbitrarily. The pseudocode for the algorithm is given in Figure 2.

This algorithm can also be used for checking whether or not a feasible solution exists: initially set $T(j)=\infty$ for all nodes in stage $n$, then solve for the minimum makespan. If $T^*(j)=\infty$ for all nodes $j$ in stage $n$, then no feasible solution exists; otherwise, the algorithm has found a solution.

### 4.2. Complexity

Proposition 1. *The complexity of the algorithm for finding the minimum makespan for n aircraft and maximum position shift of k is* $O(n(2k+1)^{(2k+2)})$.

Proof. The nodes in each stage of the network are generated by all combinations of length $2k+1$, where each position in the sequence has at most $2k+1$ possible aircraft. The number of nodes in each stage is therefore $O((2k+1)^{(2k+1)})$; because there are $n$ stages, the total number of nodes in the network is $O(n(2k+1)^{(2k+1)})$. Each node can have at most $2k+1$ predecessors because the sequence of a node differs from the sequence of its predecessor only in the first and last position. so the number of arcs is $O(n(2k+1)^{(2k+2)})$. Pruning the network requires looking at each arc at most twice—once during the forward pass and once during the backward pass. The dynamic programming recursion examines each arc at most once, so total complexity is equal to the number of arcs in the network, which is $O(n(2k+1)^{(2k+2)})$. □

In the presence of precedence constraints, each node would require $O(k)$ work because we would need to check if precedence is violated between the final aircraft in the node and each of the other aircraft in the subsequence of the node. The complexity for preprocessing the entire network would thus be the number of nodes times $k$, which is the same as the running time of the pruning and dynamic programming recursion.

Although the complexity is exponential in $k$, it is of little consequence because $k$ is typically small (at most 3 in practice) (de Neufville and Odoni 2003). Therefore, the complexity of the algorithm is essentially linear in $n$.

**Figure 2.** Algorithm for computing the minimum makespan.

```
procedure FindMakespan:
  begin
    Set $T^*(\cdot)$ to $e(\cdot)$ for all nodes in stage 1, and
      $\infty$ for all nodes in stage $n$;
    for each $p = 2, \dots, n$ do
      for each node $j$ in stage $p$ do
        $T^*(j) = \max\{e(j), \min_{i\in P(j): T^*(i)\leqslant l(i)}(T^*(i)+\delta_i(j))\}$;
        pred$(j) = \arg\min_{i\in P(j): T^*(i)\leqslant l(i)}(T^*(i)+\delta_i(j))$;
end
```

## 5. Other Objective Functions

### 5.1. Minimizing the Maximum Delay Over All Aircraft

Let $D$ be an upper bound on the maximum delay over all aircraft (a trivial upper bound on the maximum delay is obtained by $n$ times the largest value of minimum required spacing, or the largest value over all aircraft of $l(\cdot)-e(\cdot)$). Alternatively, this upper bound can be computed from the optimal sequence of the minimum makespan solution (assuming a feasible solution exists). We will now assume that the earliest times and separations are all integers (or, equivalently, floating point numbers to some fixed precision), resulting in $D$ also being an integer. The solution that minimizes maximum delay can be calculated by iteratively performing a binary search on the interval $\{0, 1, \dots, D\}$, iteratively using the minimum makespan algorithm to check for feasibility.

The complexity of the algorithm is $O(\log D)$ multiplied by the complexity of the minimum makespan algorithm, which equals $O(n(2k+1)^{(2k+2)}\log D)$.

### 5.2. Minimizing the Sum of Delay Over All Aircraft

We now propose an algorithm that computes the minimum average delay solution within the CPS framework and can handle all previously mentioned constraints except the earliest/latest time-window constraints.

Working with the same CPS network as for the minimum makespan scenario, each node $i$ in stage $p$ is associated with a function $\theta_i(p) = t_1 + t_2 + \cdots + t_{p-1} + (n-p+1)t_p$, where $t_q$ is the arrival time of the $q$th aircraft in a sequence of aircraft along some path from the source to node $i$ (note that this is different from aircraft $q$, which is the $q$th aircraft in the FCFS sequence). For each node $j$ in stage $n$, we wish to minimize $\theta_j(n) = t_1 + t_2 + \cdots + t_{n-1} + t_n$.

Let $\theta^*(\cdot)$ be the value of $\theta(\cdot)$ in an optimal solution, and $t^*_q$ be the corresponding values of $t$. In order to compute the value of $\theta^*_j(n)$, we need to find the minimum value of $\theta_j(n)$ over all paths that lead to node $j$. In order to solve for the value of $\theta^*(\cdot)$ by dynamic programming, we first establish the following properties.

Proposition 2. *In an optimal solution, each aircraft is separated from its predecessor by exactly the minimum required separation between the two aircraft.*

The proof is by contradiction: if a feasible solution exists such that the proposition does not hold, it is possible to achieve a strictly lower value of total delay by moving all aircraft ahead until the proposition holds. Note that this property holds only if there are no time-window constraints.

Proposition 3. *All paths from the source to a given node in the CPS network contain the same set of aircraft.*

Proof. Given a node $i$ in stage $p$ of the network, let $V^-(i)$ be the set of all nodes (other than $i$) that belong to all paths from the source to node $i$, and let $V^+(i)$ be the set of all nodes (other than node $i$) that belong to all paths from node $i$ to the sink. Let $A^-(i)$ and $A^+(i)$ be the set of final aircraft of nodes in $V^-(i)$ and $V^+(i)$, respectively. Due to Lemma 1, $A^-(i)$ and $A^+(i)$ cannot have any aircraft in common (otherwise, there exists a source-sink path in which an aircraft repeats, which is not possible). Further, $A^-(i)$ has at least $(p-1)$ aircraft; otherwise, there exists a path from the source to $i$ in which an aircraft repeats. Similarly, $A^+(i)$ has at least $(n-p)$ elements. Because the sum of elements in $A^-(i)$ and $A^+(i)$ exactly equals $n-1$ (all aircraft except the final aircraft of node $i$), $A^-(i)$ and $A^+(i)$ have exactly $(p-1)$ and $(n-p)$ aircraft, respectively. Because every path from the source to node $i$ corresponds to a path of $(p-1)$ aircraft, all paths must contain the same set of aircraft, i.e., the elements of $A^-(i)$. □

Consider a path in the network ending in node $j$ belonging to stage $p$, in which node $i$ is the penultimate node. Then, for arc $(i, j)$,

$$\begin{aligned}\theta_j(p) &= t_1 + t_2 + \cdots + t_{p-1} + (n-p+1)t_p \\ &\geqslant t_1 + t_2 + \cdots + t_{p-1} + (n-p+1)(t_{p-1} + \delta_i(j)) \\ &= t_1 + t_2 + \cdots + t_{p-2} + (n-p+2)t_{p-1} \\ &\quad + (n-p+1)\delta_i(j) \\ &= \theta_i(p-1) + (n-p+1)\delta_i(j).\end{aligned}$$

Due to Proposition 3, all paths to node $j$ contain the same set of aircraft. Thus, for all nodes $i$ that precede $j$, the function $\theta_i(p-1) = t_1 + t_2 + \cdots + t_{p-2} + (n-p+1)t_{p-1}$ is calculated over the same set of aircraft. Further, the $(p-1)$th node in all paths ending in node $j$ is the same (by construction, it is the second-from-final aircraft of node $j$). Therefore, given a set of aircraft landing times, the associated functions $\theta_i(p)$ of all nodes $i$ that precede $j$ are equal in value; they differ only in the order of the terms $t_1, \ldots, t_{p-2}$.

The dynamic programming recursion can hence be written as follows.

$$\begin{aligned}\theta^*_j(s) &= \min_{t_1, \ldots, t_{p-1}} \theta_i(p-1) + (n-p+1)\delta_i(j) \\ &= \min_{i \in P(j)} \theta^*_i(p-1) + (n-p+1)\delta_i(j).\end{aligned}$$

Solving for $\theta^*(\cdot)$ for all nodes in stage $n$ is equivalent to solving the following shortest-path problem: assign each arc $(i, j)$ in the CPS network from stage $p-1$ to stage $p$ a "distance" of $(n-p+1)\delta_i(j)$, and solve for the shortest source-sink path in the network.

The proof of correctness follows from properties of shortest paths and is omitted here. The complexity of solving a shortest path on this network is linear in the number of arcs and is thus the same as that for minimizing makespan, which is $O(n(2k+1)^{(2k+2)})$.

## 6. Discrete-Time Models

Although the framework proposed in §3 can solve the versions of the runway-scheduling problem considered so far, it is not immediately applicable to complex objectives such as the minimization of the sum of arbitrary aircraft-dependent costs or to situations in which the separation requirements do not satisfy the triangle inequality (for example, for mixed arrival-departure operations on a runway). In this section, we present discrete-time models for solving these extensions. We assume that all input data (such as separations, time windows, etc.) are integer multiples of some time period. For instance, an interval of about five seconds is appropriate in the context of arrival scheduling given that the separations are of the order of a minute, and it is difficult to control aircraft positions and landing times to a very high degree of accuracy. We then create a copy of the CPS network for each time period of interest and solve the problems using dynamic programming on this discrete-time CPS network. Because the number of nodes and arcs in the network now depends on the number of time periods, the algorithms presented in this section have pseudopolynomial complexity (polynomial in the number of time periods).

### 6.1. Minimizing the Sum of Arbitrary Aircraft-Dependent Cost Functions

We now describe a pseudopolynomial algorithm for the CPS scheduling problem with arbitrary cost structures, assuming that time can be discretized to some appropriately small interval. For instance, an interval of about five seconds is appropriate in the context of arrival scheduling given that the separations are of the order of a minute and it is difficult to control aircraft positions and landing times to a very high degree of accuracy. It is assumed that all inputs to the problem (time windows, separation, etc.) are integer multiples of the interval. We consider scenarios in which each aircraft has its own delay cost function, and the objective is to determine the schedule that minimizes the sum of these costs.

Let $c(a, t_a)$ denote the cost of landing aircraft $a$ at time $t_a$. We wish to minimize $\sum_{a=1}^{n} c(a, t_a)$. Without loss of generality, we can assume that $c(\cdot)$ is positive (if not, we can add a constant to all values of $c(\cdot)$ to ensure that it is positive).

**6.1.1. Dynamic Programming Algorithm.** Given the CPS network as defined in earlier sections, let $\mathcal{T}(i)$ denote the set of feasible times that the final aircraft of node $i$ can land (generated from the time-window constraints). Define $J(i, t)$ to be the cost of a feasible schedule generated by a sequence of nodes starting at stage 1 and ending at node $i$, given that the final aircraft of node $i$ lands at time $t$. Let $J^*(i, t)$ represent the minimum value of $J(i, t)$ over all feasible schedules ending in node $i$ at time $t$. We then wish to minimize $J^*(i, t)$ over all nodes in the $n$th stage of the network and over all feasible time periods $t \in \mathcal{T}(i)$.

The optimal schedule is generated by the following dynamic programming recursion, where fin$(i)$ denotes the final aircraft of node $i$. If fin$(j)$ lands at time $t''$, then

$$J^*(j, t'') = \min_{\substack{i \in P(j),\\ t' \in \mathcal{T}(i):\,(t'' \geqslant t' + \delta_i(j))}} \left\{ J^*(i, t') + c(\mathrm{fin}(j), t'') \right\}. \tag{2}$$

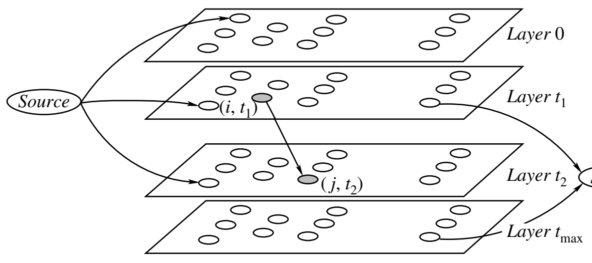
The following boundary condition applies to all nodes $i$ in stage 1.

$$J^*(i, t) = c(\mathrm{fin}(i), t) \quad \forall\, t \in \mathcal{T}(i). \tag{3}$$

The next section describes the network representation of the above recursion.

**6.1.2. Network Representation of the DP.** The solution to the recursion is equivalent to a shortest-path problem on a network that is constructed as follows. We begin by creating a copy of the original CPS network for each time interval, henceforth referred to as a *layer* of the network, as shown in Figure 3. Each node in this network is characterized by the pair $(i, t)$, representing node $i$ in the original CPS network belonging to layer $t$, i.e., the "state" corresponding to the final aircraft of node $i$ landing at time $t$. Note that node $(i, t)$ exists in the network only for $t \in \mathcal{T}(i)$. An arc exists from node $(i, t')$ to $(j, t'')$ only if arc $(i, j)$ exists in the original CPS network and $(t'' - t') \geqslant \delta_i(j)$, the minimum spacing between the final aircraft of nodes $i$ and $j$. Finally, the network contains arcs from the source to every node in stage 1, and an arc from every node in stage $n$ to the sink. Thus, a source-sink path, source $\to (i_1, t_1) \to \cdots \to (i_n, t_n) \to$ sink, represents a solution in which the final aircraft of node $i_1$ arrives at time $t_1$, that of node $i_2$ arrives at time $t_2$, and so on. We refer to this network as the discrete-time CPS network.

**Figure 3.** Schematic description of discrete-time network for minimizing the weighted sum of arrival times.



Lemma 4. *A schedule that is feasible* (*to CPS constraints, precedence requirements, time-window restrictions, and separation requirements that satisfy the triangle inequality*) *exists if and only if there exists a source-sink path in the discrete-time modified network.*

The proof is omitted here because the more general case of when the separations may violate the triangle inequality is proved in §6.2.

As a consequence of the above lemma, the problem can be solved as a shortest-path problem on the network with appropriately weighted arcs. Each arc entering node $(i, t)$ is weighted by the cost $c(\mathrm{fin}(i), t)$ of the final aircraft of node $i$ arriving at time $t$; all sink-adjacent arcs are assigned a cost of zero. The length of the shortest path from the source to node $(i, t)$ equals $J^*(i, t)$, and the length of the shortest path from the source to the sink, *assuming that such a path exists*, is the minimum cost solution. The absence of a source-sink path proves infeasibility of the problem.

**6.1.3. Bounding the Set of Feasible Arrival Times.** Because the computational efficiency of a shortest-path algorithm is determined by the number of nodes/arcs in the network, it is critical to be able to bound the total number of time intervals and the set of feasible times $\mathcal{T}(i)$ for each node. We now briefly describe an approach to obtaining good bounds in the absence of tight time-window constraints. From the makespan-minimization algorithm presented in §4, we can calculate for each node the earliest time that the final aircraft of that node can land, which gives a bound on the set of feasible landing times that is necessarily tighter than $e(\cdot)$ determined by the time-window constraint. However, obtaining a tighter bound than $l(\cdot)$ on the latest arrival time is slightly harder.

We first observe that in any optimal solution to the problem, an aircraft must either land at its earliest time $e(\cdot)$, or the separation between an aircraft and its predecessor is upper bounded by the largest required separation between all pairs of aircraft (denoted by $\delta_{\max}$). (Otherwise, because $c(\cdot)$ is positive, a better solution may be obtained by slightly advancing the aircraft, which contradicts optimality.) Let $\lambda(i)$ be the latest time that $\text{fin}(i)$ lands, given that it must land either at $e(\cdot)$ or is separated by $\delta_{\max}$ from its predecessor. Then, $\lambda(\cdot)$ is calculated using the following recursion.

$$\lambda(j) = \max\left\{e(j), \max_{i \in P(j)}(\lambda(i) + \delta_{\max})\right\}. \tag{4}$$

The proof of correctness of this recursion follows from the fact that $\lambda(j) \geqslant e(j)$ and $\lambda(j) \geqslant \lambda(i) + \delta_{\max}$ for all $i \in P(j)$, and one of these inequalities must hold as an equality in an optimal solution.

The lower and upper bounds obtained above (equivalent to a shortest- and longest-path calculation on the original CPS network) are then intersected with the time-window constraints for each aircraft to obtain $\mathcal{T}(\cdot)$.

**6.1.4. Complexity.** Let $L$ denote an upper bound on the number of layers (time intervals) during which an aircraft may land, which is the maximum cardinality of $\mathcal{T}(i)$ over all nodes $i$ (and is bounded above by $l(i) - e(i)$). The number of nodes in the discrete-time network is $O(nL(2k+1)^{(2k+1)})$. Each node in this network could have arcs leading into $O(L(2k+1))$ other nodes, because a node in the original CPS network can lead into up to $2k+1$ nodes, each of which now has up to $L$ copies. Thus, the number of arcs in the discrete-time network is $O(nL^2(2k+1)^{(2k+2)})$, which is the complexity of solving the shortest-path problem on the network. This algorithm has been shown to be amenable to real-time implementation in computational experiments conducted by Lee (2008).

## 6.2. Scenarios in Which the Triangle Inequality Is Violated

We now describe modifications to the algorithm in order to handle cases where the triangle inequality is violated (for instance, this situation occurs when simultaneously scheduling both arrivals and departures on a single runway). The key idea is to expand the state space to keep track of spacings that may violate triangle inequality.

In the modified network, each node represents a 3-tuple $(i, t, d)$ corresponding to the final aircraft of node $i$ of the original CPS network arriving/departing at time $t$ such that the separation between $\text{fin}(i)$ and its predecessor is greater than or equal to $d$ intervals (this will be made more precise shortly). Note that the sequence of aircraft in each node contains at least three aircraft $(2k+1)$; therefore, the aircraft immediately preceding $\text{fin}(i)$ is not ambiguous (it is the penultimate aircraft of node $i$). Nodes in stage 1 represent the 3-tuple $(i, t, 0)$ because they do not have a predecessor aircraft. A schematic description of the modified network is shown in Figure 4.

A node $(i, t, d)$ exists if:
1. Node $i$ exists in the original CPS network,
2. $t \in \mathcal{T}(i)$, and
3. $d_i^{\min} \leqslant d \leqslant d_i^{\max}$, as computed below.

The parameter $d_i^{\min}$ equals the minimum separation required between the penultimate and final aircraft of node $i$ (say, aircraft $a$ and $b$, respectively). The parameter $d_i^{\max}$ is defined to be the minimum separation that is required between $a$ and $b$ in order to ensure that any aircraft that trails $b$ is sufficiently separated from $a$ as long as the minimum separation between $b$ and that aircraft is maintained. More formally,

$$d_i^{\min} = \delta_{ab}$$
$$d_i^{\max} = \max\left\{\max_{j:\, i \in P(j)}\left\{\delta_{a,\,\text{fin}(j)} - \delta_{b,\,\text{fin}(j)}\right\}, d_i^{\min}\right\}.$$

Example 1. Given node $i$ in the CPS network with penultimate and final aircraft $a$ and $b$, respectively, let the minimum required separation between $a$ and $b$ be 60 seconds; then, $d_i^{\min} = 60$ seconds. Suppose there exists a node $j$ with final aircraft $c$ such that arc $(i, j)$ exists in the CPS network, and $\delta_{ac} = 135$ seconds and $\delta_{bc} = 70$ seconds. Then, the triangle inequality for the triplet of aircraft $(a, b, c)$ is violated. However, if aircraft $a$ and $b$ were separated by any interval greater than or equal to 65 seconds, then the minimum spacing between $a$ and $c$ is satisfied as long as we ensure the minimum spacing between $b$ and $c$. Therefore, in this case, $d_i^{\max} = 65$ seconds.

An arc exists between $(i, t', d')$ and $(j, t'', d'')$ if:
1. Arc $(i, j)$ exists in the original CPS network,
2. $(t'' - t') \geqslant d_j^{\min}$,
3. $d'' = \min\{(t'' - t'), d_j^{\max}\}$, and
4. $d' + t'' - t'$ is greater than or equal to the separation between the penultimate aircraft of $i$ and the final aircraft of $j$. (Note that all predecessors of node $j$ have the same penultimate aircraft because the aircraft subsequence of node $j$ has at least three aircraft.)

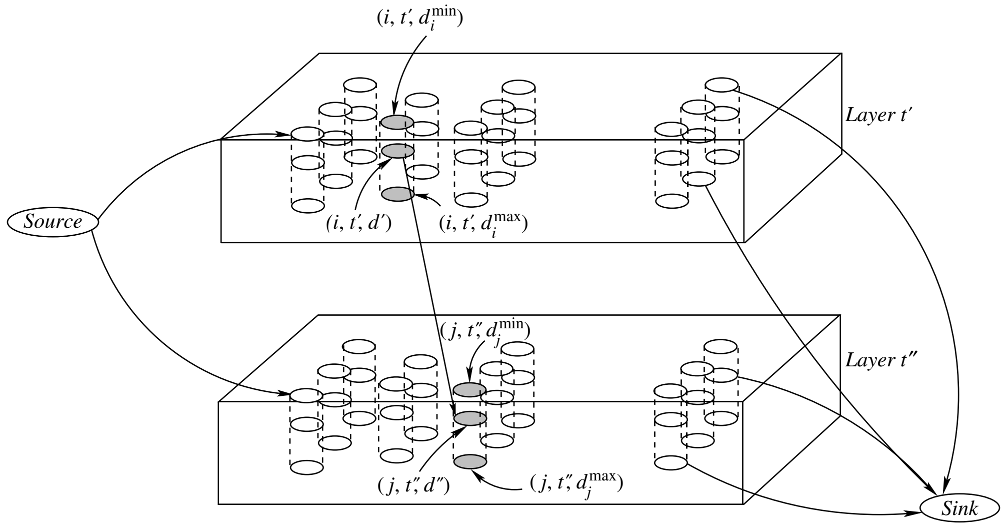
Finally, adding arcs from the source to each node in stage 1 and from each node in stage $n$ to the sink completes the network.

Lemma 5. *Every source-sink path in the discrete-time triangle-inequality modified network represents a feasible schedule* (*one that satisfies CPS constraints, precedence requirements, time-window restrictions, and separation requirements*).

Proof. Consider a path defined by a sequence of nodes $(v_1, t_1, d_1), (v_2, t_2, d_2), \ldots, (v_n, t_n, d_n)$ and arcs between successive nodes. By the properties of the CPS network, the sequence of nodes $v_1, v_2, \ldots, v_n$ satisfies CPS and time-window constraints. Also, by construction of $\mathcal{T}(\cdot)$, every $t_i$ explicitly obeys time-window restrictions. Therefore, it only remains to be shown that minimum separations are satisfied in any source-sink path.

The separation between $\text{fin}(v_i)$ and $\text{fin}(v_{i+1})$ is satisfied because an arc from node $(v_i, t_i, d_i)$ to $(v_{i+1}, t_{i+1}, d_{i+1})$ exists only if $(t_{i+1} - t_i) \geqslant d_{v_{i+1}}^{\min} = \delta_{v_i}(v_{i+1})$. We now show

**Figure 4.** Discrete-time CPS network when the triangle inequality is violated.



that the separation between $\mathrm{fin}(v_{i-1})$ and $\mathrm{fin}(v_{i+1})$ is satisfied. By construction,

$$d_i + t_{i+1} - t_i \geqslant \delta_{\mathrm{fin}(v_{i-1}),\,\mathrm{fin}(v_{i+1})}. \tag{5}$$

Because $d_i = \min\{t_i - t_{i-1}, d_{v_i}^{\max}\}$ (also by construction), we get $d_i \leqslant t_i - t_{i-1}$. Substituting this back into Equation (5) gives $t_{i+1} - t_{i-1} \geqslant \delta_{\mathrm{fin}(v_{i-1}),\,\mathrm{fin}(v_{i+1})}$, which is feasible. □

Lemma 6. *An optimal schedule* (*one that minimizes the sum of positive separable costs for each aircraft subject to CPS constraints*, *precedence requirements*, *time-window restrictions*, *and separation requirements that may violate the triangle inequality*) *is represented by a source-sink path in the discrete-time triangle-inequality modified network.*

Proof. Let an optimal solution to the problem be represented by the set of pairs $\{(i_1, t_1), (i_2, t_2), \ldots, (i_{n-1}, t_{n-1}), (i_n, t_n)\}$, where $(i_q, t_q)$ represents aircraft $i_q$ arriving/departing at time $t_q$. Due to earlier proved properties of the basic CPS network (in §3), there exists a *unique* path in the original CPS network corresponding to this sequence. Let the sequence of nodes be denoted by $v_1, v_2, \ldots, v_n$ belonging to stage 1 through $n$, respectively.

Because the given solution is optimal, each aircraft lands either at the earliest time, or its separation from the previous aircraft is lower bounded by the minimum separation between the aircraft pair and upper bounded by the maximum separation between all aircraft pairs (otherwise, it is possible to reduce the solution cost by moving the aircraft ahead by one time interval). Thus, the given arrival time $t_q$ must belong to the interval $\mathcal{T}(v_q)$ as computed in §6.1.3, and there exists at least one copy of node $v_q$ in layer $t_q$. Because $(t_q - t_{q-1}) \geqslant d_{v_q}^{\min}$, each aircraft can be assigned to a node $(v_q, t_q, d_q)$, where $d_q \geqslant d_{v_q}^{\min}$ and $d_q = \min\{(t_q - t_{q-1}), d_{v_q}^{\max}\}$.

To complete the proof, we use an inductive argument to show that there exist arcs between these nodes, forming a source-sink path in the discrete-time triangle-inequality modified network. Consider three nodes $(v_{q-1}, t_{q-1}, d_{q-1})$, $(v_q, t_q, d_q)$, and $(v_{q+1}, t_{q+1}, d_{q+1})$ corresponding to arrival events $(i_{q-1}, t_{q-1})$, $(i_q, t_q)$, and $(i_{q+1}, t_{q+1})$, respectively. We show that if an arc exists between $(v_{q-1}, t_{q-1}, d_{q-1})$ and $(v_q, t_q, d_q)$, then the nodes $(v_q, t_q, d_q)$ and $(v_{q+1}, t_{q+1}, d_{q+1})$ satisfy the four requirements for an arc to exist between them.

1. The arc $(v_q, v_{q+1})$ exists in the original CPS network because the nodes were chosen such that they form a source-sink path.
2. $(t_{q+1} - t_q) \geqslant d_{v_{q+1}}^{\min}$ holds true because $(t_{q+1} - t_q) \geqslant \delta_{v_q}(v_{q+1}) = d_{v_{q+1}}^{\min}$.
3. $d_{q+1} = \min\{(t_{q+1} - t_q), d_{v_{q+1}}^{\max}\}$ by construction.
4. We need to prove that $d_q + t_{q+1} - t_q \geqslant \delta_{i_{q-1}, i_{q+1}}$. Because there exists an arc between $(v_{q-1}, t_{q-1}, d_{q-1})$ and $(v_q, t_q, d_q)$, $d_q = \min\{(t_q - t_{q-1}), d_{v_q}^{\max}\}$. There are two possible cases:

*Case* 1. $d_q = (t_q - t_{q-1})$. In this case,

$$\begin{aligned} d_q + t_{q+1} - t_q &= t_q - t_{q-1} + t_{q+1} - t_q \\ &= t_{q+1} - t_{q-1} \qquad \text{(due to solution feasibility)} \\ &\geqslant \delta_{i_{q-1}, i_{q+1}}. \end{aligned}$$

*Case* 2. $d_q = d_{v_q}^{\max}$. In this case,

$$\begin{aligned} &d_q + t_{q+1} - t_q \\ &= d_{v_q}^{\max} + t_{q+1} - t_q \\ &\geqslant \delta_{i_{q-1}, i_{q+1}} - \delta_{i_q, i_{q+1}} + t_{q+1} - t_q \quad \text{(by definition of } d^{\max}\text{)} \\ &\geqslant \delta_{i_{q-1}, i_{q+1}} - \delta_{i_q, i_{q+1}} + \delta_{i_q, i_{q+1}} \qquad \text{(due to solution feasibility)} \\ &= \delta_{i_{q-1}, i_{q+1}}. \end{aligned}$$

The arc from $(v_1, t_1, d_1)$ to $(v_2, t_2, d_2)$, i.e., an arc from stages 1 to 2, satisfies the first three requirements by construction and therefore exists in the network. By induction, this implies that all arcs from $(v_q, t_q, d_q)$ and $(v_{q+1}, t_{q+1}, d_{q+1})$ for $q \in \{2, \ldots, (n-1)\}$ exist. Because all source-adjacent and sink-adjacent arcs exist by construction, we have identified a sequence of nodes and arcs forming a source-sink path in the network that represents the given optimal solution. □

Lemmas 5 and 6 together imply that minimizing the schedule cost over all paths in the constructed network will yield an optimal schedule. The problem can be solved as a shortest-path computation by weighting the arcs of the network as follows: each arc entering node $(i, t, d)$ is assigned a cost $c(i, t)$, and arcs to the sink are assigned zero cost.

### 6.3. Complexity

The discrete-time triangle-inequality modified network has $O(n(2k+1)^{(2k+1)}L\delta_{\max})$ nodes, where $L$ is the maximum cardinality of $\mathcal{T}(i)$ over all nodes $i$ (which is bounded by $(l(i)-e(i))$, and $\delta_{\max}$ is the maximum violation of the triangle inequality, which is bounded by the largest minimum separation among all pairs of aircraft. Each node $(v, t, d)$ can lead into up to $(2k+1)L$ other nodes $(v', t', d')$ because $v$ has at most $2k+1$ successors, and there are up to $L$ values of $t'$. (Note that the number of values of $d'$ does not enter the complexity because $d'$ is defined by the values of $t$ and $t'$.) Thus, the complexity of the algorithm is $O(nL^2\delta_{\max}(2k+1)^{(2k+2)})$. Thus, accommodating the triangle inequality only adds a complexity of $\delta_{\max}$ to an equivalent problem in which the separation requirements obey the triangle inequality.

Remark 1. If the triangle inequality is obeyed, then $d_i^{\min} = d_i^{\max} = \delta_{ab}$. In this case, the modified network (Figure 4) is equivalent to the discrete-time CPS network when the triangle inequality is satisfied (Figure 3). The proof of Lemma 5 therefore also proves Lemma 4.

## 7. Simultaneous Scheduling of Departures and Arrivals

We now consider the scheduling problem when arrivals and departures share a common runway. The separation requirements between arrival and departure aircraft are shown in Table 1. The triangle inequality is violated only in the following cases: (i) heavy arrival, followed by any departure, followed by large arrival (violated by 22 seconds) and (ii) heavy arrival, followed by any departure, followed by small arrival (violated by 61 seconds).

The quadrilateral and all higher polygon inequalities are valid. One important observation here is that the triangle inequality is violated only by two arrivals separated by a departure. The implication is that it is sufficient to ensure that the triangle inequality is satisfied between pairs of arrival aircraft in order to ensure that it is satisfied across the entire sequence, a property that will be exploited in one of our algorithms.

### 7.1. Coupled Arrivals and Departure Position Constraints

The first possible scenario for scheduling mixed operations on a single runway is that the FCFS sequence consists of both arrival and departure aircraft, and every aircraft has a CPS constraint with respect to its FCFS sequence position. The challenge here is that the triangle inequality is violated when departures and arrivals interact.

Following the algorithm described in §6.2, the problem can be solved with a complexity of $O(nL^2\delta_{\max}(2k+1)^{(2k+2)})$. In this case, $\delta_{\max}$ is 61 seconds (divided by an appropriate time discretization, if applicable). Further, because only six triplets of aircraft out of a possible 216 can lead to violations of the triangle-inequality, we expect that only a small fraction of nodes will require the triangle-inequality modification, leading to significantly reduced complexity in practice (the exact reduction in complexity depends on the specific instance being solved).

### 7.2. Optimal Merging of Arrivals and Departures on a Single Runway with Independent Position Shift Constraints

In this section, we study a more realistic scenario of arrival-departure trade-offs on a single runway that is fed by several departure queues (each following FCFS) and one arrival stream (with CPS constraints). This is a good representation of current airport operations, in which aircraft queue up for takeoff in staging areas at the departure runway threshold. Due to the narrow layout of the taxiways, aircraft within the same departure queue cannot overtake each other, but position swaps can occur between aircraft in different departure queues.

Arrivals are constrained as before by CPS, precedence, and time-window constraints. Because departures are already in their respective queues, it is not necessary to consider earliest time restrictions for the departures. In addition, we do not consider latest time restrictions on departures either, thereby giving arrivals greater priority than departures (which can be delayed indefinitely in order to accommodate arrivals with latest time constraints). The departure queues are coupled via position constraints (for example, aircraft $a$ must be between the 10th and 15th aircraft to depart) as well as precedence constraints (for example, aircraft $a$ in the first queue must depart before aircraft $b$ in the second queue).

The problem can be stated as follows: Given an arrival sequence of $n^A$ aircaft with an associated FCFS sequence, $n^D$ departure aircraft over $q$ queues, each having $n_1^D, n_2^D, \ldots, n_q^D$ aircraft, position constraints for each departure aircraft, position shift constraints for the arrivals (with maximum position shift $k$), precedence constraints between pairs of arriving aircraft or pairs of departing aircraft, time-window restrictions on arriving aircraft, and separation requirements listed in Table 1, find a schedule (sequence of aircraft and associated runway operation

**Table 3.** Notation used in the algorithm for merging departure queues with an arrival stream.

| | |
|---|---|
| $\mathbf{x}$ | The vector $(x_1, x_2, \ldots, x_q)$ representing the number of aircraft that have departed from each departure queue, i.e., $x_1$ from the first queue, $x_2$ from the second, etc. |
| $\mathbf{x} \mid r$ | State of the departure queues given that the last departure was from the $r$th queue. |
| $\delta_i(j)$ | Minimum separation between the final aircraft of nodes $i$ (leading) and $j$ (trailing). |
| $\mu_{rs}(\mathbf{x}, \mathbf{y})$ | The minimum possible makespan of a sequence of departures starting from departure state $\mathbf{x} \mid r$ and ending in departure state $\mathbf{y} \mid s$. |
| $\delta_i(\mathbf{x} \mid r)$ | The minimum required separation between the final aircraft of node $i$ and the $x_r$th aircraft of the $r$th queue (leading and trailing, respectively). |
| $\delta_{\mathbf{x} \mid r}(i)$ | The minimum required separation between the $x_r$th aircraft of the $r$th queue and the final aircraft of node $i$ (leading and trailing, respectively). |

times) that minimizes the makespan while satisfying all the listed constraints.

The following section describes a strongly polynomial algorithm to solve this problem.

**7.2.1. Algorithm Description.** For simplicity of description, we introduce a dummy arrival aircraft that is preceded by all aircraft in the system (we will describe later how this is enforced). The separation between the dummy aircraft and any other aircraft is then set to zero, forcing the makespan of the solution with the dummy aircraft to equal that of the original problem.

The notation we use to describe our algorithm is listed in Table 3.

*Calculating the makespan of departure subsequences.* We pose the problem of finding the values of $\mu_{rs}(\mathbf{x}, \mathbf{y})$ as a shortest-path problem on a network shown in Figure 5. The network has a "state" node for every possible value of $\mathbf{x} \mid r$; an arc exists from state node $\mathbf{x} \mid r$ to $\mathbf{x}' \mid s$ if $x_i = x'_i$ for all $i \neq s$ and $x'_s = x_s + 1$, i.e., if state $\mathbf{x}'$ is reached from state $\mathbf{x}$ by one departure from queue $s$. The arc from $\mathbf{x} \mid r$ to $\mathbf{x}' \mid s$ is assigned a "distance" equal to the minimum separation required when the $x_r$th aircraft of queue $r$ is followed by the $x_s$'th aircraft of queue $s$.

Nodes that violate the position constraints or the precedence constraints are then removed from the network.

Example 2. If the fifth aircraft in queue $r$ must be within the first 10 departures, the constraint is imposed by removing all nodes $\mathbf{x}$ in which $x_1 + x_2 + \cdots + x_q > 10$ and $x_r < 5$.

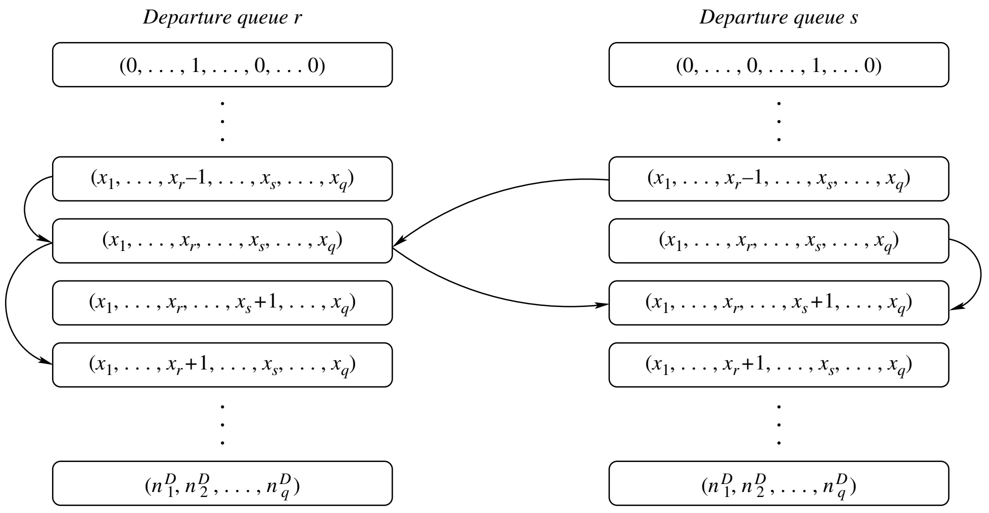
Example 3. If the fourth aircraft in the $r$th queue must depart before the sixth aircraft in the $s$th queue, the constraint is imposed by removing all nodes $\mathbf{x}$ in which $x_r < 4$ and $x_s \geqslant 6$. This network is referred to as the departure position-constrained network.

Proposition 4. *The makespan $\mu_{rs}(\mathbf{x}, \mathbf{y})$ is given by the shortest-path distance from node $\mathbf{x} \mid r$ and ending in state $\mathbf{y} \mid s$ in the departure-position constrained network.*

The proof follows from the fact that all feasible states and transitions between states of the departure queue are explicitly enumerated; therefore, every path represents a feasible sequence of departures. Because the triangle inequality is not violated and departures have no earliest/latest time constraints, the makespan is the sum of minimum separations between successive departures.

*Calculating the makespan of combined arrivals and departures.* We construct a CPS network for the arrival sequence that satisfies CPS and precedence constraints for the arrival stream as described in §3. We then associate each node $i$ in the network with the function $J_i(\mathbf{x})$, denoting the arrival time of the final aircraft of node $i$ given that it is immediately preceded by a departure operation and the departure queue is in state $\mathbf{x}$ at the time of arrival. Finally, a dummy node is created (with the dummy aircraft as its final aircraft) that is preceded by all nodes in stage $n^A$ of the arrival CPS network. The dummy node is associated with

**Figure 5.** Shortest-path network for computing the makespan of departure subsequences.


Departure queue r
Departure queue s
(0, . . . , 1, . . . , 0, . . . 0)
(0, . . . , 0, . . . , 1, . . . 0)
(x1, . . . , xr–1, . . . , xs, . . . , xq)
(x1, . . . , xr, . . . , xs, . . . , xq)
(x1, . . . , xr, . . . , xs+1, . . . , xq)
(x1, . . . , xr+1, . . . , xs, . . . , xq)
(nD1, nD2, . . . , nDq)

the state $(n_1^D, n_2^D, \ldots, n_q^D)$, corresponding to all departures having occurred before the dummy aircraft's arrival.

Let $J_i^*(\mathbf{x})$ denote the minimum value of $J_i(\mathbf{x})$ over all feasible paths ending in node $i$ given that the departure queues are in state $\mathbf{x}$. We wish to minimize $J_i^*(n_1^D, n_2^D, \ldots, n_q^D)$, where $i$ is the last (dummy) node in the CPS network.

Given two nodes $i$ and $j$ such that the final aircraft of node $i$ lands when the departure queue is in state $\mathbf{x}$ and that of node $j$ lands when the departure queue state is $\mathbf{y}$, and the final aircraft of nodes $i$ and $j$ are separated in the runway sequence only by departing aircraft, we denote the minimum separation required between the final aircraft of nodes $i$ and $j$ by $\sigma_{ij}(\mathbf{x}, \mathbf{y})$. There are three components to this separation:

1. the separation between the final aircraft of node $i$ and the first departure after state $\mathbf{x}$, or $\delta_i(\mathbf{x}' \mid r)$, where $x' \mid r$ is obtained from state $\mathbf{x}$ by one additional departure from queue $r$;
2. the separation between the last aircraft to depart (say, from queue $s$) given state $\mathbf{y}$ and the final aircraft of node $j$, or $\delta_{\mathbf{y}\mid s}(j)$;
3. the separations between departures starting at state $\mathbf{x}' \mid \mathbf{r}$ and ending at state $y \mid s$, or $\mu_{rs}(\mathbf{x}', \mathbf{y})$.

The value of $\sigma_{ij}(\mathbf{x}, \mathbf{y})$ is calculated by minimizing over all values of $r, s$. As noted earlier, the separations in Table 1 are such that the triangle inequality is violated only by two arrival aircraft separated by a departure aircraft. Therefore, given $\delta_i(j)$ (the separation between the final aircraft of nodes $i$ and $j$), $\sigma_{ij}(\mathbf{x}, \mathbf{y})$ can be corrected for triangle inequality violations and is calculated as follows.

$$\sigma_{ij}(\mathbf{x},\mathbf{y}) = \max\left\{\delta_i(j),\ \min_{\substack{r,s\in\{1,\ldots,q\}\\ x'_t=x_t\ \forall t\neq r\\ x'_r=x_r+1}} \delta_i(\mathbf{x}'\mid r)+\mu_{rs}(\mathbf{x}',\mathbf{y})+\delta_{\mathbf{y}\mid s}(j)\right\}. \quad (6)$$

Lemma 7. *The values of $J^*(\cdot)$ are calculated by the following recursion*

$$J_j^*(\mathbf{y}) = \max\left\{e(j),\ \min_{\substack{i\in P(j)\\ \mathbf{0}\leqslant x<y:\, J_i^*(x)\leqslant l(i)}} (J_i^*(\mathbf{x})+\sigma_{ij}(\mathbf{x},\mathbf{y}))\right\}.$$

The proof of this recursion is omitted here because it is very similar to the proof of Lemma 3: $J_j^*(\mathbf{y})$ must exceed $e(j)$ (due to time-window constraints) as well as the minimum value of $J_i^*(\mathbf{x})+\sigma_{ij}(\mathbf{x},\mathbf{y})$ (due to minimum separation requirements), and one of these inequalities must hold as an equality due to optimality.

Applying the boundary condition to all nodes in stage 1,

$$J_i^*(\mathbf{x}) = \min_{r,s\in\{1,\ldots,q\}} \mu_{rs}(\mathbf{0},\mathbf{x}), \quad (7)$$

and unrolling the recursion to compute $J^*(\cdot)$ for the last dummy aircraft with the departure queue state $\mathbf{x} = (n_1^D, n_2^D, \ldots, n_q^D)$ yields the minimum makespan solution.

#### 7.2.2. Complexity.

Lemma 8. *The complexity of the proposed algorithm for optimally merging arrivals and departures on a single runway with independent position shift constraints to minimize the makespan is $O((\gamma^2q^5+n^A(2k+1)^{(2k+2)})(1+n^D/q)^{2q})$.*

*Calculating the values of $\mu$ and $\sigma$*: The number of departure queue states is $(1+n_1^D)\times(1+n_2^D)\times\cdots\times(1+n_q^D)$, which is $O((1+n^D/q)^q)$. The number of departure queue nodes is therefore $O(q(1+n^D/q)^q)$. Because each node can lead to up to $q$ nodes, corresponding to one additional departure from each queue, the number of arcs is $O(q^2(1+n^D/q)^q)$. The complexity of computing the shortest path from a given node to all other nodes in a directed acyclic graph equals the number of arcs in the network (Cormen et al. 1990, p. 536). Therefore, the complexity of computing all values of $\mu_{ab}(\mathbf{x},\mathbf{y})$ is the number of arcs multiplied by the number of nodes, which is $O(q^3(1+n^D/q)^{2q})$.

In order to determine the values of $\sigma_{ij}(\mathbf{x},\mathbf{y})$ for all pairs of nodes $i$ and $j$, it is sufficient to compute it for every pair of arriving aircraft *types* corresponding to nodes $i$ and $j$ and is not necessary for all pairs of nodes. Given the number of arrival aircraft types $\gamma$, the work done to compute $\sigma$ is $O(\gamma^2q^2)$ times the work to calculate $\mu$. Thus, the complexity of calculating all values of $\sigma$ is $O(\gamma^2q^5(1+n^D/q)^{2q})$.

*Calculating the values of $J^*(\cdot)$*: The number of nodes in the CPS network is $O(n^A(2k+1)^{(2k+1)})$. Because there are $O((1+n^D/q)^q)$ possible states of the departure queue, the number of values of $J^*$ that need to be computed is $O((2k+1)^{(2k+1)}(1+n^D/q)^q)$. Each computation of $J^*$ is performed over $O(2k+1)$ predecessor nodes and over $O((1+n^D/q)^q)$ preceding departure queue states, which is $O((2k+1)(1+n^D/q)^q)$ times. Thus, the complexity of the recursion is $O(n^A(2k+1)^{(2k+2)}(1+n^D/q)^{2q})$.

Thus, the complexity of the entire algorithm (required to calculate both $J^*$ and $\mu$) is $O((\gamma^2q^5+n^A(2k+1)^{(2k+2)})\cdot(1+n^D/q)^{2q})$. □

Although it is possible to extend our algorithm to other objectives (such as minimizing average delay) using time discretization, the resulting algorithms are unlikely to be computationally tractable due to the increased complexity.

## 7.3. Scheduling Operations on Multiple Runways

The problem of scheduling operations on multiple runways is more complex, because it depends greatly on the layout of the airport and the relative orientation and geometry of the runways. For example, depending on the distance between the centerlines, operations on parallel runways may or may not need to be coordinated. In the case of runways with sufficient separation, runway assignments are made based on the location of the gates, departure fixes, taxi routes, or the length of runway required by an aircraft, etc. Any realistic runway assignment algorithm must take into account factors like airline gate use agreements, departure fix assignments, missed approach paths, and taxi routes, and is beyond the scope of this paper.

Many space-constrained airports operate in practice as single-runway airports (depending on maintenance schedules or wind direction). In addition, when the runway centerlines are closely spaced, interarrival separations between the two runways are effectively equal to the single-runway separation requirements, reducing the problem of scheduling operations (with runway assignments) to that of single-runway scheduling (de Neufville and Odoni 2003, Lee 2008). For the intermediate separations between parallel runways, and for intersecting runways, separation requirements vary depending on the airport and the country. It is therefore difficult to draw any general conclusions; however, if the separation requirements satisfy the quadrilateral inequality, the approaches proposed in this paper can be used for scheduling operations on multiple runways.

## 8. A Prototype Implementation

This section presents a proof-of-concept implementation, describes its performance on various realistic scenarios, and provides some insight into the nature of the minimum makespan solution.

### 8.1. Experimental Design

The Denver ARTCC airspace is shown in Figure 6 (left), with the jet routes that carry arrival traffic. Arrivals at DEN are routed through one of eight arrival gates into the Denver TRACON airspace (a 42 nmi radius around the airport). We consider a scenario in which all traffic from the NE and NW gates land on a single runway, and we wish to minimize the makespan of the arrival sequence. The number of aircraft arriving through each of the gates was based on the historical fraction of traffic through the different gates (Bureau of Transportation Statistics 2007). We used a Poisson arrival process to generate the sequence of times at which aircraft enter the Denver Air Route Traffic Control Center (ZDV); this assumption has been validated by Willemain et al. (2004), who showed that the interarrival times at airports before the final control actions are executed are nearly Poisson. Given the times that aircraft enter ZDV, it is possible to compute the FCFS order of arrivals at the runway and the estimated times of arrival (ETAs) at the airport by using the average time spent by aircraft on the different jet routes. This data is shown for the northern arrivals in Figure 6 (right).

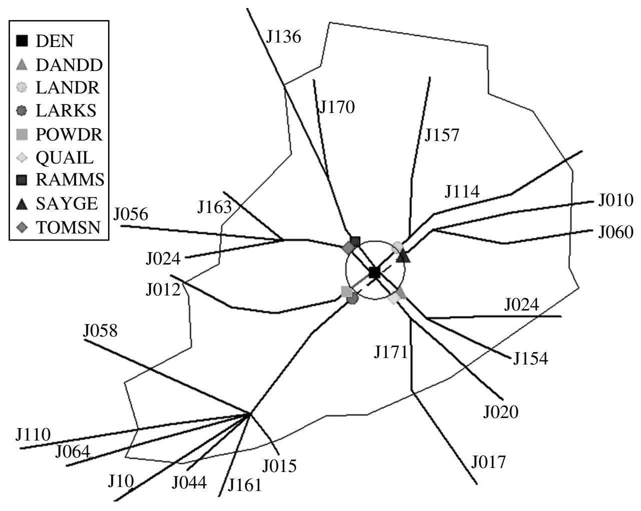
For each of the directions of arrival, we divided the traffic equally among all the corresponding jet routes. As described in Neuman and Erzberger (1991), it is necessary to maintain FCFS order among aircraft in the same jet route. Therefore, we used the jet routes to determine the landing precedence relationships. We set the earliest arrival time at one minute less than the ETA, because it is often not economically worthwhile to move the landing time forward by more than a minute (Neuman and Erzberger 1991). We set the latest possible arrival time at 60 minutes after the ETA, implying that we would not put an aircraft on hold for more than an hour.

Because the extent of the benefit of resequencing aircraft would depend on the relative fractions of different sizes of aircraft, we considered two mixes of aircraft types: one a 40% heavy, 40% large, and 20% small mix, and the other a 45% heavy, 45% large, and 10% small mix of aircraft, which are practical because most major airports are likely to have more heavy and large aircraft operations than small ones.

The data for a single instance of our experiment was thus constructed as described in Table 4.

Given an FCFS sequence with precedence and minimum spacing requirements, we applied our algorithm to minimize the makespan of the sequence and measured

**Figure 6.** [Left] Denver airspace, showing jet routes and arrival gates. [Right] Jet route traversal times and percentage of traffic for northern arrivals (Neuman and Erzberger 1991).



| Jet route no. | Time within ZDV | Gates | |
|---|---|---|---|
| J136 | 42.30 | TOMSN | NW Gates (45% of traffic) |
| J056 | 45.45 | TOMSN | |
| J170 | 45.00 | RAMMS | |
| J024 | 47.78 | TOMSN | |
| J136 | 45.00 | RAMMS | |
| J114 | 41.43 | LANDR | NE Gates (55% of traffic) |
| J010 | 45.00 | SAYGE | |
| J157 | 45.00 | LANDR | |
| J060 | 45.00 | SATGE | |

**Table 4.** Method for construction of a problem instance in our experiments.

| | |
|---|---|
| *Step* 1. | Choose values for $n \in \{10, 20, 30, 40, \text{ or } 50\}$, $k \in \{1, 2, 3\}$, and $r \in \{24, 40, 60\}$. |
| *Step* 2. | Construct a sequence of $n$ arrival times at the boundary of ZDV with exponentially distributed interarrival times with mean $1/r$. |
| *Step* 3. | Assign a jet route to each aircraft based on the fraction of traffic on each jet route. |
| *Step* 4. | Construct the precedence relationships among aircraft within the same jet route. |
| *Step* 5. | Compute the ETA for each aircraft using transit time within ZDV, and construct the FCFS sequence. |
| *Step* 6. | Assign an earliest arrival time of ETA minus 1 minute, and a latest arrival time of ETA plus 60 minutes to each aircraft. |
| *Step* 7. | Assign an aircraft type to each aircraft based on the fleet mix (using either 40% Heavy, 40% Large, and 20% Small, or 45% Heavy, 45% Large, and 10% Small), and use this to determine the required spacing between the aircraft. |

the difference between the FCFS makespan and the CPS makespan. We performed Monte Carlo simulations for various values of $n$, $k$, and $r$, the results of which are presented in the next section.

### 8.2. Results

The improvement in makespan of the optimal CPS sequence over that of the FCFS sequence is shown in Figure 7. Each data point in the figure is an average of 100 randomly generated instances.

The experiments demonstrate that there is greater benefit to rescheduling when the fleet mix is more homogeneous (40% heavy, 40% large, and 20% small). Also, the benefit to using CPS is greater when the arrival rate is greater (under lower arrival rates, the gaps in the FCFS sequence are difficult to fill up because we allow an aircraft to advance up to a minute ahead of the ETA). Finally, the experiments show that at rates of 40 aircraft per hour, which are achieved at most major airports during peak demand, the benefit to rescheduling can be around 5% of the FCFS makespan (or three minutes saved over a one-hour period, equivalent to a throughput increase of two to three aircraft per hour), which is significant in today's operating environment.

We next study the effect of minimizing makespan on the average delay, because it is conceivable that makespan is minimized at the cost of total system delay, which would not be a desirable outcome. Figure 8 shows the dependence of gain in average delay over the gain in makespan. Each data point is one instance of a scenario (10, 20, 30, 40, or 50 aircraft; an arrival rate of 20, 40, or 60 aircraft per hour; and a fleet mix of 40% heavy, 40% large, and 20% small, or 45% heavy, 45% large, and 10% small). The $x$-axis (makespan gain) is the difference between the makespan of the FCFS sequence and the makespan of the CPS sequence; the $y$-axis is the difference in average delay per aircraft between the FCFS sequence and that of the optimal makespan CPS sequence. The main observation is that a decrease in makespan is usually (and perhaps not surprisingly) accompanied by a decrease in average delay, which is a desirable by-product of the algorithm.

The run times of our algorithm are shown in Figure 8 (right), each data point being the average over 600 instances (100 instances each of three different arrival rates and two fleet mixes). The program was written in C and was run on a PC with a 2 GHz processor and 2 GB RAM.

## 9. Conclusion

We have developed a unified framework for runway scheduling under constrained position shifting and demonstrated that the problems of enhancing throughput, decreasing delay, and ensuring fairness can be effectively modeled

**Figure 7.** Percentage improvement in makespan using CPS over FCFS when fleet mix is [left] 40% heavy, 40% large, 20% small, and [right] 45% heavy, 45% large, 10% small.

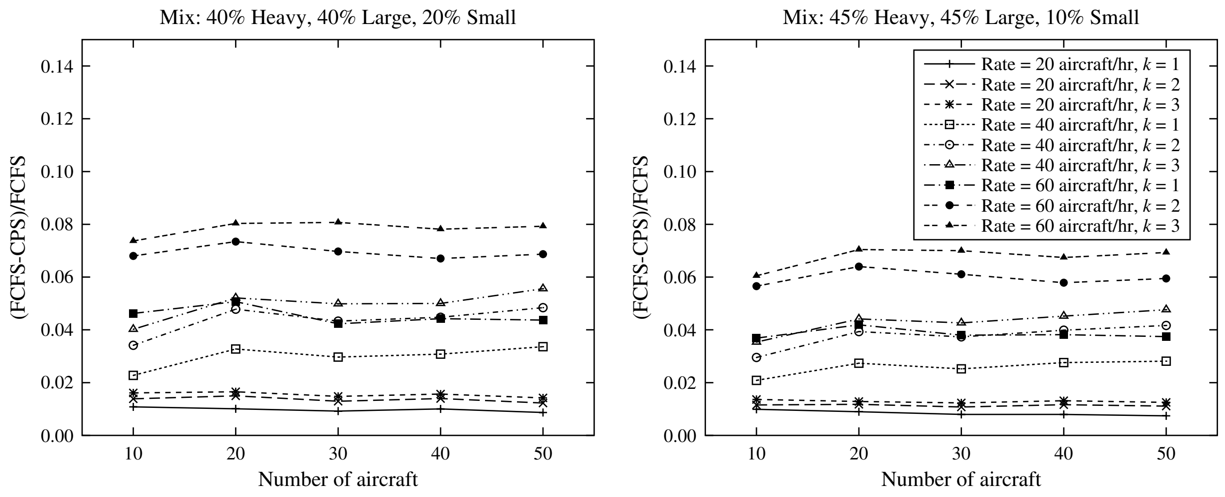
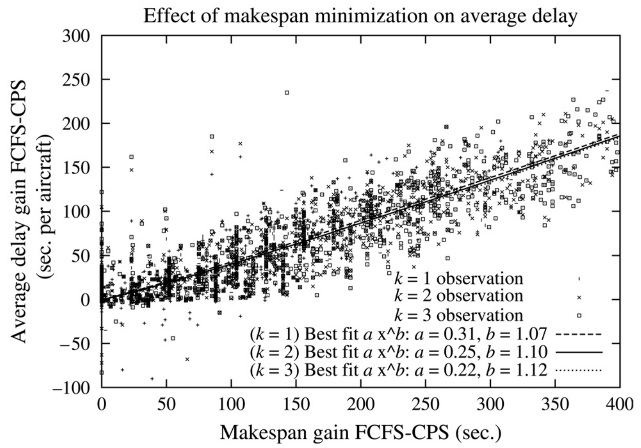


**Figure 8.** [Left] Comparison of gain in seconds of CPS over FCFS for average delay per aircraft vs. makespan in optimal makespan solution. [Right] Run times (sec.) for different values of $n$ and $k$.



| $n$ | $k=1$ | $k=2$ | $k=3$ |
|---|---|---|---|
| 10 | <0.001 | 0.001 | 0.017 |
| 20 | <0.001 | 0.004 | 0.236 |
| 30 | <0.001 | 0.006 | 0.449 |
| 40 | <0.001 | 0.009 | 0.698 |
| 50 | <0.001 | 0.012 | 0.914 |

and solved in polynomial time (linearly in the number of aircraft) while accounting for most operational constraints. We also extended the framework to include more general cost functions (by using discrete-time models) and to mixed arrival and departure operations, including the merging of multiple departure queues. The algorithms can be easily implemented, and a prototype implementation for arrival scheduling demonstrates that the run times are sufficiently small to enable real-time deployment.

## Acknowledgments

This research was supported by NSF contract ECCS-0745237 and NASA contracts NNA06CN24A and R8771-G1. The authors thank the anonymous reviewers for their insightful comments.

## References

Anagnostakis, I., J.-P. Clarke, D. Böhme, U. Völckers. 2001. Runway operations planning and control: Sequencing and scheduling. *J. Aircraft* **38**(6) 988–996.

Anagnostakis, I., H. R. Idris, J.-P. Clarke, E. Feron, R. J. Hansman, A. R. Odoni, W. D. Hall. 2000. A conceptual design of a departure planner decision aid. 3rd USA/Europe ATM R&D Seminar, Naples, Italy.

Arkind, K. D. 2004. Requirements for a novel terminal area capacity enhancement concept in 2022. *American Institute of Aeronautics and Astronautics, Guidance, Navigation and Control Conf. Exhibit, Providence, RI.* American Institute of Aeronautics and Astronautics, Reston, VA.

Atkins, S., C. Brinton. 2002. Concept description and development plan for the surface management system. *J. Air Traffic Control* **44**(1).

Beasley, J. E., M. Krishnamoorthy, Y. M. Sharaiha, D. Abramson. 2000. Scheduling aircraft landings—The static case. *Transportation Sci.* **34**(2) 180–197.

Boehme, D. 1994. Improved airport surface traffic management by planning. H. Winter, H.-G. Nüsser, eds. *Advanced Technologies for Air Traffic Flow Management. Lecture Notes in Control and Information Science 198*. Springer, Berlin, 191–224.

Böhme, D. 2005. Tactical departure management with the Eurocontrol/DLR DMAN. 6th USA/Europe ATM R&D Seminar, Baltimore.

Bureau of Transportation Statistics. 2007. Accessed August 2008, http://www.bts.gov.

Carr, F. R. 2004. Robust decision-support tools for airport surface traffic. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Cormen, T. H., C. E. Leiserson, R. L. Rivest. 1990. *Introduction to Algorithms*. MIT Press—McGraw-Hill, Cambridge, MA, 536.

de Neufville, R., A. R. Odoni. 2003. *Airport Systems: Planning, Design and Management*. McGraw-Hill, New York.

Dear, R. G. 1976. The dynamic scheduling of aircraft in the near terminal area. MIT Flight Transportation Laboratory Report R76-9, Massachusetts Institute of Technology, Cambridge.

Dear, R. G., Y. S. Sherif. 1991. An algorithm for computer assisted sequencing and scheduling of terminal area operations. *Transportation Res., Part A, Policy Practice* **25**(2–3) 129–139.

Federal Aviation Administration. 2004. Airport capacity benchmark report. http://www.faa.gov/about/office_org/headquarters_offices/ato/publications/bench.

Federal Aviation Administration. 2006. Air traffic control: Order 7110.65P. Incl. Change 1, effective 8/3/06. Accessed October 2006, http://www.faa.gov/air_traffic/publications/atpubs/atc/.

Federal Aviation Administration. 2009. Accessed October 2009, http://aspm.faa.gov/main/opsnet.asp.

Idris, H. R., B. Delcaire, I. Anagnostakis, W. D. Hall, N. Pujet, E. Feron, R. J. Hansman, J.-P. Clarke, A. R. Odoni. 1998. Identification of flow constraint and control points in departure operations at airport systems. *American Institute of Aeronautics and Astronautics, Guidance, Navigation and Control Conf., Boston, AIAA-1998-4291*. American Institute of Aeronautics and Astronautics, Reston, VA.

Lee, H. 2008. Tradeoff evaluation of scheduling algorithms for terminal-area air traffic control. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Neuman, F., H. Erzberger. 1991. Analysis of delay reducing and fuel saving sequencing and spacing algorithms for arrival spacing. NASA technical report A-91203; NAS 1.15:103880; NASA-TM-103880. Retrieved August 2008, NASA Technical Reports Server (NTRS).

Psaraftis, H. N. 1980. A dynamic programming approach for sequencing groups of identical jobs. *Oper. Res.* **28**(6) 1347–1359.

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

Willemain, T. R., H. Fan, H. Ma. 2004. Statistical analysis of intervals between projected airport arrivals. DSES Technical Report 38-04-510, Rensselaer Polytechnic Institute, Troy, NY.