A meta-heuristic approach to aircraft departure scheduling at London Heathrow airport.

Jason A. D. Atkin[1], Edmund K. Burke[1], John S. Greenwood[2], and Dale Reeson[3]

[1] School of Computer Science and Information Technology, University Of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 2BB
{jaa,ekb}@cs.nott.ac.uk

[2] National Air Traffic Services Ltd, NATS CTC, 4000 Parkway, Whiteley, Fareham, Hampshire, PO15 7FL, England

[3] National Air Traffic Services Ltd, Heathrow Airport, Hounslow, Middlesex, TW6 1JJ

Abstract: London Heathrow airport is one of the busiest airports in the world. Moreover, it is unusual among the world's leading airports in that it only has two runways. At many airports the runway throughput is the bottleneck to the departure process and, as such, it is vital to schedule departures effectively and efficiently. For reasons of safety, separations need to be enforced between departing aircraft. The minimum separation between any pair of departing aircraft is determined not only by those aircraft but also by the flight paths and speeds of aircraft that have previously departed. Departures from London Heathrow are subject to physical constraints that are not usually modelled in departure runway scheduling models. There are many constraints which impact upon the orders of aircraft that are possible and we will show how these constraints either have already been included in the model we present or can be included in future. The runway controllers are responsible for the sequencing of the aircraft for the departure runway. This is currently carried out manually. In this paper we propose a meta-heuristic-based solution for determining good sequences of aircraft in order to aid the runway controller in this difficult and demanding task. Finally some results are given to show the effectiveness of this system and we evaluate those results against manually produced real world schedules.

1 Introduction

London Heathrow is a busy two-runway airport which, due to its popularity with both airlines and passengers, suffers severe aircraft congestion at certain times. Traffic in airports is not evenly spread, for obvious reasons which pertain to airline and passenger preferences. There are inevitably times when the departures process is congested but the arrivals are sparse, vice versa, and times when both are congested. London Heathrow airport is actually situated on an extremely small plot of land in comparison both to how busy the airport is and to other airports around the world.

The airport capacity problem is concerned with estimating the capacity of an airport in terms of arrivals and departures. It has been examined for a number of years. Newell [14] provided a model and showed that the capacity of the airport is increased when arrivals and departures can be alternated on both runways. Although mixed mode, where arrivals and departures are intermixed on a runway, is preferable for increasing the throughput, this is not currently possible at Heathrow due to the proximity of the surrounding residences, although it may begin to be considered for peak times.

The departure flow at Logan airport was analysed in [11] and [12] and Logan airport was compared to other major airports. Runway scheduling was seen to be a bottleneck upon the departure process and the authors concluded that it is vital to increase the throughput of the departure runway.

There are some similarities between the arrival and departure processes for the runways at an airport. Both processes are subject to sequence-dependent separation times between aircraft. Previous research has looked at the arrivals problem with the goal being to order arriving aircraft for a single runway so as to either minimise the total completion time or to minimise the total deviation from an ideal arrival time for each aircraft. Mixed integer zero-one formulations were presented in [6] and Genetic Algorithms were shown to be effective in [7].

Abela et al [1] looked at the arrivals problem for a set of aircraft with landing time windows. They presented a genetic algorithm to give an approximate solution and branch and bound algorithm for solving the problem when formulated as a 0-1 mixed integer programming problem to give an exact solution.

A heuristic approach for an upper bound and a branch and bound algorithm for the arrivals problem were given in [10]. A network simplex method was used to assign arrival times given any partial ordering of aircraft.

The arrivals problem, as it is presented in the literature, however, does not address the major constraints upon the departures problem at London Heathrow airport.

A constraint satisfaction based model for the departure problem was presented in [13] for solution by ILOG Solver and Scheduler. A fifteen minute time slot was assigned to each aircraft and separations were assigned based upon the size and speed of the aircraft and upon the exit point that the departing aircraft were going to use.

The departure process was analysed and a departure planner proposed by Anagnostakis et. al. in [3], [4] and [5]. A search tree was described and branch and bound techniques or an A* algorithm were recommended for solving the departure problem in [2]. A dynamic program was suggested in [15] to solve the departure order problem by limiting the possible number of aircraft that are considered for any place in the schedule, reducing the search space dramatically.

If only considering separations between adjacent aircraft and ignoring the physical constraints from the holding points, the departure problem can be seen to be a variant of the single machine job sequencing problem where jobs have sequence-dependent processing or set-up times. Substantial research has been undertaken into this problem. For example Bianco et. al. [8] looked at the generalised problem with release dates as well as sequence-dependent processing times, showing the equivalency to the cumulative asymmetric travelling salesman problem with release dates. To ensure safety in the departure process, however, it is not possible to only consider adjacent pairs of aircraft and it is easy to produce schedules where all adjacent pairs have the required separations but other aircraft pairs do not.

Craig et. al. [9] did look at the effects of one holding point structure and gave a dynamic programming solution for scheduling take-offs. In practice, however, the holding point structures are more flexible than the one described here and a more general solution needs to be developed.

There are important constraints at London Heathrow airport that are not normally considered in the departure problem as it is presented in the current scientific literature. These are identified in the problem description below.

2 Problem description

The objective of this paper is to increase the throughput of the departure runway subject to various constraints, with safety being paramount.

There are currently only two runways in normal use at Heathrow, however, if environmental targets are met, there may be a possibility to add a third, parallel runway in the future. At any time of the day only one runway can currently be used for departures.

The direction of the wind determines the direction in which the runways are used. The runways are labelled according to the direction in which they are employed and whether they are on the right or the left when facing that direction. The four runway configurations have been labelled in Fig. 1. For example, when arriving or departing heading west, the northern runway is referred to as 27R as it has a direction of 270 degrees and is the runway on the right.

There is actually a third runway already but this can only ever be used for arrivals. It is shorter than the other two and not long enough for many Heathrow departures. It is used no more than twice per year. It also intersects both of the other runways so it is not practical to use it if either of the other two runways is in use. Indeed, it is usually used as a taxiway.

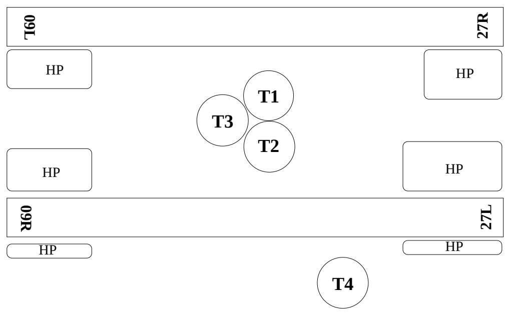

Fig. 1. The layout of London Heathrow Airport

There are currently four terminals at London Heathrow, labelled T1 to 4 in Fig. 1. Three terminals are situated between the runways but the fourth is to the south of the southern runway.

When a flight is ready to depart a delivery controller has to give permission for engine start up. A ground controller then instructs the pilot in order to control the movement of the aircraft around the taxiways. Once an aircraft approaches the runway end and is no longer in conflict with any other aircraft the ground controller will relinquish control of the aircraft to the runway controller.

In this paper, we are concerned only with the operations of the runway controller. We assume that the ground controller and delivery controller are currently outside of the system and merely feed aircraft into the start of the system. Later research will look to include these roles into the model.

There are holding points, labelled HP in Fig. 1 at each end of each of the runways, and both north and south of the southern runway. Within these physical holding point structures the runway controller can reorder the aircraft before they reach the runway.

2.1 Holding point constraints

Aircraft go through holding points to get to the runways. Holding points can be considered to be one or more entrance queues to some maneuvering space then finally to a single take-off order on the runway. Where there are different entrance queues available, the ground controller will usually send an aircraft into the most convenient queue. The runway controller can request aircraft to be sent to specific queues but in practice, as the runway controller is very busy with the aircraft already in the holding points, there is rarely sufficient time to also consider the aircraft the ground controller has.

As mentioned before, Heathrow has very limited space so the holding point and taxi space is limited. Given the initial order of aircraft in the input queues to the holding points the runway controller has to decide how to sequence the take-offs in order to maximise the throughput at the runway. This can be a very difficult task at times.

Only limited amounts of reordering are possible at these holding points. The configuration of the holding points varies greatly between runway ends and will determine what reordering operations can take place and the costs involved in each operation.

2.2 Minimum separations

To ensure safety, minimum separation times are imposed between aircraft taking off. The order of the aircraft for take-off can make a significant difference to the total delay that needs to be imposed upon the aircraft.

The minimum separation between aircraft is determined by:

- Wake Vortex: Large aircraft leave a stronger wake vortex than smaller/lighter aircraft and are also less affected by wake vortex. Every aircraft has a weight category and the wake vortex separation for any pair of aircraft can be determined by comparing their weight categories.
- Departure Routes: Aircraft will usually have a Standard Instrument Departure (SID) route assigned to them, giving a pilot a known departure route to follow. The relative SID routes of any two aircraft will impose a minimum departure interval between them. This ensures that safe minimum separation distances are kept while in flight. At times of congestion in the airspace a larger than normal separation may be required between certain SID routes, in order to increase the separation between flights heading into the congestion. These separations differ depending upon the runway in use at the time.
- Speed Group: The relative flight speeds of the aircraft can also make a difference to the separations which must be imposed upon aircraft flying the same or similar routes. The relative speed groups of the two aircraft modify the separation required for the relative SID routes. If the following aircraft will close the distance then a larger initial separation is necessary. Conversely, if the following aircraft is slower then a lower separation can sometimes be applied.

The runway controller will aim for minimum separations between aircraft wherever possible. It should be noted here that a controller has some discretion as far as some separations are concerned. In particular some of the SID route based separations can be reduced in good visibility.

2.3 Other constraints

The departure process is a dynamic system where aircraft are added to, and removed from, the system over time. The runway controller will have only limited knowledge about the aircraft that are not currently at the holding points.

The runway controller has a lot of information that is very hard to capture as hard data. In many cases a controller will be weighing the effects of contradictory constraints such as maximising throughput while minimising overtaking, to ensure fairness and minimising maneuvering, to reduce workload.

2.4 Overall objective

The objective is to find candidate solutions for which the runway throughput is maximised and all constraints are met. We were told by one air traffic controller that the best figure obtained for Heathrow was 54 aircraft in an hour and that this figure is so good that it is extremely unusual.

For our research we use a reduction in the holding point delay as a surrogate objective. Holding point delay is measured as the amount of time the aircraft spend in the holding point. Any objective to minimise this will have the effect of reducing the number of large separations and also of moving larger separations later in the take off order, so that they delay less aircraft. Moving larger separations later means that there is more opportunity to deal with them using new aircraft entering the system later, so a delay based objective for the problem at any instant in time is a good surrogate for a throughput based approach for the overall schedule. As the holding point arrival times are constant, the sum of take-off times could be used as an equivalent, but less meaningful, objective function.

3 Model description

In this model we aim to maximise the throughput of the runway by minimising the total delay, D, suffered by the aircraft at the holding points. Let h_i be the arrival time for aircraft i at the holding point, where i is an integer ≥ 1. The integer i represents the position of the aircraft in the take-off order. If d_i is the take-off time for aircraft i from the runway, then we can calculate the total delay at the holding points using equation 1 where n is the total number of aircraft departing.

We define a function $S(j,i)$ to give the minimum separation necessary between leading aircraft j and (not necessarily immediately) following aircraft i to meet all separation requirements. Function $S(j,i)$ incorporates all separation rules for weight classes, SID routes and speed groups.

If we assign each aircraft a route through the holding point structure then, given a holding point entry time, h_i, and a suitable function, $T(t_i)$, for the traversal time through the holding points along a traversal path t_i for aircraft i, the earliest time the aircraft can reach the runway can be calculated as $h_i + T(t_i)$.

For the model, we assume that all aircraft take off as early as possible, so for any aircraft, i, the take-off time, d_i, can be predicted as the earliest point that both allows sufficient time to reach the runway and complies with all required separation rules, equation 2.

Function $S(j,i)$ can be taken to be the maximum of two functions: $W(w_j,w_i)$ which will calculate the required wake vortex separation from the weight categories w_i and w_j of aircraft i and j and $R(r_j,s_j,r_i,s_i)$ which will calculate the required separation based upon the SID routes, r_i and r_j, and the speed groups, s_i and s_j, of the aircraft i and j (see equation 3). The separations for SID routes differ depending on which runway the aircraft are departing from, so $R(r_j,s_j,r_i,s_i)$, like $T(t_i)$, is runway specific.

Both functions $W(w_j,w_i)$ and $R(r_j,s_j,r_i,s_i)$ are defined to return standard separation values in accordance with current regulations. It should be noted that the runway controller has some flexibility in good weather to reduce the separations given by $R(r_j,s_j,r_i,s_i)$ and a fully operational decision support system would allow the controller to do just that.

3.1 Formal description of the mathematical model

We can express this model as follows:

Minimize

$$D = \sum_{i=1}^{n}(d_i - h_i) \tag{1}$$

Where

$$d_i = \max(h_i + T(t_i), \max_{j=1..(i-1)}(d_j + S(j,i))) \tag{2}$$

$$S(j,i) = \max(W(w_j,w_i), R(r_j,s_j,r_i,s_i)) \tag{3}$$

3.2 Holding point constraints

Any practical model must incorporate the holding point constraints. There is no point in presenting candidate solutions to a runway controller if he/she cannot actually achieve the order due to the physical constraints.

An example of a holding point structure can be seen in Fig. 2. The nodes are the valid positions for aircraft and the arcs show moves that aircraft could make. This network is more restrictive than the actual network at the associated holding point at Heathrow and is deliberately so. Any solution which is feasible for this network should be both feasible and sensible for the real network.

We investigate meta-heuristic local search, as specified in section 4. This means that the search will move from one solution to the next. A solution could consist of just a final take-off order or it could give details about all of the taxi movement within the holding points and a take-off order could be derived from this.

If a solution consisted of the order in which individual moves were made within the holding point, specifying details of how aircraft attain the reordering as well as the final take-off order achieved, the search space would be extremely large. Many solutions would give the same take-off order but differ in the paths used to traverse the holding point or in the order in which moves were made. The relative order in which many actions take place often does not matter. So, many apparently different solutions may, in fact, be identical. Some paths take longer to traverse than others, so some solutions will be much better than others that have the same take-off order. This manoeuvring cost would have to be considered within the objective function.

Rather than modelling the movement within the holding points, the selected model instead has solutions which specify only a take-off order rather than how the order is achieved. Not all potential take-off orders will be achievable however, so this must be verified. The method, in which the reordering is attained, does have an impact and some ways are obviously better than others. We use a heuristic to assign holding point traversal paths to aircraft, then perform a feasibility check to verify that the solution is achievable, given the holding point structure.

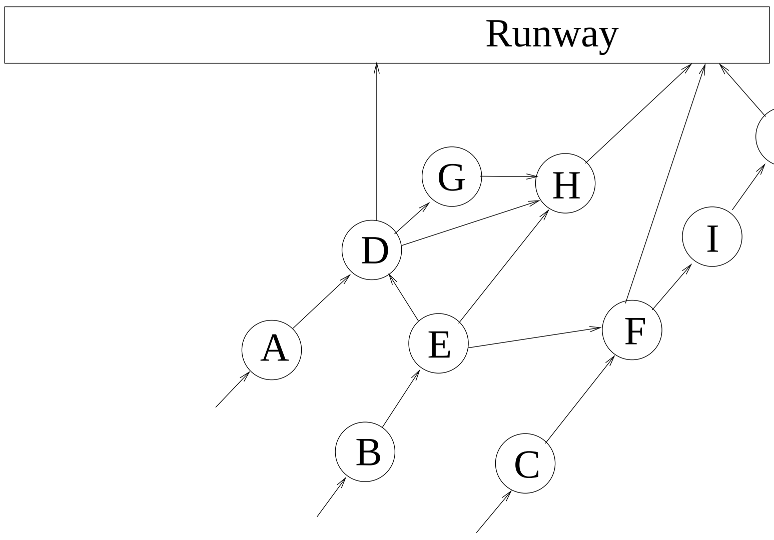

Fig. 2. An example holding point network structure.

3.3 Path assignment heuristic

The heuristic to assign paths through the holding point to aircraft is holding point specific. The first stage in the design is to identify the good paths through the holding point. This is performed by asking the runway controllers about the ease and feasibility of using possible paths and eliminating from consideration any which are difficult to use, leaving only good paths. Given each entrance point, multiple paths are available.

Some paths are faster than others, but all paths are easy to use even though some will be longer than others. The allocation heuristic allocates slower paths to aircraft that are overtaken and faster paths to aircraft that overtake. This ensures that all aircraft on longer, slower paths are being overtaken in the holding point and therefore have much more time available to traverse the holding point.

For example, if two aircraft arriving at entrance A in Fig. 2 needed to reverse their order before take-off, the first would be assigned path ADGH and the second path ADH. The first would then hold at G while the second overtook it.

Once an aircraft is in the holding point the heuristic does not allow the assigned path to be changed so it is important to attempt to maintain flexibility when assigning paths to aircraft close to the holding point.

3.4 Directed graph model of the holding point

Once paths have been assigned to aircraft, the feasibility of the schedule is checked by feeding aircraft into the start nodes of the directed graph for the holding point, in the order they will arrive at the holding point. Fig. 2 shows the graph used for the 27R holding point. Rules are used to determine which aircraft to move next and whether moving a specific aircraft could block another aircraft. If the aircraft can exit the graph onto the runway in the desired take-off order then the schedule is deemed feasible.

Two levels of pre-processing are used. The first is based purely upon the holding point structure and the possible paths that could be used. This stage is performed for each holding point graph prior to the start of the tests and can be performed off-line. It caches information about the later structure of the holding point beyond each node, recording for each of the paths entering the node, details of which other paths converge with it and how many nodes are not shared between them. The second pre-processing stage requires knowledge of the desired take-off order so is performed before each feasibility check. This stage calculates partial take off orders at each node, for sets of converging paths, ensuring that, for any pair of aircraft for which there is no possibility of changing order beyond this node, the aircraft enter the node in the correct order. Together, the pre-processing results provide knowledge about whether any aircraft can move without blocking another aircraft, ensuring that the feasibility check can be made both deterministically and quickly.

4 Departure scheduling algorithms

All of the search heuristics that we investigated had the same basic format but differed in the details. They are described below.

First descent

The first descent algorithm is the most simplistic algorithm and has the following structure.

1. Obtain initial current solution. An initial current solution will usually be a solution where the aircraft are in the order at which they arrived at the holding points. This solution has the advantage that it will always be feasible as no reordering is necessary within the holding points.
2. Evaluate the solution as described in section 4.2, using the default holding point paths as no reordering is necessary so feasibility is guaranteed.
3. Generate a new candidate solution by selecting a solution from the neighbourhood of the current solution, as described in section 4.1.
4. Heuristically assign holding point paths to aircraft, as in section 3.3.
5. Check the feasibility at the holding point structure to ensure that the order of take-off is possible, as described in section 3.4.
6. Evaluate the cost of the solution, as shown in section 4.2.
7. If the candidate solution has a lower cost than the current solution then accept it as the new current solution.
8. If the given number of evaluations have been completed then stop the algorithm and report the best result so far, otherwise return to step 3.

Simulated annealing

The simulated annealing algorithm has the same structure as the first descent algorithm except in step 7. In step 7, rather than only accepting better solutions, the simulated annealing algorithm will sometimes accept moves to worse solutions, allowing it to escape local optima. If the cost of the new solution is less than the cost of the current solution then the new solution will always be accepted. If the cost of the new solution is more than the cost of the current solution then there is a small chance to still accept the new solution.

Let D_{curr} be the cost of the current solution and D_{cand} be the cost of the candidate solution.

The candidate solution will be accepted in step 7 if:

$$D_{cand} < D_{curr} \tag{4}$$

or

$$R < e^{-\delta/T} \tag{5}$$

where $\delta = D_{cand} - D_{curr}$ is the difference between the current and candidate solutions, R represents a uniform random variable in the range [0..1] and T is a temperature which is initially large, so that many bad solutions are accepted, but decreases over time so that the simulated annealing algorithm slowly converges on the first descent over time.

Steeper descent

The steeper descent and tabu search algorithms are similar to the first descent algorithm but both generate fifty candidate solutions at a time in step 3 rather than just one. All of the fifty candidates are evaluated simultaneously in steps 4, 5 and 6. In step 7 the best of the feasible candidate solutions is adopted as the new current solution in step 7. The best candidate is adopted even if it is worse than the current solution, which means this is more than a strict descent algorithm. This gives the algorithm a limited ability to move out of local optima but no method to avoid it moving straight back to the local optimum it just left.

Evaluations of candidates are expensive so, for comparison, the searches are limited to a number of evaluations rather than a number of iterations. This means that the first descent and simulated annealing algorithms run for fifty times as many iterations as the steeper descent and tabu search algorithms.

Tabu search

The tabu search algorithm is similar to the steeper descent algorithm except that it maintains a list of tabu moves. When a move is made, details of the move are stored on a tabu list. The tabu list stores details of which aircraft were moved and the absolute positions they were moved from, for the last 10 moves made. If a future move attempts to place all of these aircraft back at the position from which they were moved then it will be declared tabu and rejected.

Like the steeper descent algorithm, the tabu search evaluates fifty candidate solutions at once. The only difference between the two algorithms is that, in step 7, each candidate is evaluated and tested to see if it matches a move on the tabu list. The best of the feasible, non-tabu candidates is adopted and the details of the move made are stored on the tabu list. Again, the best candidate is adopted even if it is worse than the current solution, allowing the search to escape loal optima. The tabu list ensures the search cannot quickly return to a local optimum it has escaped.

4.1 Neighbourhood design

These algorithms all rely upon the selection of neighbouring solutions. Choosing a neighbouring solution is a matter of first randomly determining the move to use then randomly determining the details of that move. A large number of moves are available to the searches.

Swap single aircraft

The *swap single aircraft* move takes two aircraft from the schedule and swaps the positions of the aircraft in the final take-off order. There is a 30% chance that this move will be used, selecting two aircraft at random.

Shift aircraft

The *shift multiple aircraft* move selects a consecutive group of one to five aircraft and moves them to a new random position in the schedule, either forwards or backwards. There is a 50% chance that this move will be made. Moving multiple aircraft is especially useful once the aircraft are in north/south alternating pattern as moving a single aircraft would usually make the schedule worse in that case.

Randomise a set of aircraft

The *randomise a set of aircraft* move selects a consecutive set of aircraft as the target. Each aircraft within this set is then moved to a random position in the set. This move may emulate a shift, swap or a reversal in the order in some cases but some of the schedules attainable through this move are not attainable otherwise. There is a 20% chance that this move will be used. In experimental results this move has shown a valuable contribution in finding good schedules, when not overused.

4.2 Objective function

It is advisable to limit the amount of deviation from the holding point arrival order as well as to limit the delay. Reducing the number of 'swaps' of aircraft in the take-off order will aid in reducing workload for the pilots and controllers and it will also make it easier for the next iteration to build a feasible schedule.

With this goal in mind, the following objective function is used by the search algorithms:

$$D = \alpha \sum_{i=1}^{n} (A_i - i)^2 + \beta \sum_{i=1}^{n} (d_i - h_i) \tag{6}$$

Where n is the number of aircraft in the take-off schedule, d_i is the take-off time and h_i is the holding point arrival time of the ith aircraft in the take-off queue. A_i is the position, $1, 2...n$, in the initial holding point arrival order, of the ith aircraft in the take-off queue.

With the delay measured in seconds and separation rules specifying a minimum number of minutes separation, the constants α and β were chosen to be 1 and 5 respectively to ensure that reducing the delay was the primary objective and reducing the reordering was only secondary.

4.3 Testing the search algorithms

We aim to determine the feasibility of a meta-heuristic based approach to real-time scheduling of aircraft at Heathrow given the holding point constraints that must be considered at Heathrow. We therefore test our algorithms by providing them with static problems of a type that may occur in a real system, where there is limited visibility of future aircraft and some constraints upon what can be done with aircraft already in the holding point. We form a series of these problems by applying a rolling window of 25 aircraft at a time to each input dataset and applying the results of each search to the input for the next search. In a real system not all suggested reorderings will be accepted, as the controller has a number of other objectives to keep in mind. Here we are assuming that the meta-heuristic order will always be accepted. It is important to attempt to automate the system, so that it can be tested in an objective rather than subjective manner, even though this is not how it would be used in practice.

An initial schedule was first built for the first 25 aircraft.

1. Add the first 20 aircraft to the system.
2. Run the search algorithms for 10000 evaluations. Keep the best result found.
3. Fix the take-off order, take-off time and traversal paths of the first 5 aircraft to take off. Traversal paths for aircraft overtaken by these aircraft were also fixed.
4. Add the next 5 aircraft to the system.
5. Run the algorithms for 5000 evaluations. Keep the best result found.

A second, iterated stage was then entered. This is the stage that more closely emulates what will happen in practice, with some aircraft having take-off slots or traversal paths already assigned. Each iteration took between 0.4 and 0.8 seconds.

1. Fix the take-off order, take-off times and traversal paths of the first 10 aircraft to take off. Again this also fixes the traversal paths of all aircraft they overtake.
2. Add the next aircraft to the system.
3. Remove the first aircraft from the system.
4. Run the search algorithms for 5000 evaluations. Keep the best result.
5. If there are no more aircraft to add then stop, otherwise return to step 1.

As aircraft are removed from the system the take-off order is recorded and at the end, the combined schedule of all of the departures is built and evaluated. This test was applied ten times to each dataset for each of the algorithms.

We have two main concerns in our testing. Firstly, we must verify whether our algorithms can find good results for the sub-problems within a very short search time, to verify their feasibility for use in a real-time system. Secondly, although the searches are considering only a subset of aircraft at once, it is the value of the entire schedule as a whole which actually matters. We would like to verify that solving the sub-problems will give good results for the entire schedule, validating the approach for a real system. To answer both of these questions we evaluate the final schedule as a whole, predicting take-off times and calculating a total delay for all of the aircraft in the dataset.

5 Results

5.1 Input data and assumptions

Historical recorded data was used for the evaluation. Three datasets were used with different numbers of aircraft (123, 189 and 299 respectively).

The most convenient holding point entrance for the allocated stand was assigned to each aircraft. The real holding point arrival times from the historic data were used. In a real system, precise arrival times would not be known until the aircraft actually arrived at the holding points and estimated arrival times would have to be used until then.

Recorded data shows that it takes a minimum of just over a minute for an aircraft to traverse the holding point structure and get airborne but this time can vary widely. For this paper, all holding point traversal times were assumed to be equal and independent of the route taken, as only good paths were used. Two values for this time were tested: one and two minutes. A traversal time of one minute has the advantage of allowing aircraft to arrive, enter the runway and take-off very quickly, which is what often happens in practice at quiet periods. A two-minute traversal time, although no longer allowing fast entry at times when this is possible, seems better suited for the model in many ways as it can be assumed to account for some of the uncertainty in arrival time or traversal time in real life.

The real situation would have some aircraft already in the holding point. We simplify these tests by always starting aircraft at the holding point entrances to avoid having to make predictions for the positions of aircaft within the holding point. The danger of not predicting holding point positions for aircraft already in the holding point is that the reordering of earlier aircraft that have already taken off may have enforced certain manoeuvring upon the aircraft that haven't taken off yet. To ensure that restarting aircraft at the holding point entrances does not increase the flexibility of later take-offs we leave earlier aircraft in the system until they can no longer have any effect on the aircraft that haven't yet taken off, thus re-enforcing the manoeuvring on the later aircraft.

Our model can easily consider aircraft already in the holding point by modifying the earliest take-off time appropriately and starting the feasibility check with the aircraft already in the intermediate nodes rather than at the holding point entrance. This would considerably reduce the complexity of the feasibility check in the holding point graph, but introduce a great deal of complexity into the test simulation with the need for a position prediction system.

5.2 Total delay on aircraft

The test schedule was executed ten times for each of the search approaches, on each set of data, for both one and two minute holding point traversal times. The mean values of the total delay in seconds for the ten runs are shown in the tables below. The best figures are presented in bold.

Table 1. Comparison of mean delays - 1 minute traversal time

Algorithm	Dataset 1	Dataset 2	Dataset 3
Manual schedule	55140	136168	103692
First Descent	23548	49966	51438
Steeper Descent	**23511**	49158	50977
Simulated Annealing	**23511**	**48613**	50788
Tabu Search	23516	48767	**50661**

Table 2. Comparison of mean delays - 2 minute traversal time

Algorithm	Dataset 1	Dataset 2	Dataset 3
Manual	62244	142828	121632
First Descent	**30831**	59170	69377
Steeper Descent	**30831**	58275	68916
Simulated Annealing	**30831**	57815	68728
Tabu Search	**30831**	**57504**	**68601**

5.3 Search times

We aim to verify the feasibility of implementing a meta-heuristic based system to provide real-time advice to a runway controller. One of the key objectives for this research is that results must be returned extremely quickly from each individual search. Although the important consideration for our research is the search time for a single iteration, the total test time is useful for evaluating the relative speeds of the algorithms. Tables 3 and 4 give the mean execution time, in seconds, for the tests performed with each of the four algorithms.

Table 3. Comparison of search time and total time - 1 minute traversal time

Algorithm	Dataset 1	Dataset 2	Dataset 3
First Descent	69.4	114.0	193.1
Steeper Descent	67.3	110.1	187.5
Simulated Annealing	71.6	116.5	197.8
Tabu Search	80.9	132.4	225.4

Table 4. Comparison of search time and total time - 2 minute traversal time

Algorithm	Dataset 1	Dataset 2	Dataset 3
First Descent	69.2	114.7	194.5
Steeper Descent	67.7	110.9	199.1
Simulated Annealing	72.0	117.2	238.2
Tabu Search	80.9	132.0	225.4

5.4 Evaluation of the results

The meta-heuristic solutions provide much lower total delays than the manual solution and this provides significant evidence for the high value of such approaches. However, there are a number of reasons why our automated solutions are so superior (in terms of delay). In fact, the manual solutions are very good, with very few separations above the minimum. These reasons are outlined below.

1. Maximising throughput is not the same as minimising delay. The controller is trying to maximise throughput not directly to minimise total delay. Minimising delay will have the effect of moving larger separations as late as possible in the schedule. Minimising the delay will maximise the throughput but the converse is not true. For example assume a six minute period with only three aircraft available to take off. Two minute separations would give the same throughput as one minute separations but a lot larger delay. Where larger separations will be necessary, a runway controller may sometimes wish to have them earlier to avoid delaying aircraft which take advantage of these to cross the runway.
2. Some aircraft have a Calculated Time of Take-off (CTOT) which effectively designates a fifteen minute take-off time slot. It is important that such aircraft take off within this window. For the results in this paper, we have no CTOT information so we assumed no CTOT limitations.
3. In bad weather, a Minimum Departure Interval (MDI) could be applied to some routes. This temporarily increases the minimum separation allowed between aircraft using certain routes and so can increase delay. We have no data for whether any MDIs were present on the specified days so were forced to exclude MDIs from the evaluation.

4. This is a multi-objective problem and minimising delay only looks at one objective. Many conflicting objectives need to be satisfied and this is one reason why an automated solution can only ever be advisory.
5. Taxi times are not actually identical or predictable. We have no way of knowing whether certain aircraft were exceptionally slow or fast in practice.
6. The meta-heuristics have more knowledge about the future than the runway controller did. Sometimes a good order from the meta-heuristics has been a result of knowing which aircraft are going to be arriving later. Reducing the load on the runway controller via an advisory system should allow a runway controller to take account of these later arrivals themselves; something they do not currently have the time to do.

Minimising the delay is a good way to try to ensure maximal throughput of the runway as it makes it easier to reschedule as new aircraft enter the system.

The fact that the meta-heuristics give better delays than the manual solution means that they hold significant promise for forming the basis of an advisory system. By reducing the work load of the runway controller and allowing more aircraft to be considered than are currently in the holding point structure it should be possible to reduce the delay and increase throughput in practice.

Dataset 1 was from a less busy time of the day than the other two datasets. There were less possibilities to reorder aircraft as there were less aircraft in the holding points at any time. All but the first descent algorithm found the same good schedule for the aircraft in this dataset, the mean values of 23511 and 30831 were also the minimum values found for this dataset, by any of the algorithms. The tabu search failed on one execution to find this good schedule hence the slightly higher mean for the tabu search with one minute traversal time.

Datasets 2 and 3 were from busier times of the day. For both traversal times, for both datasets 2 and 3, student t-tests showed that tabu search performed significantly better than the steeper descent algorithm and that both simulated annealing and tabu search performed significantly better than the first descent algorithm, with a confidence level of 99% in each case.

The simulated annealing algorithm gave good results across the datasets. It got the best results for dataset 2 on table 1 and equal best on dataset 1 for both tables. Student t-tests performed on the results, however, failed to show a significance in the difference between the results for simulated annealing and tabu search, for either of the traversal times for dataset 2, despite the difference in the mean values of the results.

With ten executions of the algorithms on each dataset for each traversal time, there are forty executions that can be compared for these datasets. Tabu search gave better results that the steeper descent algorithm on 39 of the executions and the same result on the other execution. The only difference between the two approaches is the presence of the tabu list so we conclude that the tabu list is contributing to the success of the search.

Tabu search produced the best result for dataset 3 on table 1 and the best results for all three datasets on table 2, although all of the automated results got equal best results for dataset 1. Student t-tests showed that tabu search performed significantly better than simulated annealing for both traversal times for dataset 3, with a confidence level of 99%.

However, there is a significant cost to maintaining and checking the tabu list, this being shown in the greater time that the tabu search takes to perform the search.

We aimed to determine whether a meta-heuristic approach could solve the scheduling problem fast enough to be of use to a real time system and whether an approach which solves a number of sub-problems could attain a good overall delay for the entire schedule. The good overall delay for the schedule obtained when applying either the Tabu Search or Simulated Annealing algorithms to the problem assures us that the meta-heurisitic approach is a promising approach for a real-time decision support system for a runway controller as it can, with a very short search time, provide very good results for the sub-problems a real controller would have to deal, leading to very good overall delays.

6 Conclusions

The departure problem is a complicated one due to the many constraints upon the schedule and the sequence-dependent separations between aircraft. Most of the existing research has looked at the arrivals problem rather than the departure problem where the separations are based on the wake vortex categories of aircraft. In that case it is only necessary to check the separations between adjacent aircraft. However, the route and speed based separations at Heathrow are not only asymmetric, but also do not obey the triangle inequality, so it is not sufficient merely to look at adjacent pairs of aircraft. A schedule that provides safe separations for all adjacent pairs of aircraft will not necessarily provide safe separations for other aircraft pairs.

Many different techniques have previously been applied to this problem yet none account for the physical constraints upon reordering that exist at an airport like London Heathrow. There are many constraints upon a departure system that are not normally modelled and any solution should also aim to minimise other aspects such as controller and pilot workload and fairness.

This paper has presented a model for the system that can take account of the real life constraints. The initial results presented here include some of the constraints that are particularly important at Heathrow.

The results show that it is feasible to check the effects of the holding points after schedules have been generated and that the meta-heuristics will still perform well in the limited time that they have.

From the experiments carried out here we can conclude that Tabu search obtained the best delays overall, although it was the worst performer on dataset 1 in table 1 and it did take the longest to run due to the overheads associated with the tabu list. Simulated Annealing performed well across all the experiments but not always as well as tabu search. Further research will include much more experimentation to see whether these results apply in general for the Heathrow problem.

Both the Tabu search and Simulated Annealing algorithms perform well in the very short search time permitted. We can determine from the results that the meta-heuristic searches form a promising basis for an advisory system for a controller as they are suggesting schedules which improve on the delay in the schedules the controllers are currently implementing.

Further research will add to this model and evaluate the effects of the constraints that have not yet been included. Implementation using genetic algorithms and hybridised meta-heuristics are also planned.

7 Acknowledgements

This work was supported by EPSRC (The Engineering and Physical Sciences Research Council) and NATS (National Air Traffic Services) Ltd. from a grant awarded via the Smith Institute for Industrial Mathematics and Systems Engineering.

References

1. Abela J, Abramson D, Krishnamoothy M, de Silva A, Mills G (1993) Computing optimal schedules for landing aircraft. Proceedings of the 12th National Conference of the Australian Society for Operations Research, Adelaide, July 7-9, 1993, p71-90 Available at: http://www.csse.monash.edu.au/davida/papers/asorpaper.pdf [30 March 2004].
2. Anagnostakis I, Clarke J-P, Böhme D, Völckers Uwe (2001) Runway operations planning and control, sequencing and scheduling. Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS-34), Hawaii, January 3-6, 2001.
3. Anagnostakis I, Idris HR, Clarke J-P, Feron E, Hansman RJ, Odoni AR, Hall WD (2000) A conceptual design of a departure planner decision aid. 3rd FAA/Eurocontrol International Air Traffic Management R & D seminar, ATM-2000, Naples, Italy, June 13-16, 2000. Available at: http://atm-seminar-2000.eurocontrol.fr/acceptedpapers/pdf/paper68.pdf [30 March 2004]
4. Anagnostakis I, Clarke J-P (2003) Runway operations planning, a two-stage methodology. Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36), Hawaii, January 6-9, 2003.
5. Anagnostakis I, Clarke, J-P (2002) Runway operations planning, a two-stage heuristic algorithm. AIAA Aircraft, technology, Integration and Operations

Forum, Los Angeles, CA, October 1st-3rd, 2002. Available at: http://icat-server.mit.edu/ Library/Download/167_paper0024.pdf [30 March 2004]

6. Beasley JE, Krishnamoorthy M, Sharaiha YM, Abramson D (2000) Scheduling aircraft landings - the static case. Transportation Science 34:180-197
7. Beasley JE, Sonander J, Havelock P (2001) Scheduling aircraft landings at London Heathrow using a population heuristic. Journal of the Operational Research Society 52:483-493
8. Bianco L, Dell'Olma P, Giordani S (1999) Minimizing total completion time subject to release dates and sequence-dependent processing times. Annals of Operations Research 86:393-416
9. Craig A, Ketzscer R, Leese R A, Noble S D, Parrott K, Preater J, Wilson R E, Wood D A (2001) The sequencing of aircraft departures. 40th European Study Group with Industry, Keele 2001. Available at: http://www.smithinst.ac.uk/ Projects/ESGI40-NATS/Report/AircraftSequencing.pdf [30 March 2004]
10. Ernst A T, Krishnamoorthy M, Storer R H (1999) Heuristic and Exact Algorithms for Scheduling Aircraft Landings Networks, Vol 34, Number 3, p229-241
11. Idris HR, Delcaire B, Anagnostakis I, Hall WD, Clarke JP, Hansman RJ, Feron E, Odoni AR (1998a) Observations of departure processes at Logan airport to support the development of departure planning tools. Presented at the 2nd USA/Europe Air Traffic Management R&D Seminar ATM-98, Orlando, Florida, Dec 1st-4th 1998. Available at: http://atm-seminar-98.eurocontrol.fr/ finalpapers/track2/idris1.pdf [15 December 2003]
12. Idris HR, Delcaire B, Anagnostakis I, Hall WD, Pujet N, Feron E, Hansman RJ, Clarke JP, Odoni A (1998b) Identification of Flow Constraint and Control Points in Departure Operations at Airport Systems. Proceedings of the AIAA Guidance, Navigation and Control conference, Boston, MA, August 1998.
13. van Leeuwen P, Hesselink H, Rohling J (2002) Scheduling Aircraft Using Constraint Satisfaction. Electronic Notes in Theoretical Computer Science 76.
14. Newell GF (1979) Airport Capacity and Delays. Transportation Science 13:201-241
15. Trivizas DA (1998) Optimal Scheduling with Maximum Position Shift (MPS) Constraints: A Runway Scheduling Application. Journal of Navigation 51:250-266