Flight sequencing and scheduling:
A data-driven approach

Jitamitra Desai* and Rakesh Prakash
School of Mechanical and Aerospace Engineering
Nanyang Technological University
50 Nanyang Avenue, Singapore 639798
* Corresponding Author: jdesai@ntu.edu.sg

Abstract

This research addresses the static aircraft sequencing and scheduling problem under a mixed-mode, single runway operating scenario. This problem is formulated as a 0-1 mixed-integer programming problem, taking into account several realistic constraints, including safety separation standards and airline equity considerations, with the objective of minimizing the total delay in the system. Recognizing the computational difficulty involved in solving large-scale instances by directly employing the MIP model, in our second thrust, a novel data-driven splitting algorithm is postulated, wherein the given set of flights is split into two distinct sets, and the optimization formulation is individually applied on these two smaller data sets, while taking into account the effect of the leading set of flights onto the following set. Computational results on illustrative examples show that the overall time taken to solve large-scale (realistic) instances can be reduced significantly, while achieving the optimal solution in nearly all of the instances.

Keywords
Air traffic management; aircraft sequencing problem; 0-1 mixed-integer programming; data-splitting algorithm; constrained position shifting; flight delays.

1. Introduction

Flight delays pose a very serious and widespread problem, and with rising air traffic demand, ever-increasing flight delays have placed a significant stress on the entire Air Traffic Management (ATM) system, costing airlines, passengers, and the overall economy several billions of dollars each year. The critical bottleneck of an ATM system is the capacity within a radius of about 50 nautical miles (nm) from an airport, namely the Terminal Maneuvering Area (TMA). In the United States, terminal area congestion accounted for 13% of all delays in 2005 and this number has been steadily increasing every year, accounting for nearly 21% of delays in 2013 (1).

According to [2], significant improvements can be achieved by optimizing critical bottleneck operations within the TMA related to arrivals, departures, runways, and taxiways. One such aspect is the joint sequencing and scheduling of arriving and departing aircraft, which is commonly referred to in the literature as the Aircraft Sequencing Problem (ASP). Specifically, in the static version of this problem; given a set of aircraft, along with information on the earliest/latest operation time for each aircraft (be it an arrival or a departure), and the minimum safety regulations to protect trailing aircraft from wake vortices generated by leading aircraft; the objective is to determine the optimal sequence that maximizes runway throughput or minimizes the total delay in the system, when operating under a mixed-mode of operations. Usually, air traffic controllers follow the first-come first-serve (FCFS) rule for sequencing flight arrivals and departures based on their estimated time of arrival or departure (ETA or ETD) on the runway. But, it is well-known that such a sequencing rule is very inefficient in practice as it induces a lot of delay into the system (3). This necessitates the development of computationally efficient scheduling models and algorithms, but the challenge lies in simultaneously achieving safety, efficiency, and equity in the ATM system, where: (i)
safety is achieved by maintaining the required separations between aircraft; (ii) efficiency is equivalent to achieving low average delay or high throughput, and; (iii) airline equity is modeled by implementing a constrained position shifting (CPS) strategy wherein an aircraft cannot be shifted by more than \( k \) positions (the so-called maximum position shifting (MPS) parameter) from its initial FCFS-based position. The equivalence of this ASP to the NP-Hard Traveling Salesman Problem (TSP) and job shop scheduling problem with sequence dependent set-up times ([5]) is well-known in the literature, and hence obtaining real-time optimal solutions is a challenging task.

There are many variations of the ASP that can be found for the single runway case depending upon the mode of runway operations (segregated or mixed); problem objectives (minimizing delay or maximizing throughput); and constraints such as inclusion of time windows, permissibility of early landings, and CPS requirements. Moreover, in scouring exact approaches for solving this problem, we found that LP-based branch-and-cut algorithms ([6, 7]) have mostly been used when the CPS constraint is ignored, whereas dynamic programming ([4, 8, 9]) has been the preferred approach to solve the ASP with CPS constraints included in the formulation. Furthermore, the efficiency of determining an optimal solution was also found to be heavily dependent on the structure of the arrival and departure time-windows ([10]).

As stated in various instances in the literature, problem scenarios with wide time-windows and an objective of minimizing delay are the hardest to solve, even for cases where the search space is curtailed by the CPS constraint. This is because tighter time-windows can be leveraged to enforce precedence relationships between leading/following aircraft and this advantage is lost when dealing with wider time-windows. In our problem formulation of the ASP on a single runway, we considered the most difficult version of this problem, which comprises a mixed-mode of operations, while incorporating all practical constraints including CPS, wide time-windows, early landings/departures, and the objective of minimizing total delay. To the best of our knowledge, no efficient solution exists for solving the ASP under the aforementioned scenarios.

In this paper, we begin by formulating the ASP as a 0-1 mixed-integer program (MIP), which is an adaptation of the model in [6] enhanced with the inclusion of the CPS constraints and having a modified objective of minimizing the total delay in the system. Recognizing that solving this model is not computationally viable for large-scale instances, the main thrust of this paper postulates a novel data-splitting algorithm, which we refer to as DS-ASP, that optimizes flight sequences by a repeated application of this 0-1 MIP on smaller data sets, and demonstrates the efficacy of this approach.

The remainder of this paper is organized as follows. In Section 2, we present the 0-1 MIP formulation for the ASP along with some preliminary computations. Next, in Section 3, the details of the data-splitting algorithm, and its pseudo-code are described. Then, Section 4 demonstrates some computational results for large-scale (realistic) instances, and finally, Section 5 summarizes the contributions of this work and suggests extensions for future research.

2. Optimization Problem

We are now ready to formulate the ASP as a 0-1 mixed integer program, as detailed below.

Description of Index Sets and Parameters
- \( \mathcal{F} \) : Set of all arriving and departing flights
- \( E_i \) : Earliest time of arrival (departure) of aircraft \( i \)
- \( T_i \) : Target time of arrival (departure) of aircraft \( i \)
- \( L_i \) : Latest time of arrival (departure) of aircraft \( i \)
- \( \Delta s_{ij} \) : Required safety separation (in seconds) at runway threshold, if flight \( i \) is ahead of flight \( j \)
- \( \text{seq}_i \) : Position of flight \( i \) based on the FCFS sequence
- \( k \) : Specified maximum position shifting (MPS) parameter

Decision Variables
- \( x_{ij} \) = \begin{cases} 1, & \text{if flight } i \text{ is ahead of flight } j \text{ in sequence} \\ 0, & \text{otherwise} \end{cases}
- \( t_i \) = Scheduled time of arrival (departure) of flight \( i \)
\textbf{ASP:} \quad \text{Minimize} \quad \sum_{i,j \in \mathcal{F}} \left| t_i - T_j \right| \quad (1a)

subject to:

\begin{align*}
  x_{ij} + x_{ji} &= 1, \quad \forall i < j, \quad (i, j) \in \mathcal{F} \\
  t_j &\geq t_i + \Delta s_{ij} - M(1 - x_{ij}), \quad \forall (i, j) \in \mathcal{F} \\
  -k &\leq (F - \sum_{j \in \mathcal{F} \setminus \{i\}} x_{ij}) - \text{seq}_i \leq k, \quad \forall i \in \mathcal{F} \\
  E_i &\leq t_i \leq L_i, \quad \forall i \in \mathcal{F} \\
  x_{ij} &\in \{0, 1\}, \quad \forall (i, j) \in \mathcal{F}; \quad t_i \geq 0, \quad \forall i \in \mathcal{F}. \quad (1f)
\end{align*}

In the above formulation, the objective function (1a) seeks to minimize the total delay; constraint (1b) enforces the order precedence relationship between flights $i$ and $j$; constraint (1c) ensures the time-based separation requirements between flights at the runway (as specified by the FAA for various flight classes), where $M \equiv (L_i + \Delta s_{ij} - E_j)$; constraint (1d) imposes the CPS constraint that an aircraft cannot be shifted by more than $k$ positions from its initial (FCFS) position, where $F \equiv |\mathcal{F}|$; constraint (1e) maintains the scheduled time of arrival (departure) at the runway to be between the earliest and latest times for each aircraft; and finally constraint (1f) imposes binary and non-negativity restrictions on the $x$- and $t$-variables, respectively. Note that the ASP formulation described above can be further improved by using variable-fixing strategies and addition of valid inequalities that serve to tighten the underlying linear programming representation \cite{7,11}. For the sake of brevity, we only present variable fixing strategies based on the CPS constraint and the well-known earliest landing time window (ELW) rule here, and ignore other model enhancements in our discussion below.

\textbf{Proposition 1.} For a pair of aircraft $(i, j) \in \mathcal{F}$, if the FCFS sequence positions of $i$ and $j$ satisfy the condition: $\text{seq}(j) - \text{seq}(i) \geq 2k$, where $k$ is the MPS parameter, then we can fix $x_{ij} = 1$. \hfill \Box

\textbf{Proposition 2.} (ELW Rule) Consider a pair of aircraft $(i, j) \in \mathcal{F}$ that belong to the same category. Suppose that the time-window restrictions satisfy the following conditions: (i) $E_i \leq E_j$; (ii) $T_i \leq T_j$; and (iii) $L_i \leq L_j$; then we can fix $x_{ij} = 1$. \hfill \Box

To gain an understanding of the computational difficulties involved in solving Problem ASP, we tested the optimization model, given by (1a)-(1f), on several randomly generated large-scale (realistic) instances. From the results recorded in Table 1 it can be clearly observed that the computational time increases exponentially, particularly for cases where the maximum position shifting parameter $k \geq 2$, where TL denotes the solver time limit, which is set as 1800 seconds. Motivated by the lack of responsiveness of this model in handling large-scale instances, in this paper, we postulate a novel data-splitting algorithm, which involves repeatedly solving Problem ASP on relatively smaller data-sets, and can solve realistic instances in an efficient manner to enable real-time implementation in practice.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Instance & # of flights & Mode of Operation & MPS (s) & \\
\hline
1 & 30 & Arrival & T & 5.16 & 688 & TL \\
2 & 30 & Arrival & 3 & 138 & 1780 & \\
3 & 40 & Arrival & 6.63 & TL & TL & \\
4 & 40 & Arrival & 4.66 & TL & TL & \\
5 & 40 & Mixed & 2.2 & TL & TL & \\
6 & 40 & Mixed & 5.25 & 964 & TL & \\
\hline
\end{tabular}
\caption{Computational times based on a direct implementation of Problem ASP on large-scale instances}
\end{table}

\section{3. Algorithm}

The proposed data-splitting algorithm (\textbf{DS-ASP}) essentially involves dividing the original flight data-set into several possible pairs of equal sets of leading and following aircraft, then determining the optimal solution for each pair, and finally, comparing these solutions to get the overall optimum. The structure of this data-splitting algorithm has been exploited on the basis of three key observations: (i) Given two sets of aircraft, each of size $m$, under the condition that one set has to necessarily follow the other, determining the optimal solution for each set separately, while accounting for the effect of the resulting schedule on their respective objective functions, is always computationally more efficient; (ii) Our empirical observations on realistic flight data also revealed that, in an optimal sequence, at least half of the aircraft land/depart after their respective target time of arrival/departure, thereby allowing for easier accounting of the effect of the leading set of flights onto the following set; and (iii) As a result of the CPS constraint, the number of pairs
of leading and following data-sets are very limited, i.e., of the order of \( \binom{2k}{k} \), where \( k \) is value of the MPS parameter, and is independent of the total number of flights \( n = 2m \). (Throughout this discussion, we are restricting ourselves to realistic instances of flight data with number of flights \( \geq 30 \), average inter-arrival times of no more than 70 seconds, an aircraft mixture containing a high percentage of heavy and large aircraft (\( \geq 80\%) \), and wide time-windows.)

Note that both the aircraft type and arrival time of the last flight in the leading set affect the overall delay because aircraft in the following set have to maintain the required separation from this last aircraft, and a constraint on the arrival time in turn affects the delay for each individual set. Hence, in order to account for such phenomenon, the number of pairs are further enlarged by fixing all possible aircraft that can be positioned last in sequence and a constraint is added to the following set to ensure that all the flights arrive after the landing time of the last aircraft in the leading set. Furthermore, to account for the effect of the landing time of this last aircraft on the delay of the leading and following sets, the objective functions of both sets are modified accordingly. A more detailed description of the algorithmic stages is given below, where we first describe the procedure for only arriving traffic and subsequently extend it for the case of mixed traffic.

3.1 Algorithmic outline for arriving traffic

**Stage A: Preparation of instance pairs**

Let \( n = 2m \) denote the number of aircraft positioned at \( \{1, ..., m - 1, m, m + 1, ..., n\} \) in FCFS order. After splitting this flight data into two equal halves, the leading set \( A \) and following set \( B \) are composed of aircraft at positions \( \{1, ..., m\} \) and \( \{m + 1, ..., n\} \), respectively. Owing to the CPS constraint, as only aircraft between \( (m - k + 1) \) and \( (m + k) \) positions can crossover between the leading/following sets (see Proposition 1), there are only \( \binom{2k}{k} \) such pairs. Furthermore, observing that only aircraft between positions \( (m - k) \) to \( m \) can occupy the last position in set \( A \), each pair \( (A, B) \) results in additional combinations, denoted as \( (A', B) \), which will finally be used by the algorithm. This pairing scheme is illustrated in the example below.

Consider an instance of eight aircraft \( \{H1, H2, H3, LA, L5, H6, S7, S8\} \), where each aircraft is represented by its aircraft type (\( H \): Heavy, \( L \): Large, \( S \): Small) and its relative position in the FCFS sequence. Assuming \( k = 2 \), there are \( \binom{4}{2} = 6 \) resulting pairs \( (A, B) \), which are given by:

1. \( A = \{H1, H2, H3, LA\}, B = \{L5, L6, S7, S8\} \)
2. \( A = \{H1, H2, H3, L5\}, B = \{L4, H6, S7, S8\} \)
3. \( A = \{H1, H2, H3, H6\}, B = \{L4, L5, S7, S8\} \)
4. \( A = \{H1, H2, LA, L5\}, B = \{H3, H6, S7, S8\} \)
5. \( A = \{H1, H2, LA, H6\}, B = \{H3, L5, S7, S8\} \)
6. \( A = \{H1, H2, L5, H6\}, B = \{H3, L4, S7, S8\} \)

For each of these listed pairs \( (A, B) \), permuting flights within set \( A \), predicated on the aircraft occupying the last position in this set, results in several additional combinations, which we denote as \( (A', B) \). As an example, for the case: \( A = \{H1, H2, H3, LA\} \) and \( B = \{L5, H6, S7, S8\} \), such permutations within set \( A \) result in the following three pairs:

- \( A' = \{H1, H2, LA, H3\}, B = \{L5, H6, S7, S8\} \)
- \( A' = \{H1, H2, H3, L4\}, B = \{L5, H6, S7, S8\} \)
- \( A' = \{H1, H3, LA, H2\}, B = \{L5, H6, S7, S8\} \)

Note that, when \( A = \{H1, H2, H3, H6\} \) and \( B = \{LA, L4, S7, S8\} \), as aircraft \( H6 \) cannot be scheduled before position 4 in the sequence (as this would violate the CPS constraint), no further pairings are feasible in this case. Furthermore, if all the aircraft satisfy the ELW rule (see Proposition 2), then some of the pairs, e.g., \( A' = \{H1, L3, H4, H2\} \) and \( B = \{L5, H6, S7, S8\} \), can be discarded.

**Stage B: Solving the instance pairs**

To solve the optimization problem related to each pair, we begin by calculating the effect of the leading set of flights onto the following set. Considering that all aircraft in the following set arrive after their target time (which is a realistic assumption for practical instances), then any change in the throughput (arrival time of the last flight) of the leading set impacts the delay of the following set, i.e., the dependency of delay of the following set on the completion time \( z \) of the leading set is equal to \( |B| z \), where \( |A| \) denotes the cardinality of set \( A \). Note that, in certain cases, while a higher completion time of set \( A' \) may result in lower delay for set \( A' \), it may result in a higher delay for set \( B \), and vice-versa, and therefore, accurately capturing the effect of throughput is key to determining the optimal solution using the proposed algorithm. Hence, to measure such deviations in throughput (and consequently delay), we modify the objective function for flights in set
B by introducing two non-negative continuous indicator variables \(\delta_1\) and \(\delta_2\), which take on a value of zero if no difference in the carry-over effect is detected, thereby confirming optimality. A formal statement of the pseudo-code of the proposed data-splitting algorithm is given below.

**Algorithm 1** Pseudo-code for the proposed data-splitting algorithm.

1. Set iteration counter \(p \leftarrow 0\), \(OptVal^{(p)} \leftarrow +\infty\), incumbent \(\leftarrow OptVal^{(p)}\), \(S = \{\text{set of all pairs } (A', B)\}\). Go to Step 2.
2. If \(S = \emptyset\), stop; return incumbent as best solution. Else set \(p \leftarrow p + 1\), and go to Step 3.
3. Arbitrarily select one instance pair \((A'_p, B_p)\) from \(S\). Go to Step 4.
4. Solve ASP for set \(A'_p\) with \(OptVal^{(p)}_{A'_p} = \min \left\{ \sum_{i \in A'_p} (|t_i - T_i|) + |B_p| \epsilon : (1a)-(1f), \epsilon \geq t_i, \forall i \in A'_p \right\}\). Go to Step 5.
5. Solve ASP for set \(B_p\) with
   \[
   OptVal^{(p)}_{B_p} = \min \left\{ \sum_{i \in B_p} (|t_i - T_i|) + |B_p| \epsilon (\delta_1 + \epsilon (\delta_1 + \delta_2)) : (1a)-(1f), t_i \geq t_i + \Delta s_{li} + \delta_1 - \delta_2 \right\},
   \]
   where \(i \in B_p\) and \(l\) is the last aircraft in \(A'_p\), and \(\epsilon\) is a preset tolerance. Go to Step 6.
6. If \(\delta_1 > 0\) or \(\delta_2 > 0\), print (“optimality can’t be concluded”) \textbf{break}; else, go to Step 7.
7. If \(\delta_1 = 0\) and \(\delta_2 = 0\), concatenate the optimal sequences obtained in Steps 4 and 5, set \(OptVal^{(p)} \leftarrow \sum_{i \in A'_p, B_p} (|t_i - T_i|)\). Go to Step 8.
8. If \(OptVal^{(p)} < \text{incumbent}\), set incumbent \(\leftarrow OptVal^{(p)}\), Update \(S \leftarrow S \setminus \{A'_p, B_p\}\). Go to Step 2.

### 3.2 Extending the algorithm to mixed traffic

When dealing with the segregated case of only arriving traffic, as the triangular inequality with respect to the safety separation distance matrix is always satisfied, it is sufficient to maintain the separation of the first aircraft in set \(B\) from the last aircraft in \(A'\). But in the case of mixed traffic, this inequality may get violated for some instances, depending upon the mixture of traffic. Hence, in order to account for such violations, we would need to add additional (case-specific) set pairs \((A', B)\), along with the necessary non-triangular separation constraints.

### 4. Computational Results

The proposed data-splitting algorithm was tested on various randomly generated instances comprising thirty or forty aircraft, under both arrival and mixed-mode traffic scenarios. The traffic mixture for these instances was set as: 40% Heavy + 40% Large + 20% Small aircraft, and the target times of aircraft were also randomly generated, assuming that each aircraft appears every \(\gamma\) seconds, where \(\gamma\) was set as either 60 or 70 seconds. We also assume that a flight can arrive or depart up to 60 seconds earlier and no more than 1800 seconds later than its scheduled target time. All of our computations are performed on a Windows machine, equipped with a 4th Gen Intel Core i3-4030U 1.9GHz 4GB RAM processor, using MATLAB R2011b in conjunction with GuRoBi 6.0.4 as the MIP-solver. The time-limit (TL) was once again set as 1800 seconds, and the \((\ast)\) indicates that the LB-UB gap at the time of termination was greater than 60%.

From the results recorded in Table 2 on average, we can observe more than a 2454% reduction in the computational time required to solve all randomly generated instances by using the proposed DS algorithm compared to solving the original ASP formulation (1a)-(1f) using the commercially available solver GuRoBi. This tremendous reduction in computational effort can be attributed to two reasons: (i) a significant reduction in search space; and (ii) a difference in solver behavior when solving large-scale versus small-scale instances, with the solver being able to close the gap more efficiently for smaller instances. Moreover, the data-splitting algorithm was able to conclude optimality in all instances of arrival traffic, but it fails to converge to optimality for some instances of mixed traffic, usually for the case of \(k = 3\). The reason for lack of convergence of the algorithm in these instances lies in the interaction between \(\gamma\) and the required time-separations. If the time-separation matrix allows for aircraft to be placed too closely in the range of \(\gamma\) (such as is the case for mixed traffic), then our assumption that all aircraft in the following set arrive after their respective target times may not remain valid, thereby affecting the objective functions.
Table 2: Computational time comparison between original MILP model and data splitting method

<table>
<thead>
<tr>
<th>Mode of Operation</th>
<th>F = 30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>ASP</td>
<td>DS-ASP</td>
</tr>
<tr>
<td>Arrival</td>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>3</td>
</tr>
<tr>
<td>Arrival</td>
<td>70</td>
<td>8</td>
</tr>
<tr>
<td>Arrival</td>
<td>70</td>
<td>4.80</td>
</tr>
<tr>
<td>Mixed</td>
<td>60</td>
<td>0.71</td>
</tr>
<tr>
<td>Mixed</td>
<td>60</td>
<td>0.73</td>
</tr>
<tr>
<td>Mixed</td>
<td>70</td>
<td>0.39</td>
</tr>
<tr>
<td>Arrival</td>
<td>60</td>
<td>0.14</td>
</tr>
</tbody>
</table>

5. Conclusions

In this paper, we proposed a novel data-splitting algorithm, which provides real-time optimal solutions for the aircraft sequencing problem with the objective of minimizing total delay under segregated and mixed traffic conditions. The performance of the DS algorithm clearly demonstrates the algorithmic speed-up that can be obtained, while achieving optimality (or near-optimality) in all practical test-bed instances. Our computational experience also demonstrates for the first time in the literature that, on average, an LP-based branch-and-cut approach, applied on the above model, outperforms dynamic programming-based approaches under wide time-window scenarios. Accounting for wide time-windows and early landings results in an exponential number of states when using a dynamic programming approach, whereas the pruning of the search space by utilizing the data-splitting algorithm results in a dramatic reduction in the computational times required to determine the optimal solutions. We are also in the process of further improving the efficiency of the algorithm, notably by reducing the number of instance-pairs through model-enhancements, pruning techniques, and associated flight data analysis.

Acknowledgment: This research has been partially supported under ATMRI (NTU-CAAS) Grant No. M4061216.

References