31st Congress of the International Council of the Aeronautical Sciences
Belo Horizonte, Brazil; September 09-14, 2018

ADAPTIVE LANDING SEQUENCING USING BIPARTITE GRAPHS AND PERFECT MATCHING

Matthew Hickson , Cees Bil
School of Engineering, RMIT University, Melbourne, Australia

Keywords: *aircraft landing problem, bipartite graph, runway engagement time*

Abstract

An optimal aircraft landing sequence is a sequence that results in the lowest Runway Engagement Time (RET). The approach presented in this paper uses a generic arrival fleet analyser, a cluster finding algorithm to isolate independent groups and a permutation enumeration algorithm to determine valid landing sequences. The optimal sequence of each cluster is determined and is used to develop a global optimal sequence for arriving aircraft. The model was validated by simulating 24-hour arrival periods with varying arrival frequency distributions. The results show that by exploiting the clustering effect, computational time can be conserved, and RET can be reduced by several minutes over a 24-hour period.

1 Introduction

Due to increasing number of flights, en-route and airport traffic control has become central to improving the efficiency of aircraft operations [1]. Improvements have been made to en-route air traffic management and the primary limiting factor to air traffic capacity has shifted from airspace to airports (Figure 1). To improve airport air traffic management, specifically the Aircraft Landing Problem (ALP), the development and implementation of a dynamic landing sequencer is required to improve productivity. The cost of delaying an aircraft in the air is about twice as much as on the ground [2]. However, nearly all airports predominately use First Come First Served (FCFS) sequencing, where aircraft land as determined by their proximity and scheduled arrival. This sequencing is popular due to simplicity, reduction in controller workload and sense of fairness, but it can lead to reduced runway throughput [3].

Fig. 1. Aircraft lining up for landing at Heathrow.

Efforts have been focused on automated computer aided scheduling, trying to minimise RET by examining possible landing sequences and determining the most favourable one. To create an optimal, but fair, landing sequence, aircraft preferences and cost incurred by each stakeholder, must be considered, as aircraft often resist sequence changes due to the perceived unfairness. Therefore, optimal models that rely on aircraft cooperation are difficult to implement and situations with fixed aircraft positions can occur. Allowable position shifts for each aircraft provide a raw data set that can be parsed into an algorithm for optimal sequencing. Adaptive sequencing requires a robust model that can accommodate unexpected perturbations. The aircraft landing problem is an NP-hard problem, thus the possibility of enumerating all possible sequences in polynomial time is currently infeasible. This means time sensitive decision making requires a fast algorithm for computing optimal sequences.

Several strategies have been suggested to solve the ALP, that either involve position shifting managed independently by the arrival airport or a Collaborative Decision Making (CDM) process that controls departure times and arrival times. Changing the landing position outside the FCFS order will incur a cost if aircraft are consequently delayed. To determine the most optimal scheduling sequence, all potential sequences must be examined [4]. Implementing Constrained Position Shifting (CPS) programs drastically reduces computation time by limiting the search to fewer sequences, by limiting allowable aircraft repositions to only a few places from their original sequence. There are limitations to this strategy related to compliance of aircraft positioning and unexpected disturbances. To account for this, a CPS model was created that incorporates aircraft precedence relationships, air traffic control over-taking procedure restrictions and specified time windows [5]. This model uses a tree search algorithm to determine the highest runway throughput. Applying the technique to existing schedule data, showing significant efficiency improvements as the allowable repositions increased and the constrained sequences were nearly identical to the optimal ordering. This research is most promising for standalone ALP solvers, however unexpected disturbances in scheduling and inadequate compliance reduces the effectiveness of the model.

For CPS models, Dear used an algorithm that enumerates all possible sequences and determines the RET for each, which, as the number of aircraft increases became impractical [4]. It was improved by introducing allowable time windows, their algorithm was significantly faster through reducing the number of possible sequences due to their constraints [6]. CPS models still lack robustness in the event of disturbances and become slower as the number of aircraft increases. Pre-emptive node generation by limiting the number of sequences through tree pruning and constraint application, shows that CPS models can scale linearly [3].

For any strategy, the willingness of the airlines to agree to variations in their preferred scheduling is assumed, so fairness of the order in which aircraft land is an important factor. To uphold stakeholder satisfaction, CDM programs are suggested that involve airlines in the decision-making process and give them flexibility to satisfy their priorities [7]. CDM utilises variable take-off times to ensure improved runway throughput upon arrival [8,9].

Another suggestion is using arrival timeslot auctions, giving flexibility to stakeholders [10]. Due to cost incursions, the landing sequence will naturally develop towards more efficient sequencing and airline satisfaction will remain high. The scheme requires a weighting and scoring system that accurately determines bidding prices.

Terrab and Odoni focused on airport capacity as the primary limiter to cost effective sequencing, suggesting two models that prioritise ground holding before take-off [11]. One model idealises the problem by constraining aircraft to their predefined flight times and prioritises aircraft with the highest marginal cost of delay, while the other model presents a probabilistic dynamic approach to scheduling by using a stochastic model that improves in accuracy over time as the capacity for each airport becomes clearer.

Applying ground holding requires dynamic programming for multiple airports, which increasing the data required to determine optimal schedule planning. Determining an optimal computing method depends on complexity and the desire for real time results. This model was applied to minimise overall cost of delay whilst attempting to alter original scheduling as little as possible [14]. They provided a heuristic based approach that computes results in real time although derives suboptimal paths due to this limitation. They applied their model to existing scheduled data, only constraining the airport capacity at a given time interval and the allowable number of delays before cancellation of the flight occurred. Ground holding and ‘fixes’ are a common occurrence in schedule recovery. Tactical Air Traffic Management is crucial for efficient and safe operations, but is not sufficient as schedules can change due to bad weather [18].

The model presented in this paper incorporates the common detriment to other strategies by utilising low compliance and the resulting clustering effect to compute optimal sequences in real time for a generic arriving fleet. This paper presents a mathematical representation of the proposed solution and an algorithm for enumerating optimal sequences, by adopting strategies exploiting Constrained Position Shifting (CPS), CDM, their fairness models and the computational techniques used.

2 Problem Modelling

For a given set of aircraft, RET is the time from landing of the first aircraft to the time of landing of the last aircraft in a sequence. The default landing order of arriving aircraft can be determined by their scheduled arrival time. This information provides a control comparison for RET for different sequences. The scheduled sequence is prone to changes as FCFS landing sequences are typically determined by the actual order of arrival.

For any given aircraft, the allowable positions indicate at what slots in the landing sequence it can be placed. The time of this arrival will vary slightly depending on the aircraft landing before it and it is assumed the aircraft can land as soon as it is required to. The allowable positions of an aircraft will default to its FCFS order and allowable position shifts are dependent on other aircraft inside the ARTCC boundary limits. In each time interval the number of position swaps will also depend on how many aircraft are within the boundary and the level of compliance of aircraft cooperating with landing sequence alterations. The approach presented in this paper represents allowable positions as a set of positions including the original position i, e.g. $\mathrm{P}(A_i) = \{i-2, i-1, i, i+1\}$, with each aircraft having its own set of allowable positions. No overtaking is allowed for aircraft on the same path with allowable speed variations having the most influence on the allowable position re-assignments [12].

ICAO outlines aircraft approach procedures, which influence the landing precedence constraints for arriving aircraft. The simplified version of the procedures utilises jet routes to determine aircraft precedence, while advanced versions also utilise altitude management and Vertical Guidance (APV). Civil aviation authorities enforce minimum wake separation between successive aircraft, dependent on the weight class of the two aircraft. For aircraft combinations between lower and higher or equal weight classes regulations use a distance requirement of 3 nm, except in the case of heavy trailing heavy scenarios, which use a minimum distance separation of 4 nm. To simplify this distance requirement, ICAO minimum separation time of 120 seconds is used in lieu of these distances.

In certain scenarios, there are instances where an aircraft must land before another aircraft or group of aircraft. The precedence relationships of arriving aircraft are assigned to each aircraft and only sequences that satisfy these precedence relationships are considered valid. Each aircraft has its own precedence relationships represented by a set of values denoting the other aircraft it must be land before, e.g. $P_R(A_i) = \{A_j, A_k\}$, where j and k represent two different aircraft. In the event of no precedence relationships present, the set will be empty.

In summary, the problem can be defined as follows: Given a set of n aircraft, indexed by their scheduled or FCFS arrival order 1, 2, 3,.., n, each aircraft i containing their own information of allowed positions $P(A_i)$, weight class and precedence relationships $P_R(A_i)$. Using the separation matrix, denoted as I_s, determine an optimal sequence that minimises RET for the entire set. For simplicity, the aircraft are referred to as being in their indexed position relative to where appropriate in the original schedule or FCFS order. The value being minimised is RET or alternatively the arrival time of the final aircraft given a set of n aircraft. Each successive two entity sequence of aircraft are denoted as s_i, where i represents a given aircraft index. Applying minimum time separation between all successive aircraft pairs in a sequence gives the total time function:

$$T(s) = \sum_{i=1}^{n-1} s_i \qquad (1)$$

Using this time function, the total time for the default sequence, $1 \rightarrow n$, can be determined and used for comparison. Using the allowable position sets (A_i) of each aircraft, a table can be generated containing each aircraft and their allowed position assignments, as shown in Table 1. Using the allowable position sets, the problem can be expressed as a bipartite graph complete matching problem. This term relates to the mathematical problem of successfully assigning an edge connecting each vertex from one set of vertices to at least one vertex in a different set of vertices, where no edges share a vertex.

Table 1. Allowable position assignments.

Aircraft	Allowable Position Assignments
1	{1, 2}
2	{1, 2, 3, 4}
3	{2, 3, 4}
…	…
n - 1	$\{n-3, n-2, n-1\}$
n	$\{n-2, n-1, n\}$

The complete matching problem becomes: Let $G = (V = V_1 \cup V_2, E)$ be an undirected bipartite graph with vertex sets V_1 and V_2 and an edge set E composed of edges $V_1 \times V_2$. A perfect matching M is an edge set contained in G such that no two edges share the same endpoints and all vertices of G are connected to at least one other vertex not contained within its own vertex set. Let N be the number of perfect matchings produced by G, enumerate all unique perfect matchings M.

This problem is a research field in mathematics with current algorithms enumerating all perfect matchings in $O(\sqrt{(|V|)} * |E| + N * \log|V|)$ time, where V, E, and N denote the number of vertices, edges, and perfect matchings respectively. This algorithm currently serves as the best lower bound estimate for complexity when determining perfect matchings [13].

2.1 Bipartite Graph and Bi-adjacency Matrix

For any given set of aircraft, there is at least one complete matching, which is the default sequence of successive integers $1 \rightarrow n$. The number of perfect matchings increases exponentially as the number of possible position assignments increases. This implies that the problem of bipartite graph complete matching is NP-hard since it there is currently no known polynomial time solution. An example bipartite graph G is presented in Figure 2.

For any given bipartite graph, an edge between two vertices denotes an allowable assignment and a perfect matching M for n number of aircraft would contain an edge set beginning from every vertex in V_1 connecting to a distinct endpoint in vertex set V_2.

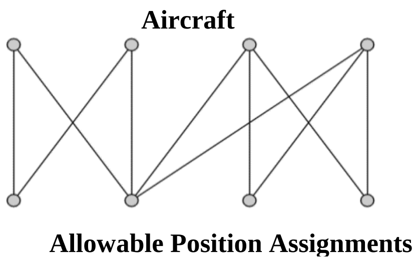


Fig. 2. Bipartite graph with four vertices, each edge denotes an allowable position assignment.

Using the assignment information from the bipartite graph, a bi-adjacency matrix can be generated. Let B be the matrix of size $\{n \ x \ n\}$, where every position denoted by B_{ij} is equal to 1 when an edge connects vertex i from set V_1 to j from set V_2 and 0 otherwise.

$$B = \begin{Bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{Bmatrix}$$

Fig. 2. Bi-adjacency matrix.

The permanent of a bi-adjacency matrix can be used to determine the number of permutations with restricted positions. In the case of bipartite graphs this is the number of perfect matchings. The permanent of matrix B is the number of complete matchings in bipartite graph G, which is a stop point for perfect matchings enumeration and prevents unnecessary sequence finding as once the number of perfect matches enumerated equals N, all matches have been found. Methods for exact permanent determination are dependent on the size and characteristics of the bi-adjacency matrix, but for a general (0,1) $\{n \times n\}$ matrix, Ryser's method is considered the best algorithm, operating in $O(2^{n-1} * n)$ time when processing in Gray code order [19].

Clusters are independent groups of permutations that do not influence each other and can be identified by analysing a bi-adjacency matrix and resolving smaller matrices within it.

$$B = \begin{Bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{Bmatrix}$$

Fig. 3. Bi-adjacency matrix with resolvable lower dimensional bi-adjacency matrices.

In Figure 3, lower dimensional bi-adjacency matrices can be observed in the top-left and bottom-right corners. Matrix B can be resolved into clusters since there are no edges connecting vertices between these two matrices. This is visually observable where minor $\{m \times m\}$ matrices are surrounded by zeros vertically and horizontally. The number of successful permutations for the major bi-adjacency matrix becomes the multiplication of the permanents of each minor matrix, considerably accelerates the algorithms designed to determine perfect matchings of an undirected bipartite graph.

2.2 Pruning

This stage reviews the active fleet data and searches for unachievable allowable positions due to restricting positions of other aircraft. For example, if aircraft 1 can only be allocated to landing position 1 and aircraft 2 can be allocated to both position 1 and 2, then, aircraft 2 cannot land in position 1 and therefore requires position 1 removed from its allowable position set. An example fleet flowchart is presented in Figure 4.

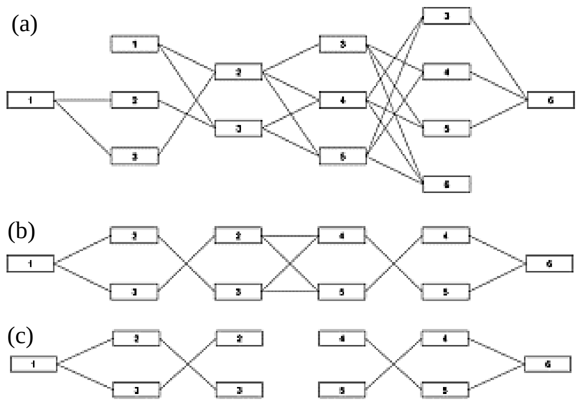


Fig. 4. Initial fleet flowchart (a), pruned fleet flowchart (b) and cluster search (c).

In Figure 4(a) there are allowable positions that are not valid and need to be removed. Pruning a system before permutation enumeration is beneficial since the amount of possible permutations enumerated and tested for validity is reduced. The total number of possible permutations before pruning is the product of the allowable number of aircraft. For (a), the total number of permutations with duplicates is $1 * 3 * 2 * 3 * 4 * 1 = 72$, which is not the number of perfect matchings since many of those permutations contain repeated aircraft entities. Pruning the system before enumerating the sequences will minimise the number of computations required. For example, by pruning the amount of possible permutations, even with repetition, is reduced to $1 * 2 * 2 * 2 * 2 * 1 = 16$. This number still does not represent the number of perfect matchings, but it has reduced the number of permutations required to test for.

2.3 Clusters

Clusters are isolated sections of a fleet that do not influence the possible sequences of each other. Clusters are a group of positions, e.g. $\{2 \rightarrow 4\}$ that contains several unique entities equal to the amount of positions contained with the group. Using Fig. 4 as an example, there are two clusters present, positions $\{1 \rightarrow 3\}$ and $\{4 \rightarrow 6\}$, each containing 3 landing positions and 3 unique entities. Isolating these two clusters reduces the number of permutations that need to be examined for each cluster. The number of possible permutations is only $1 * 2 * 2 = 4$ and $2 * 2 * 1 = 4$, for each cluster respectively. Thus, the amount of permutations that need to be tested for validity is only 8 and the resulting number of successful permutations only needs to be multiplied by each other to generate the total number of perfect matches. Using the perfect matches of each cluster and sequencing the matches together to generate all permutations will produce a final set list of valid sequences.

Using the example flowchart, the process of pruning the original fleet and separating the fleet into clusters shows a fast-logical process for generating all possible sequences with minimal examination of possible permutations. All clusters can be determined, thus reducing a large set of aircraft into several independent groups, significantly reducing the number of permutations needed to be examined for validity. Clustering is a highly effective process in reducing tested permutations as the number of permutations for each individual cluster is orders of magnitudes times smaller than the number of permutations for the entire set. All valid permutations are still enumerated but determining a global perfect matching requires significantly less computations.

2.4 Permutation Enumeration

The primary task of any global ALP solving algorithm is the enumeration of all valid sequences of aircraft. A valid sequence is a sequence of aircraft that contain only one of each aircraft and every aircraft in an arrival fleet. For example, the set of aircraft $A = \{1,2,3\}$, the valid sequences will contain one of each aircraft and assign a landing position to each aircraft, e.g. $\{1 \rightarrow 2 \rightarrow 3\}$. A valid sequence must only assign positions to aircraft with the capacity to land in that position and adhere to the precedence relationships where appropriate. An example process of inputting a fleet or cluster and outputting a set list of valid sequences and the methodology involved is shown in Figure 5.

Position	**1**	**2**	**3**	**4**	**5**
Allowable Aircraft Assignment	1	1	2	2	5
	2	2	3	3	
		3	4	4	

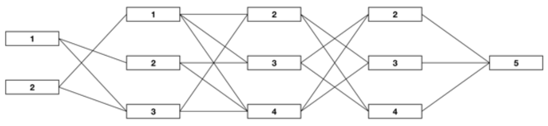


Fig. 5. Fleet flow chart for positions 1 to 5.

There are no smaller clusters inside Figure 5 and the fleet has been completely pruned. There is a total of 54 possible permutations to be tested for validity. Precedence constraints will also be considered where appropriate to satisfy all validity requirements. However, to simplify the example no precedence relationships will be considered here. The bi-adjacency matrix B is created and the permanent of that matrix determined so that final number of perfect matchings is known. The bi-adjacency matrix is shown in Figure 6.

$$B = \left\{\begin{matrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}\right\}, perm(B) = 6$$

Fig. 6. Bi-adjacency matrix and its permanent.

The permanent value 6 indicates the number of perfect matchings, thus the number of valid sequences for the example data. This provides a stop point when enumerating sequences. The algorithm presented in this paper operates by assigning a vacant landing position to an aircraft and looping for all aircraft in a fleet. In the event of no vacant positions being available given an aircraft's allowable positioning assignments, the latest position that hasn't reached the final aircraft in its independent set of aircraft assignments is updated and all trailing positions reset to their first aircraft assignment. If all aircraft are assigned a position, the sequence is valid and the latest position that hasn't reached the final aircraft in its independent set of aircraft assignments is updated to be assigned to the next aircraft in the set. The trailing positions are all reset to their first aircraft assignment and this process is repeated until the number of valid sequences enumerated is equal to the permanent of B determined prior to the enumeration process.

2.5 Position Assignment

The logic flowchart described previously to enumerate all sequences can be visualised as a dynamic assignment that changes as each valid and invalid sequence is tested. This can be understood by creating a sequence and altering the assigned aircraft of the latest position until the algorithm assigns a position to all aircrafts. Using a table that shows the active assignments and a comparison table showing how many aircrafts are available for assignment in a given position allows for quick manipulation of assigned aircraft. To update the active assignment, the latest position is compared to the total number of aircraft that can be assigned to position n and the active aircraft assignment is increased by 1 and all trailing active aircraft assignments are set to 1. The resulting sequence is not valid, so the latest position is updated by increasing the active aircraft assignment by 1 and all trailing positions reset their active assignment to the first aircraft in their independent set of allowable aircraft assignments. The resulting sequence is still not valid, so the active assignment increment process is repeated until the number of valid sequences found is equal to the permanent found prior to the enumeration process.

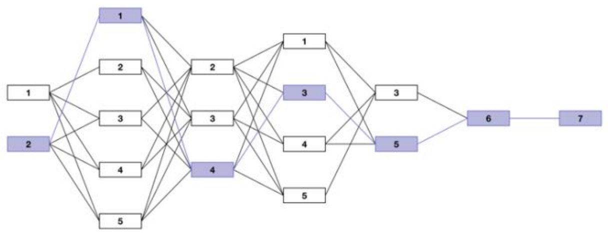


Fig. 7. Fleet flow chart and valid sequence (shaded).

The efficiency of the permutation enumeration is determined by the number of sequences examined, so reducing the amount of unnecessary active aircraft assignments will decrease the required computation time. To limit unnecessary assignment, as the algorithm assigns an aircraft to positions $1 \rightarrow n$, if there are no available assignable aircraft to position j, the latest position before position j that is not at its final aircraft assignment will be updated. Thus, skipping a significant number of unnecessary assignments and sequence tests since no valid permutation will exist past position j. There are several optimisation techniques to reduce the number of computational comparisons. These processes involved optimised function calls and minimising recursive calls of nested functions. When multiple clusters are present, the valid sequences of each cluster are determined and the permutations of each sequence when concatenated together generate the full list of valid sequences. The sequences that violate precedence relationships can be eliminated from the valid set list and all valid sequences are analysed individually for optimal path finding.
For increasing number of aircraft, the probability of clustering increases and the computation time scales almost linearly. However, predictions of future allowable positions become less accurate. An adaptive procedure that generates an optimal sequence quickly from new information is paramount. Figure 7 is an example of a cluster present in a much larger set, however, the sequence illustrated will be valid regardless of the trailing aircraft to final node since there will be numerous permutations all beginning with the sequence above.

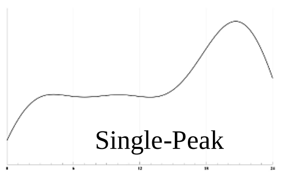
3 Simulations

The model simulation uses numerous randomly generated arrival fleets and determines statistics for each simulation. The randomly generated data is based off probability density functions and multimodal distributions to represent expected air traffic and aircraft variety. Each aircraft is given a time of arrival with respect to expected order of arrival. The aircraft are assigned a constrained earliest and latest time of arrival, weight class, precedence jet route relationship, and a randomly determined compliance factor. This information is converted into an unpruned set of allowable position assignments and the fleet is pruned to improve optimisation speed. The fleet is parsed into the optimal sequence enumerator and statistics for each simulation are recorded to determine averages for each set.

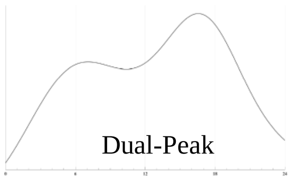
Figure 8 shows the density distributions and the associated statistical data. The distributions are used to indicate the dense and sparse periods during a 24-hour period. Simulation blocks with standard and high compliance used a compliance factor 0.65 and 0.90 respectively. Each block consists of the same number of individual simulations, jet routes, and weight class distribution variables except for the low-density block, which featured low and standard weight class distributions.

Each distribution was separated into relevant blocks consisting of 1000 individual simulations each and 4 jet routes uniformly distributed. The compliance factors and weight class distributions were selected to demonstrate the effectiveness of the model with varying arrival conditions.

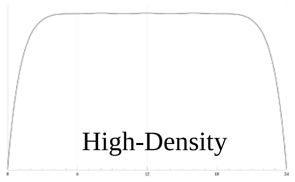
From Table 2, the greatest influence of RET reduction comes from the number of aircraft, this is to be expected as more aircraft landing in a single period will increase the density and often the possible number of successful permutations. A higher compliance also results in further reduction.



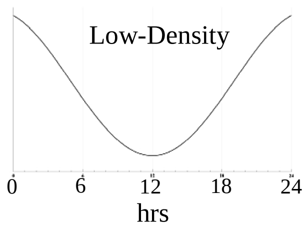
Input Variable	Value	
Number of Simulations	1000	
Number of Jet Routes	4	
Weight Distribution	$\begin{cases}\mu = 3\\ \sigma = 1\end{cases}$	
Multimodal Distribution	$\begin{cases}\mu = [2,10,19,22]\\ \sigma^2 = [3.9,4.2,5,4.1]\end{cases}$	
Compliancy Factor	0.65	0.90



Input Variable	Value
Number of Simulations	1000
Number of Jet Routes	4
Weight Distribution	$\begin{cases}\mu = 3\\ \sigma = 1\end{cases}$
Multimodal Distribution	$\begin{cases}\mu = [5,10,15,17,22]\\ \sigma^2 = [3.6,5,7,3,7]\end{cases}$
Compliancy Factor	0.65



Input Variable	Value
Number of Simulations	1000
Number of Jet Routes	4
Weight Distribution	$\begin{cases}\mu = 3\\ \sigma = 1\end{cases}$
Multimodal Distribution	$\begin{cases}\mu = [-4,0,4,8,12,16,20,24,28]\\ \sigma^2 = [3,3,3,3,3,3,3,3,3]\end{cases}$
Compliancy Factor	0.65



Input Variable	Value	
Number of Simulations	1000	
Number of Jet Routes	4	
Weight Distribution	$\begin{cases}\mu = 3\\ \sigma = 1\end{cases}$	
Multimodal Distribution	$\begin{cases}\mu = [-5,12,29]\\ \sigma^2 = [12,20,12]\end{cases}$	
Compliancy Factor	0.65	0.90

Fig. 8. Distribution densities for a 24-hour period with probability of an aircraft landing at a given time.

Table 2. Simulation output.

Simulation Block	#Aircraft	#Clusters	RET Reduction (sec)	Comp. Time (sec)
Single-peak standard compliance	283.668	31.971	422.280	0.331
Single-peak high compliance	283.965	35.616	788.880	3.044
Dual-peak	315.046	34.332	599.880	4.412
High-density	408.006	44.500	986.640	3.815
Low-density standard compliance	171.905	15.048	131.280	0.109
Low-density high compliance	171.884	21.860	245.040	0.124
Low-density low weight class and high compliance	172.093	21.498	260.520	0.134

This is expected as the number of successful permutations would exponentially increase with both variables. The computational time for all data sets is on average only a few seconds, showing that a real-time response rate can be achieved through this model.

An adaptive sequencer relies on a generic model that, when given new information, can readily update the optimal solution. The number of clusters does not correlate to the computational time required, therefore utilising the clustering effect to dynamically determine optimal orders will minimise the effect of unfavourable events occurring. The primary detriment in using this simulation data is the assumed distributions. However, the model incorporates distribution variables, so with empirical data, accurate simulations can be readily executed to validate the success of the model.

Table 3. Example of arrival aircraft fleet.

ID	Weight Class	Arrival Time	Earliest Arrival Time	Latest Arrival Time	Jet-route	Compliant
1	Light	00:00	23:54	00:06	4	Y
2	Heavy	00:07	00:04	00:20	1	Y
3	Light	00:20	00:14	00:28	4	Y
4	Medium	00:22	00:09	00:36	1	Y
5	Light	00:27	00:18	00:30	4	Y
6	Super	00:36	00:22	00:39	4	Y
7	Super	00:38	00:23	00:49	3	Y
8	Heavy	00:45	00:30	00:57	2	Y

Table 3 shows typical input data for an arriving fleet of aircraft. For simplicity, this example assumes complete compliance and no further variables alter during analysis. This sequence produces a RET of 17 minutes, after optimisation the time reduces to 15 minutes.

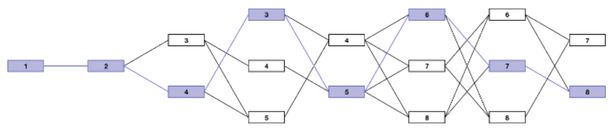


Fig. 5. Flowchart and optimal path nodes (shaded) nodes in order $\{1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8\}$.

A flowchart of the optimal path and the arrival fleet is Figure 9. Assuming this fleet is an isolated cluster the model has the capacity to optimise this cluster dynamically as information changes and allows for near instantaneous computation and output to ATC.

4 Conclusions

This paper presents an approach for optimising aircraft landing sequencing by exploiting the resulting clustering effect created with fixed aircraft landing positions. The model isolates clusters by identifying time periods where there is no interaction of arrival behaviour between clusters. These clusters are examined and parsed into an optimal sequence enumerator that determines valid sequences and the total RET of each sequence. The clusters and the intervals between the successive clusters is analysed to determine the optimal sequence. As the number of clusters increases this methodology allows for a mostly linear scaling of computational time, with most of it taken up by the permutation enumeration process and the minimum RET determination algorithm.

The model was validated by performing simulations using assumed distributions of weight class and arrival density. The compliance factor determines the percentage of aircraft that are willing to cooperate with ATC. To demonstrate the effects of different distributions and compliance factors, a variety of inputs were selected for further analysis. The daily arrival frequencies selected for analysis included single-peak, dual-peak, and low and high-density distributions. Moderate and high compliance factors of 0.65 and 0.90 were selected to demonstrate the effects of increased aircraft compliance. The model simulated a 24-hour period of arriving aircraft and recorded the relevant statistics to measure the effectiveness of the optimisation process. These output statistics included number of aircraft, number of clusters, the RET reduction, and the computational time required to complete the optimisation.

The greatest influence on RET reduction are the compliance factor and density of aircraft arrivals. This was expected as the number of successful permutations exponentially increases with these two variables. This effect can be observed when comparing the low-density and high-density data sets as well as the standard and high compliance data sets. The results indicate that, with 65% of arriving aircraft willing to adjust their landing position within their estimated time of arrival range, the RET can be reduced by several minutes over a 24-hour period. The results also indicate the number of clusters will vary more significantly due to the number of aircraft. For low density data sets, as the compliance factor increases the number of clusters will increase regardless of the number of aircraft due to the method used by the cluster finding algorithm. This effect is not present for higher density data sets as observed in the differing compliance factor single-peak data sets. All simulation data sets show on average a computational time requirement less than 5 seconds. These results indicate that it is possible to optimise large fleets of arriving aircraft whilst ensuring real time response rates. The computational time still varies greatly for dense and complex clusters due to the magnitude of resulting successful permutations. These dense clusters formed when the random number generation produced long periods of consecutive positive compliance and large time of arrival windows, however these results are not likely to occur in a real environment and are considered outliers.

Due to the frequency of aircraft non-compliance, the proposed model takes advantage of this challenge to optimal sequencing by producing consistent real-time response results. In the worst case, the model will output the default order, which is a typical result of other strategies in real environments due to low aircraft compliance. The strength of this model derives from parsing any generic fleet into the presented algorithm and producing optimised results in real time. This requires no pre-processed network analysis or static cases of assumed allowed repositions. The model, with further development, will need to account for any typical airport with variable runways, and runway configurations, and incorporate dynamic arrival time windows to restrict allowable landing assignments more accurately. The data has been constrained to generate random parameters that emulate aircraft non-compliance and unfavourable conditions, thus, producing a resulting clustering effect. The computational time has been reduced by minimising recursive function calls and fast permutation elimination.

The results indicate that clustering is a common occurrence for large aircraft sets and the problem simplifies to optimisation of local clusters. This significantly reduces effort required to enumerate all permutations. The computational time required to optimise each cluster scales linearly with the number of clusters and the individual cluster time complexity exponentially increases with the number of successful permutations. Due to this exponential permutation growth, any further optimisation of the computational model would greatly accelerate the process and these dense clusters can be optimised in real time.

References

[1] Soomer, M & Franx, GJ 2008, 'Scheduling aircraft landings using airlines' preferences', European Journal of Operational Research, vol. 190, no. 1, pp. 277-91.

[2] Inniss, TR & Ball, MO 2004, 'Estimating one-parameter airport arrival capacity distributions for air traffic flow management', Air Traffic Control Quarterly, vol. 12, no. 3, pp. 223-51.

[3] Chandran, B & Balakrishnan, H 2007, 'A dynamic programming algorithm for robust runway scheduling', paper presented to American Control Conference, 2007. ACC'07.

[4] Dear, RG 1976, The dynamic scheduling of aircraft in the near terminal area, Cambridge, Mass.: Flight Transportation Laboratory, Massachusetts Institute of Technology.

[5] Balakrishnan, H & Chandran, B 2006, 'Scheduling aircraft landings under constrained position shifting', paper presented to AIAA guidance, navigation, and control conference and exhibit.

[6] Venkatakrishnan, C, Barnett, A & Odoni, AR 1993, 'Landings at Logan Airport: Describing and increasing airport capacity', Transportation Science, vol. 27, no. 3, pp. 211-27.

[7] Wambsganss, M 1996, 'Collaborative decision making through dynamic information transfer', Air Traffic Control Quarterly, vol. 4, no. 2, pp. 109-25.

[8] Gilbo, EP 1997, 'Optimizing airport capacity utilization in air traffic flow management subject to constraints at arrival and departure fixes', IEEE Transactions on Control Systems Technology, vol. 5, no. 5, pp. 490-503.

[9] Gilbo, E & Howard, KW 2000, 'Collaborative optimization of airport arrival and departure traffic flow management strategies for CDM', paper presented to 3rd USA/Europe Air Traffic Management R&D Seminar.

[10] Le, L, Kholfi, S, Donohue, G & Chen, C 2003, 'Proposal for demand management using auction-based arrival slot allocation', paper presented to Proceedings of the 5th USA/Europe Air Traffic Management R&D Seminar, Budapest, Hungary.

[11] Terrab, M & Odoni, AR 1993, 'Strategic flow management for air traffic control', Operations research, vol. 41, no. 1, pp. 138-52.

[12] Neuman, F & Erzberger, H 1991, 'Analysis of delay reducing and fuel saving sequencing and spacing algorithms for arrival traffic'.

[13] Uno, T 2001, 'A fast algorithm for enumerating bipartite perfect matchings', paper presented to International Symposium on Algorithms and Computation.

[14] Andreatta, G, Brunetta, L & Guastalla, G 1997, 'Multi-airport ground holding problem: a heuristic approach based on priority rules', in Modelling and Simulation in Air Traffic Management, Springer, pp. 71-89.

[15] Beasley, J, Krishnamoorthy, M, Sharaiha, Y & Abramson, D 2004, 'Displacement problem and dynamically scheduling aircraft landings', Journal of the operational research society, vol. 55, no. 1, pp. 54-64.

[16] Bianco, L, Dell'Olmo, P & Giordani, S 1997, 'Scheduling models and algorithms for TMA traffic management', in Modelling and simulation in air traffic management, Springer, pp. 139-67.

[17] Faye, A 2015, 'Solving the aircraft landing problem with time discretization approach', European Journal of Operational Research, vol. 242, no. 3, pp. 1028-38.

[18] Filar, JA, Manyem, P & White, K 2001, 'How airlines and airports recover from schedule perturbations: a survey', Annals of operations research, vol. 108, no. 1-4, pp. 315-33.

[19] Rempala, GA & Wesolowski, J 2008, 'Permanent Designs and Related Topics', Symmetric Functionals on Random Matrices and Random Matchings Problems, pp. 121-48.

Contact Author Email Address

Mailto: bil@rmit.edu.au

Copyright Statement