TRANSPORTATION SCIENCE
Vol. 41, No. 1, February 2007, pp. 90–106
ISSN 0041-1655 | EISSN 1526-5447 | 07 | 4101 | 0090

DOI 10.1287/trsc.1060.0163
© 2007 INFORMS

Hybrid Metaheuristics to Aid Runway Scheduling at London Heathrow Airport

Jason A. D. Atkin, Edmund K. Burke
School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, United Kingdom {jaa@cs.nott.ac.uk, ekb@cs.nott.ac.uk}

John S. Greenwood
NATS CTC, 4000 Parkway, Whiteley, Fareham, Hampshire, PO15 7FL, United Kingdom

Dale Reeson
National Air Traffic Services, Heathrow Airport, Hounslow, Middlesex, TW6 1JJ, United Kingdom

Although London Heathrow is one of the busiest airports in the world, it has only one runway for use by departing aircraft at any time. Separations are required between each pair of aircraft at take-off—depending on their routes, weights, and speeds—to ensure safety. Efficient scheduling of the aircraft for take-off can reduce the total separations and increase throughput. A runway controller is responsible for take-off scheduling. This is a very intensive job with responsibility for all communication with aircraft and continuous monitoring to assure safety. The high workload limits the number of aircraft that the controller can take account of when scheduling. The geometry of the runway holding points adds physical constraints to the reordering of aircraft that are usually ignored in the academic literature. We present models for evaluating a schedule and determining the effects of the physical constraints. We propose a hybrid metaheuristic system that takes account of more aircraft than a human controller can handle, and so can aid the runway controller by recommending schedules that anticipate some future problems. We present results to show the effectiveness of this system, and we evaluate those results against real-world schedules.

Key words: airport runway operations; scheduling; metaheuristics
History: Received: March 2005; revision received: February 2006; accepted: March 2006.

1. Introduction

London Heathrow Airport is one of the busiest airports in the world, but it is situated on an extremely small area of land, for such a major airport. For reasons of passenger and airline preferences, traffic at the airport is not evenly spread throughout the day. Heathrow consequently suffers from congestion at certain times of the day.

The throughput of an airport is often measured as the total number of take-offs and landings per hour. There are two parallel runways at Heathrow. Newell (1979) showed that the best throughput for this configuration is gained by using both runways for both arrivals and departures. However, as a result of a local agreement to control noise, it is only possible to use one runway at a time for departures at Heathrow.

A symbolic representation of the layout of the airport is given in Figure 1. The two runways can be used in either direction, depending upon the direction of the wind. The runway name, labelled in Figure 1, depends upon the current direction of use. For example, the northern runway is called 27R when used by aircraft taking off or landing heading west. There are four terminals at London Heathrow, labelled T1 to T4 in Figure 1, although a fifth is currently being built. The departure system involves a ground controller directing aircraft from the stands at the terminals, around the taxiways, to holding points, labelled HP in Figure 1, near the end of the current departure runway.

There are many similarities between the departure systems of the major airports of the world. The departure flow at Logan Airport was examined in Idris et al. (1998, 1999), comparing it to other major airports. Various flow constraints upon the departure operations were identified, and the runway was seen to be the key constraint. A queuing model was proposed in Idris et al. (2002) to better predict the time from the stand to take-off, identifying and utilising the fact that the runway represents the bottleneck in the departure system.

At the departure runway, separations need to be imposed between aircraft taking off to ensure safety and control congestion. The minimum permissible separation between two aircraft depends upon the weight classes, departure routes, and speeds of the aircraft. Changing the order of take-off will often change the separations that need to be imposed between

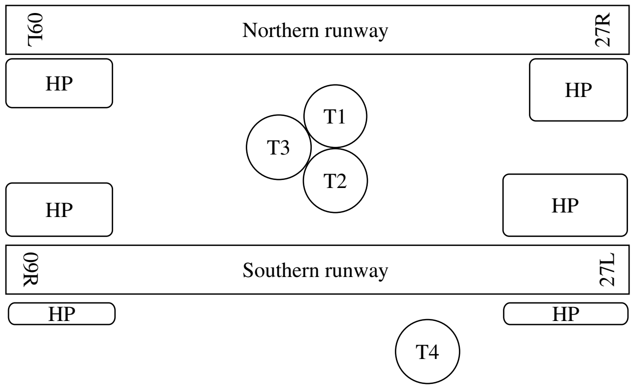

Figure 1 The Layout of London Heathrow Airport

aircraft, and hence affect the throughput of the departure runway. It is, therefore, important to attain a take-off order that will avoid large separations, helping to maintain a high throughput on the runway and low delays for aircraft. This problem is often referred to as the *departure problem*. At Heathrow, the variance in taxi times, difficulty in predicting in advance the time at which aircraft will be ready for pushback, and the contention for the stands, where unnecessarily holding an aircraft at a stand may deny that stand to an arrival that needs it, collaborate to ensure that it is far more practical to reorder the aircraft at the holding points than to attempt to do so at the stands.

A runway controller is responsible for reordering the aircraft within the holding points. The structure of the holding point determines what reordering can be performed and how difficult the reordering is to achieve. The runway controller is responsible for all communication with aircraft using the runway, and for continual monitoring to ensure safety. The task of sequencing is currently performed manually, with flight progress strips that may be used as "working memory." The speed at which decisions need to be made in this real-time system limits the number of aircraft that the controller can account for at any time.

A number of researchers have looked at the departure problem in the past. Anagnostakis et al. (2000) and Anagnostakis and Clarke (2002, 2003) analysed the problem and proposed a two-stage departure planner. Anagnostakis et al. (2001) proposed a search tree to solve the problem, and branch and bound or an A* algorithm were recommended for trimming the tree. Trivizas (1998) suggested a dynamic programming approach to solve the departure-order problem, substantially reducing the search space by limiting the possible number of aircraft that are considered for any place in the schedule. At Heathrow, both the manually produced and our automatically generated schedules often require an aircraft to move forward or backward up to seven or more places in a take-off schedule. Van Leeuwen, Hesselink, and Rohling (2002) presented a constraint satisfaction-based model for the departure problem. However, the problems that were considered were much smaller than the Heathrow departure rate dictates.

The effects of one of the simplified holding-point structures at Heathrow airport were considered in Craig et al. (2001), and a dynamic programming solution for scheduling take-offs was given. As the flexibility of the holding points and the number of possible positions for aircraft increase, the feasibility of a dynamic programming approach, accounting for aircraft positions, decreases. The holding points at Heathrow are a lot more flexible than the one presented in Craig et al. (2001), making the solution space prohibitively large. We presented a solution method for a simplified problem in Atkin et al. (2004), including the effects of the holding-point structures.

There are some similarities (but also important differences) between the arrival and departure processes for the runways at an airport. A genetic algorithm was presented in Abela et al. (1993) to give an approximate solution for the arrivals problem for a set of aircraft with landing time windows, and a branch-and-bound algorithm was presented for solving the arrival problem exactly. Mixed-integer zero-one formulations were presented for the problem in Beasley et al. (2000) and genetic algorithms were shown to be effective in Beasley, Sonander, and Havelock (2001). Ernst, Krishnamoorthy, and Storer (1999) used a branch-and-bound algorithm to determine an arrival order for aircraft, using a network simplex method to determine optimal arrival times for any partial ordering of aircraft. Bianco, Dell'Olma, and Giordani (1999) showed the equivalency to the asymmetric travelling salesman problem (ATSP) with release dates as long as separations only needed to be considered between adjacent aircraft. They presented heuristic algorithms for solving the problem.

The differences between the arrival and departure problems lie in the details of the separation rules, constraints, and objective functions. The ATSP equivalence does not hold for the departure problem at Heathrow because the flight time of an aircraft taking off can be affected by more than just the immediately preceding flight. An example of this can be seen in §4.4.

To increase the throughput of the departure process at Heathrow, it is vital to increase the throughput of the departure runway. 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. We will identify these in the problem description below. We will then present our solution method for the problem and a model to both predict take-off times for aircraft and to calculate a resultant cost for a take-off schedule.

Finally, we will present some results to show the efficacy of our approach and will draw some conclusions.

We intend to answer two questions in this paper. First, can a computerised decision-support system solve the departure problem at Heathrow in a time fast enough to be of use? Second, would a decision-support system actually be able to help the controllers improve upon what they achieve manually?

2. Problem Description

The objective of this research is to increase the throughput of the departure runway without negatively affecting safety, while satisfying the various constraints on the reordering detailed below. We propose an automated advisory system to help the runway controller to obtain take-off orders that would improve the throughput and reduce delay.

The throughput of the runway is limited by the separations that must be enforced between aircraft taking off. Reordering aircraft can often reduce the total separations needed in the take-off schedule and so increase the throughput of the runway. There are various constraints on the reordering.

When aircraft take off, they have to wait for the wake vortices left by earlier aircraft to dissipate. As larger aircraft create stronger wake vortices and lighter aircraft are more affected by them, a larger separation is required whenever a lighter-category aircraft follows a heavier-category aircraft. Additionally, aircraft depart along fixed departure routes, called standard instrument departure routes (SIDs). To ensure that aircraft will have the correct in-flight separation and that airspace is not congested, the frequency of departures along each SID and group of SIDs is controlled. This is achieved by applying a minimum separation between aircraft, depending on their SID route and speed group.

Some aircraft are assigned departure windows to avoid congestion en route and at busy destination airports. This window is based upon a calculated time of take-off (CTOT), which specifies a target time for take-off. Aircraft cannot take off more than five minutes before this target time or more than 10 minutes after the target time, giving a 15-minute take-off window. At Heathrow, specific rules are in place to allow extensions of up to five minutes to be arranged for a few aircraft a day. Although these extensions should be avoided wherever possible, using one of these extensions is by far preferable to having to arrange a new CTOT for an aircraft.

To improve the throughput and ensure compliance with CTOTs, the aircraft are reordered within the holding points by a runway controller. The configuration of the holding points at the ends of the runways, which are different at each runway end, will determine what reordering operations can take place and the costs, in terms of time and effort, involved in each operation. The holding-point structures can be considered to be one or more entrance queues to some manoeuvring space before the aircraft enter the runway.

A decision-support system for the runway controller must ensure that any reordering suggested is not only possible, given the physical holding-point restrictions, but also that it will not require excessive time and effort from the controller or pilots involved. It is also desirable to control inequitable positional delay in the take-off queue. If two aircraft have similar characteristics, it is often preferable to schedule the one that arrived at the holding point first to take off first. Additionally, some aircraft have characteristics that will usually mean a larger separation either before or after them. The system must ensure that these aircraft are not unduly penalised.

The departure problem at Heathrow is constantly evolving as current aircraft take off and new aircraft become available for sequencing. In deciding what overtaking is required, a controller uses the knowledge available at that time to make a decision. We call this problem the *static problem*, determining what to do by considering the attributes and positions of the aircraft at any instant in time. The controller can be considered to be solving a series of these static problems, each slightly different from the last, as new aircraft enter the system or aircraft react to the instructions of the controller. Any decision-support system would have to be able to do the same thing, attempting to reorder the aircraft currently under consideration, with imperfect knowledge of the aircraft that may enter the system in the future, without unduly constraining the possibilities for scheduling later aircraft. The performance of the departure system will be assessed not upon the solution of a static problem, but upon the overall performance throughout the day, so it is vital that decisions made in solving the static problem do not have a deleterious long-term effect. Furthermore, as we explain in §5.4.3, it is important that there is some similarity between the solutions to the static problems.

3. Solution Approach

We propose a hybrid approach that uses different search methodologies to solve the static problem. Our system includes aircraft as soon as they leave their stands, and so includes both aircraft at the holding point and those on the taxiways. Only the aircraft actually in the holding point will be under the control of the runway controller, so only these will have been given any instructions. The aircraft on the taxiways are included in the search to give the system information about what is due to reach the holding point soon so that the scheduling of those aircraft already in the holding point will not cause problems to the later schedule, ensuring that the solution of the static problems will provide a good solution to the dynamic problem. Our objective function also aims to ensure that the solution to the current static problem will not overly constrain the scheduling of later aircraft. We evaluate our system using tests that simulate the dynamic behaviour of the real world, presenting a series of static problems to the decision-support system. Our system performs one search for each static problem. Later searches change and refine the suggested solution from earlier searches in response to the changing circumstances.

In this paper, we present a hybridised metaheuristic approach. The term "metaheuristic" describes a class of algorithms that impose a higher-level control upon a heuristic search, with the intention of avoiding the search becoming stuck in a local optimum. A comprehensive introduction to metaheuristics can be found in Glover and Kochenberger (2003) and Burke and Kendall (2005). The search time to find a good take-off order will be perceived by the runway controller as a delay between the situation changing and the decision-support system responding. It is, therefore, vital that this delay be as low as possible. Metaheuristic searches are ideal for this problem, as there is not usually time for an exhaustive search to be performed and the characteristics of the holding point make it difficult to build a solution in the way a dynamic programming or branch-and-bound approach would require.

We use a metaheuristic called *tabu search* to search for good take-off orders. Each order found by the tabu search is tested to ensure that the reordering is feasible given the holding-point constraints, and is rejected if not. Details of the search and the feasibility test are given below.

3.1. Tabu Search

Tabu search was first introduced in Glover (1986) and is covered extensively in Glover and Laguna (1997). Since then it has been applied to many problems by many different practitioners. An introductory tutorial can be found in Gendreau and Potvin (2005). Like any heuristic, the tabu search does not guarantee that the globally optimal solution to the problem will be found, but it usually finds a good solution. Indeed, our results show that our application of it to this problem performs well even with a very short search time.

Our search considers the problem of finding the best order for take-off for the aircraft currently under consideration. A solution to the search is a take-off order, i.e., a permutation of the aircraft under consideration. Starting at an initial solution, the search investigates different possible solutions by making incremental changes to the current solution. The search is initiated by generating 50 slightly different schedules by making small changes to the initial solution, for instance, by swapping the take-off order of two aircraft. The search then adopts the best of these solutions as the new current solution and generates another 50 different schedules based on this new solution. Again, the best of these schedules is adopted as the new current solution and 50 more new schedules are created based upon the new solution. This is repeated until a stopping criterion is met. Section 3.2 gives much more detail about what is actually performed within a search iteration.

The main problem with the search summarised above is that it can very easily get stuck in locally optimal solutions. The simplest example of this is where there are two solutions, A and B, where A is the best solution that can be reached from B and B is the best solution that can be reached from A. The simple search could easily cycle between the two rather than exploring more of the potential solutions. A tabu search helps to explicitly avoid this cycling by preventing the search from immediately returning to where it has been. After moving from A to B, the search is forced to explore elsewhere, helping it to escape these local optima and giving more possibility of discovering better local optima or even the global optimum.

Our tests move through the test data sets over time, rerunning the search as the situation evolves and the problem changes. The initial solution for each search is generated by taking the best solution found by the previous search and adding any new aircraft that have subsequently entered the system, in the order they are predicted to arrive at the holding point. This solution is guaranteed to be feasible, as defined in §3.4, as the solution to the previous search was feasible and the addition of new aircraft to take-off in arrival order requires no additional overtaking.

Each search is executed for 100 iterations, ensuring that each completes in less than one second on the test PC, a 2.4 GHz Pentium 4. Section 3.2 provides more detail about the implementation of the tabu search, the moves that are used to investigate the search space, the format of the tabu list, and the work that has to be performed in each iteration of the search. As the solution considers only a take-off order, it is important to also verify that the required overtaking is achievable and to determine how to perform it. Section 3.2 also shows how this is performed, while §§3.3 to 3.7 provide more detail about each step.

3.2. A Tabu Search Iteration

This section provides more detail about the steps involved in each iteration of the tabu search, detailing the relationship between the path allocation heuristic, feasibility check heuristic, take-off time prediction model, and objective function; and how together they enable a fast search for better take-off schedules. Each iteration consists of the following steps:

Step 1. Randomly generate 50 new solutions by applying randomly chosen moves to the current solution. There is a 30% chance that a swap move will be used, selecting two aircraft at random and exchanging their positions in the schedule. There is a 50% chance that a shift move will be used to move between two and six aircraft forward or backward in the schedule. Finally, there is a 20% chance that a purely random reordering will be applied to a consecutive group of between three and five aircraft.

Step 2. Assign holding-point paths to the aircraft in each new solution and check for feasibility against the holding-point structure. This is explained in detail in §3.3.

Step 3. Test the feasibility of the reordering by feeding the aircraft through a network model of the holding point. This is explained in detail in §3.4.

Step 4. For each schedule that was determined to be feasible, use the model described in §4.4 to predict a take-off time for each aircraft.

Step 5. Using the take-off times, evaluate a cost for each schedule as described in §4.6.

Step 6. If the feasible solution of least cost is better than the current best solution, then record it as the best solution found so far, even if it involved a tabu move.

Step 7. Adopt, as the new solution, the feasible solution of least cost that did not involve a tabu move, if there is one. Details of the move that is adopted are stored in a tabu list, described below, to avoid the move being reversed in the near future.

The tabu list stores information about the last 10 moves that were made. For each move, the initial position is stored for each of the aircraft that was moved. Any move that places all of the aircraft that moved back into their initial positions, therefore reversing the effect of a move on the tabu list, is declared as tabu and rejected.

As the tabu search only has to consider the take-off order, the search space is far smaller than if it included information about how the aircraft move through the holding point. Also, as the cost of the schedule is based on the take-off order, rather than how it is attained, it is easier to design moves that will move between good take-off orders in this case. The selected moves reflect the characteristics of a good schedule and make it easier to move between good schedules in a single move. For example, where northbound and southbound departures are alternating, moving a single aircraft is much less likely to obtain a better schedule than moving two aircraft. Conversely, if the holding-point movement was included in the search space, reversing the take-off order of two aircraft may require a lot of change in the movement of other aircraft within the holding point, as well as the two concerned.

A large number of different moves are made available to the search in order to reduce the number of local optima. A solution that is a local optimum for one type of move may not be so for another type of move. The percentage distribution of move selection was determined experimentally by running the search on separate test data and evaluating the performance with different percentages of each move. The absence of any of these three move types caused a significant reduction in the performance on the test cases. Tests with a high proportion of shift moves performed well, and the randomisation moves improved the performance when used in moderation. The tests indicated that the search is not particularly sensitive to exact percentage distributions of moves, however.

3.3. Path Assignment Heuristic

A heuristic is used to assign paths through the holding point to aircraft. The heuristic takes account of the different complexities of the various paths for each holding-point entrance and assigns a path based on how quickly the aircraft needs to navigate the holding point. Although the allocation heuristic is different for each holding point, as the available paths differ, a similar approach is used for each.

An example graph for the 27R holding point is shown in Figure 2. Each node represents a valid position for an aircraft. The arcs represent the transitions that aircraft can make between nodes when moving through the holding point. The graph is deliberately overly restrictive to provide a conservative view of feasibility.

A valid holding-point traversal path for an aircraft consists of a sequence of nodes, where each adjacent pair of nodes is joined by an arc, the first node being an entrance node and the final node being the runway. We label the paths according to the nodes on the path, where node R refers to the runway. Each possible path has been discussed with an experienced

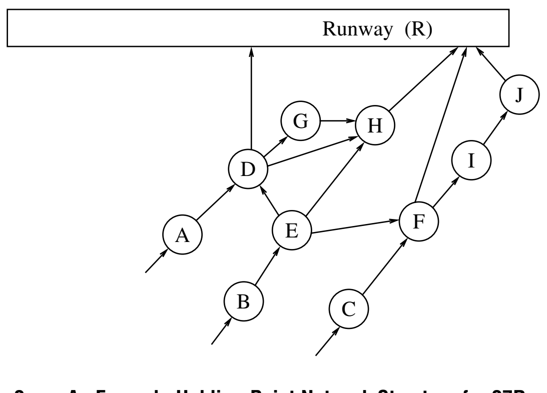

Figure 2 An Example Holding-Point Network Structure for 27R

runway controller to identify which paths it is possible to use and how desirable the use of each path is.

For example, an aircraft that enters the holding point at node B could take a number of paths. BEHR is the most straightforward path to the runway. BEDGHR is a longer path, involving two right-angle turns. This could be used, but is not favoured due to the amount of time and work needed to make the turns. BEFR is a short path, but it involves a significant turn, so it is not desirable. BEFIJR is a longer path, with fewer turns, and could be used by an aircraft that needs to be delayed for a while—for instance, to await the start of its CTOT. BEDHR is not practical, so is eliminated as an option. BEDR is a very short route, allowing aircraft to enter the runway further up than the end. This runway entrance is close enough to the end of the runway to avoid special considerations for intermediate entrances, but is not desirable, especially for larger aircraft where the pilot may require the use of the full runway.

Different path allocation heuristics can be used to reflect different modes of operation, but we will restrict our consideration here to the normal mode of operation, used in the tests we performed for this paper. To control the workload, we allow only the paths BEHR, BEFIJR, and BEDR for aircraft that enter at B. Similarly, we allow only ADHR, ADGHR, and ADR for aircraft starting at A, and CFIJR and CFR for aircraft starting at C. This ensures that none of our schedules will require excessive manoeuvring by aircraft.

Once the allowable paths have been determined, the following heuristic is used to assign paths to aircraft:

Step 1. If the aircraft is already in the holding point, then keep the current path. This reflects the fact that instructions will already have been given to the aircraft.

Step 2. If the aircraft is overtaken by another aircraft from the same entrance, and does not overtake any aircraft from the same entrance, then assign a slower path—for example, ADGHR, BEFIJR, or CFIJR.

Step 3. If the aircraft must overtake a number of aircraft from the same entrance, or must overtake an aircraft that is already overtaking another aircraft, then the other runway entrance may be needed. In this case, assign paths such as ADR or BEDR unless the aircraft needs the full runway for take-off.

Step 4. If the aircraft overtakes other aircraft, then assign the fast path—for example ADHR, BEHR, or CFR.

Step 5. Otherwise, assign the default path—for example, ADGHR, BEHR, CFR. These are usually the fastest paths, but ADGHR is used because it is fast, and using it maintains flexibility in case a later aircraft needs to overtake.

This heuristic is very quick to apply and ensures that the controllers and aircraft do not undergo unnecessary workload. It also ensures that longer paths are only ever allocated to aircraft that are overtaken, and therefore have more time available to cross the holding point. Both of these provide sensible pruning of the solution space, removing solutions that would be undesirable for reasons of workload, an objective not covered by the objective function used by the search. Because this system is intended for decision-support for a runway controller, it is pointless to suggest a schedule that the runway controller would immediately reject.

The take-off schedule is designated to be infeasible if the path allocation heuristic is unable to assign paths to the aircraft. In this case, the rest of the feasibility check is not necessary. This only happens if the desired reordering requires more overtaking than is possible using the selected paths, or if it involves a reallocation of a path to an aircraft for which the traversal path has been fixed. In both cases, these schedules can be considered to have been rejected on the grounds of prohibitive workload.

3.4. The Feasibility Check

A directed graph representation has to be designed for each holding-point structure. An example for the 27R holding point can be seen in Figure 2. The holding-point model is used to verify the feasibility of reordering the aircraft within the holding-point structure, feeding the aircraft through the holding-point structure and ensuring that they can exit the structure onto the runway in the desired take-off order. Each node in the holding-point graph can contain a maximum of one aircraft, so an aircraft can only move if the next node on its path is empty.

The feasibility check starts with a queue of aircraft at each of the holding-point entrances, queued in the order they are predicted to arrive at the entrance. For aircraft already in the holding point, the feasibility check can be started with aircraft occupying the node representing their current position in the holding point, or the next node they will enter if they are between nodes. The experiments performed for this paper did not make assumptions about where aircraft were, and instead always started them from the entrance queue. A live system at Heathrow would have this information available from the ground radar.

Aircraft that have already taken off are left in the feasibility check until they can have no more effect upon the movement of aircraft currently being scheduled. This ensures that previously enforced holding-point movement is still performed by aircraft in the holding point. In all tests performed, a final feasibility check was also executed at the end of the test, including all aircraft in the data set, to validate the feasibility of the entire schedule. Additionally, the schedule can be viewed using a graphical playback tool to ensure that decisions were sensible from the point of view of a controller.

The following algorithm is then used to determine feasibility.

Algorithm 1.

Step 1. Place the first aircraft from each entrance queue at its entrance node.

Step 2. Repeat until all aircraft have entered the runway node.

Step 3. Iterate through each aircraft currently in the graph.

Step 4. If the next node for the aircraft is empty, and moving the aircraft will not block another aircraft, then move the aircraft to the next node on its path.

Step 5. If the aircraft vacated an entrance node, then move the next aircraft in the queue for that entrance into the entrance node.

Step 6. End iteration of aircraft.

Step 7. If no aircraft moved, then declare the schedule infeasible and end.

Step 8. End repeat.

Step 9. Declare the schedule feasible as all aircraft reached the runway in order.

A fast method to ensure that blocking of other aircraft does not occur is explained in §§3.6 and 3.7. Speed is ensured by maintaining partial take-off orders at each node and by performing many of the required calculations for each holding point offline, reducing the amount of work that needs to be performed at the time of the feasibility check. By the end of the feasibility check, every aircraft will have a known path through the holding point, and the order of aircraft passing each node in the holding point will have been determined.

3.5. Traversal Times

The take-off time prediction model in §4.4 requires information about the traversal time that aircraft would be expected to need to cross the holding point. This time can be estimated because both the current position and the traversal path will be known for each aircraft as it approaches the holding point. This information is used to force the schedule to allow the aircraft at least this long to traverse the holding point and line up on the runway. We refer to this as a *minimum holding-point traversal time* because aircraft will have at least this minimum time available. This time should be a pessimistic estimate of how long the aircraft will need to avoid wasted runway throughput due to aircraft not lining up for take-off in time. These values can be provided by controllers and from historical data for the different combinations of current position, path, and aircraft type.

In this paper, we assume an extremely pessimistic value, far larger than would actually be required. We require all aircraft on the most direct, fastest traversal paths to have at least two minutes to make the traversal. Because aircraft are only allocated to the slower paths if they are overtaken, aircraft on the longer paths will have at least three minutes to make the traversal, the aircraft overtaking it will have at least two minutes, and there will be at least a one-minute separation between the overtaking aircraft taking off and the overtaken aircraft following it.

Once aircraft are in the holding point, their traversal paths are not changed. This ensures that the minimum required traversal time will also be fixed. Additionally, aircraft will always have at least a two-minute warning of take-off because the first two minutes of the take-off schedule are fixed, so pilots would be aware of any need to get to the runway quickly.

3.6. Path Suffixes and Partial Orders

It is important, when performing the feasibility check, to be able to determine whether an aircraft can safely enter the next node on its path without blocking another aircraft from reaching the runway in time for take-off. Here we explain a generic method that uses automatically generated holding-point knowledge, partly created offline, to provide two rules that can easily check whether this is the case. Some holding-point-specific additional rules may also be needed, and we discuss this in more detail later.

We define a path suffix for a path and selected node to be a set of nodes on a path beyond the selected node. For example, the possible path suffixes for Node E in Figure 2 are DR, DGHR, DHR, HR, FR, and FIJR. For the workload reasons noted above, only DR, HR, and FIJR would normally be used. Because the possible paths are known before any search begins, the path suffixes for each node, and which paths they relate to, can be calculated offline before any search is performed.

Where two or more paths do not diverge beyond a node, they have the same path suffix at that node. For example, ADGHR and BEDGHR both have the suffix GHR at Node D. Aircraft on these paths must be in the desired take-off order relative to each other prior to entering this node because there is no possibility for them to be reordered beyond that point. Once a desired take-off order is known, the paths' suffixes for each node can be very quickly populated with partial take-off orders by considering each aircraft in take-off order and storing the aircraft at the end of the current list for its path suffix for each node on its path. The lists for each path suffix then form a partial ordering, and an aircraft is only permitted to enter a node if it is the next aircraft on its path suffix for that node. Take-off order is ensured at the runway node, as it has only one path suffix, with no nodes, and all aircraft are on this suffix in take-off order.

3.7. Blocking Other Suffixes

Given two aircraft on different path suffixes that both wish to enter the same node, it is important to know whether one entering the node could block the other from reaching the runway on time. For simplicity, here we define an aircraft priority as its position in the planned take-off schedule. The next aircraft planned to take off will be the highest priority, the one after it the next-highest priority, and so on.

We only permit an aircraft to enter a node in front of a higher-priority aircraft if there is at that time room for it to immediately exit the node to get out of the way of the higher-priority aircraft. An exact feasibility check would only require that the aircraft would be able to exit the node prior to the take-off time of the higher-priority aircraft. However, we have not explicitly included the time dimension in the holding-point movement, so we use the stronger condition of requiring room for an immediate exit of the node. Moving out in front of a higher-priority aircraft risks adding a delay to it, but ensuring that an immediate exit is available ensures that any delay for the other aircraft is minimal.

The availability of an exit node can be ensured by verifying that there is at least one spare node beyond the node of contention that is not on the path of the other aircraft. This check can be performed very easily by first precalculating, for each path suffix for each node, a count of later nodes that are not shared with each of the other route suffixes at that node; then calculating how many of these nodes are occupied. For example, in Figure 2, Node D has three path suffixes: R, HR, and GHR. R has no nodes that are not shared by HR and no nodes that are not shared by GHR. HR has one node that is not shared by R and no nodes that are not shared by GHR. GHR has two nodes that are not shared by R and one that is not shared by HR.

These counts can be easily and automatically calculated offline for each holding-point structure. Assuming that all nodes were empty, the number of nodes that the aircraft could possibly use to move out of the way is given by the count of the not-shared nodes, as the aircraft must be able to move into a node that is not also on the path suffix for the aircraft it is moving out of the way of. By determining how many of these nodes are already in use, the algorithm can verify whether there is at least one free node available into which to move, and if not, then prevent the aircraft from moving.

If all later nodes for the path suffix are used only by aircraft on that path suffix—in other words, every node on the path suffix has only one input arc—then the number of occupied nodes is simply given by the number of aircraft on this path suffix that have already passed the current node but have not yet taken off. If this is not the case, then the number of nodes that are being used by aircraft on other paths must be added to the number used by aircraft on this path suffix. This cannot be performed generically because it requires examination of the holding-point structure but is usually simple to perform, as shown in the following example.

In Figure 2, after Node D the path suffixes GHR and GH converge on Node H. Assume that Aircraft 1 desires to take the path DR and Aircraft 2 the path DGH, and that Aircraft 1 should take off before Aircraft 2. Aircraft 2 is permitted to enter D before Aircraft 1 only if there is an available node for it to exit to, to vacate D. Node D will have a count of how many aircraft have already passed on the suffix GHR and have not taken off. Similarly, the not-shared nodes count will show that there are two possible nodes available. Comparing these two values makes it easy to check whether there is a spare node available to move into. However, Node H is also used by the suffix HR, so it is possible that an aircraft from HR is currently occupying it. In this case, due to the convergence of paths beyond Node D, the algorithm also needs to verify the current contents of Node H and, if the current contents is on suffix HR rather than GHR, reduce the number of available nodes by one. If Node H has no current contents, then it is important to check against the next aircraft that will enter it instead of against the current contents. This will always be known for H because it has only one path suffix, R, so the order in which aircraft must pass H is known uniquely. Doing this ensures that the algorithm will even, for example, account for the fact that if the next aircraft to pass H is following the path BEHR, it will be using one of the nodes on the Node D path suffix GHR.

Appropriate use of data structures ensures that all of these checks are extremely fast at runtime, usually involving a comparison of very few values to verify whether each aircraft can move. The major speed benefit gained is in identifying infeasibility very quickly. As soon as no aircraft can move the schedule is known to be infeasible.

This speed is achieved by the caching of the various holding-point-specific data after performing holding-point-specific calculations offline and by a linear time preprocessing stage to store partial orders of aircraft at nodes. The amount of precalculation that needs to be performed can be vastly reduced by noting that it is only needed for nodes that have at least two input arcs. If there is only one input arc to a node, then there is never any decision about which aircraft should enter it next because it will always be the same as the aircraft that is the next to enter the preceding node. Further reduction in precalculation work and runtime can be obtained by considering only the path suffixes for the paths that are actually being allocated at the time. This is easy to achieve by calculating the counts for the paths used by each of the different path allocation methods that relate to the different modes of operation and by using the value for the allocation method that is currently in use.

4. Formal Model

In this paper we aim to increase the throughput of the runway while attempting to meet all CTOT constraints and limit the workload of the pilots and runway controller. It is important for the objective function to represent the real objectives of the controllers so it has various component parts, explained below. The formal model is required to predict take-off times for aircraft in a potential schedule as well as to calculate the consequent value of the objective function.

4.1. Definitions

Let n be the number of aircraft currently under consideration and let $i \in \{1, \dots, n\}$ be an integer to represent an individual aircraft. Let a_i represent the position of aircraft i in the arrival order at the holding point, so if i is the first aircraft to arrive, then $a_i = 1$. Let c_i be the position of aircraft i in the take-off order, so if i is the second aircraft to take off, then $c_i = 2$.

As mentioned earlier, aircraft follow specific paths through the holding point. Let each possible path through the holding point be denoted by a unique integer, and let the path that has been assigned to aircraft i be denoted by t_i.

Let v_i be an integer representing the weight category of aircraft i. Let s_i be an integer to represent the speed group of aircraft i and r_i be an integer to represent the SID route of aircraft i. Let h_i be the time aircraft i entered the holding point and d_i be the take-off time of aircraft i in the schedule.

Where aircraft have a CTOT time allocated, then let b_i and l_i be the earliest and latest times, respectively, at which aircraft i can take off while complying with the CTOT, so $[b_i, l_i]$ denotes the 15-minute CTOT window. Where aircraft i has no CTOT allocated, then set b_i to be a large negative value and l_i a large positive value so that the aircraft effectively has a CTOT window spanning the entire duration of the take-off schedule.

4.2. Positional Delay

It is better to avoid unnecessary overtaking or reordering of the aircraft. Aircraft being overtaken need to wait somewhere while being overtaken, occupying part of the holding-point structure and limiting the reordering that can be performed on other aircraft. Additionally, overtaken aircraft are being penalised, so this penalty must be controlled.

The amount by which aircraft are overtaken is given by the positional delay, which can be calculated for each aircraft, i, as $\max(0, c_i - a_i)$. Small positional delays are inconsequential, but larger delays become an increasing problem both in terms of unfairness and in constraining later reordering. We therefore include a factor in the objective function, Formula 3, which is proportional to the sum of the squares of the positional delay for each aircraft.

4.3. Throughput, Duration, and Delay

For the static problem, the throughput of the runway could be considered to be the inverse of the duration of the take-off schedule. Rather than directly minimising the duration of the schedule, we instead aim to minimise the total holding-point delay for the aircraft in the schedule. The holding-point delay for aircraft i, the time spent within the holding point, can be calculated as $d_i - h_i$. As the aircraft will always take time to traverse a holding point, this "delay" consists of both travel time and queueing time.

Both duration-based and delay-based measures penalise schedules that include large separations between aircraft, but minimising delay has the effect of penalising larger separations more if they are earlier in the schedule than if they are later. Reordering aircraft so that the larger separations are placed later may also help to reduce separations overall; it allows more time in which further aircraft may appear, and these may permit smaller separations from those aircraft currently in the queue.

Although the separations depend on the circumstances, some examples should illustrate this. As there will be a larger separation after a heavy aircraft if it is followed by a medium or light aircraft than if it is followed by another heavy aircraft, it can make sense to group larger aircraft together. If a larger separation is due to a heavy aircraft, then encouraging schedules where the larger separation is later in the schedule increases the possibility of grouping the heavy aircraft with other heavy aircraft that may arrive at the holding point later. Similarly, it is often possible for a later aircraft on a different route to take off between two aircraft on similar routes without affecting the take-off times of these aircraft. Scheduling these larger separations associated with aircraft on similar departure routes later increases the possibilities for fitting in aircraft not currently available at the holding point.

Where the last aircraft in the take-off schedule needs to be delayed for some reason (for example, to wait for the start of its CTOT slot), the duration of a schedule would be artificially increased. A duration-based minimisation would make no effort to decrease the separations of the previous aircraft because this would have no effect on the duration. A delay-based approach will reduce the separations of the earlier aircraft, effectively leaving the larger gap before the delayed aircraft in a place where new arrivals at the holding point could use it. Finally, the calculated delay that aircraft experience at the holding point directly relates to the real time that aircraft spend at the holding point, and therefore to journey delay, fuel costs, and pollution.

4.4. Predicting Take-off Times

To determine a value for the holding-point delay and CTOT compliance, a take-off time must be known for each aircraft. This often has to be predicted. At busy times, when it is imperative to get the highest throughput, the aircraft usually take off as early as possible. This will mean that as one takes off, the next lines up and waits for the required separation to be attained before it sets off. To predict a take-off time for an aircraft, we calculate the earliest possible take-off time for each aircraft that has not taken off and assume that all aircraft take off as early as possible.

There is an important difference between aiming to obtain optimal take-off times given a take-off order and the analogous case for arrival scheduling of searching for optimal landing times given a landing order. In take-off scheduling there is never a benefit from delaying the take-off of an aircraft beyond the earliest time it can do so. It is, therefore, reasonable to assume that all aircraft take off as early as they can. In models for arrival scheduling this is not always the case; see, for example, Beasley et al. (2000) and Ernst, Krishnamoorthy, and Storer (1999). Landing earlier than a target time may require more fuel burn if the aircraft has to fly faster than its optimal cruise speed, and some models aim to minimise fuel burn rather than minimising landing times.

We define a function $V(v_i, v_j)$ to calculate the required wake vortex separation and $R(r_i, s_i, r_j, s_j)$ to calculate the required route and speed separation from the weight categories, v_i and v_j; departure routes, r_i and r_j; and speed groups, s_i and s_j; of leading aircraft i and following aircraft j. These functions are defined to return standard separation values in accordance with current regulations. The separation rules will change depending upon which runway is being used for departures. The runway controller also has discretion to change some in good weather and, at times of congestion in the airspace, some will increase. A fully operational decision-support system would allow these separations to be modified by appropriately modifying the values returned by $R(r_i, s_i, r_j, s_j)$.

The current wake vortex separation rules obey the triangle inequality so that $V(v_i, v_j) + V(v_j, v_k) \geq V(v_i, v_k)$ for aircraft that take off in the order i, j, k. SID route and speed separations do not; therefore, it is not possible to ensure that all separations are maintained by merely ensuring sufficient separation between adjacent take-offs. For example, imagine that a slow aircraft heads north, followed by a faster aircraft heading south, then a faster aircraft heading north. A northbound followed by a southbound departure, or vice versa, needs a one-minute separation, often regardless of speed class because the flight paths diverge so quickly. A fast northbound aircraft following a slow northbound aircraft may need a three-minute separation, but ensuring the separations between adjacent departures would only guarantee a two-minute separation. There are many other similar cases, some of which do not even need the aircraft speed groups to differ.

Given the above definitions, the earliest take-off time, e_i, for which all separations are maintained, can be calculated by Equation (1). Even though the wake vortex separations currently obey the triangle inequality, we do not assume this, so we verify the wake vortex separations for all previous aircraft. For efficiency at runtime, as this will be used by a real-time system, the value of $\max(V(v_j, v_i), R(r_j, s_j, r_i, s_i))$ can therefore be precalculated for all aircraft i and j prior to any search being performed:

$$e_i = \begin{cases} 0 & \text{if } c_i = 1, \\ \max\limits_{j \in \{1,\ldots,n\} \mid c_j < c_i} \big(d_j + \max(V(v_j, v_i), \\ \qquad\qquad R(r_j, s_j, r_i, s_i))\big) & \text{if } c_i \geq 2. \end{cases} \tag{1}$$

Every aircraft, i, will have been heuristically allocated a traversal path, t_i, through the holding point prior to the feasibility check. We define a function $T(t)$ to return the minimum traversal time for an aircraft to traverse the holding point along path t and take off. Given a holding-point arrival time, h_i, for each aircraft, i, the earliest time at which the aircraft can reach the runway and take off is given by $h_i + T(t_i)$.

The earliest take-off time for an aircraft is limited by the physical taxi time to reach the runway, the necessity to maintain safe separations from preceding aircraft, and by the start of the CTOT slot, b_i, if one applies to the aircraft. Assuming that aircraft take off as early as possible, the take-off time for an aircraft, i, can be predicted using Equation (2):

$$d_i = \max(e_i, h_i + T(t_i), b_i). \tag{2}$$

In real life there is a variation in the time taken to traverse the holding point, line up on the runway, and become airborne, and in the time of the take-off clearance and the controller reaction. This variation is discussed below.

4.5. CTOT Constraints

Aircraft must adhere to their CTOT slots. The earliest take-off time, b_i, is implemented as a hard constraint in Equation (2) because it is always possible to delay an aircraft. At congested times, it is not always possible for an aircraft to meet the assigned CTOT slot, so the objective function must penalise this noncompliance to obtain schedules where as many aircraft as possible meet their CTOT.

Take-off times do not always match the prediction to the second. A natural variance is expected. As the CTOT is an absolute time, a divergence between the predicted take-off times and the real take-off times can have an effect on whether a CTOT is missed. The CTOT evaluation function therefore includes elements to penalise schedules that place aircraft near to either extreme of the CTOT slot; otherwise, if the schedule is shifted forward or backward due to the real-life variation in take-off times, aircraft could waste time waiting for the start of a CTOT, or be forced to miss the end of a CTOT slot. The CTOT evaluation function we use, $C(d_i, b_i, l_i, h_i)$, to measure the compliance of aircraft i, is given by Equation (3), where W_3, W_4, W_5, W_6, W_7, W_8, and W_9 are weights for the different parts of the objective and F_H, F_L, and F_B are constants explained below.

$$C(d_i, b_i, l_i, h_i) = \begin{cases} W_3((d_i - l_i)^{1.1}) + W_4 \quad \text{if } d_i \geq (l_i + 300), & \text{(3a)} \\ W_5((d_i - l_i)^{1.1}) + W_6 \\ \quad \text{if } (l_i + 300) > d_i > \max((h_i + F_H), l_i), & \text{(3b)} \\ W_7(d_i - l_i) + W_6 \quad \text{if } (h_i + F_H) \geq d_i > l_i, & \text{(3c)} \\ W_8(F_L + d_i - l_i) \quad \text{if } l_i \geq d_i > (l_i - F_L), & \text{(3d)} \\ 0 \quad \text{if } (l_i - F_L) \geq d_i > (b_i + F_B), & \text{(3e)} \\ W_9(b_i + F_B - d_i) \quad \text{if } (b_i + F_B) \geq d_i. & \text{(3f)} \end{cases}$$

Term (3b) usually applies to aircraft that are scheduled too late for their CTOT slots, but within a five-minute extension. The nonlinear form of this penalty helps to avoid the larger extensions, biasing towards equity of extensions where they are required—for example, slightly preferring two two-minute extensions over a one-minute and a three-minute extension. As the separation rules are in multiples of 60 seconds, moving the position of an aircraft in a schedule will usually alter the delay by a multiple of 60 seconds. The cost escalates as the delay increases, so aircraft with problematic CTOT times are not just shifted later in the schedule in favour of aircraft with less restrictive CTOTs. W_5 is sufficiently large to ensure that the primary objective is to reduce CTOT misses. The fixed penalty for a CTOT miss, W_6, is also large to ensure that as few as possible are missed.

If aircraft are more than five minutes late for their CTOT slot, then there is a problem because the CTOT slot will need to be rearranged. Term (3a) of Equation (3) ensures that any schedule for which the departure time of an aircraft is more than 300 seconds after the end of the CTOT slot will be heavily penalised. W_4 is the largest weight in the system and is significantly larger than all of the other weights. In addition, W_3 is significantly larger than W_5 so that the model will accept schedules where multiple aircraft miss their CTOT by up to five minutes in preference to a single aircraft missing by over five minutes.

Sometimes aircraft will be delayed at the stands or in taxiing around the runway, and their CTOT may be extremely tight so that meeting it becomes unrealistic. In these cases, penalising the deviation from CTOT as severely as in (3b) will result in a schedule where the aircraft is scheduled as early as possible, regardless of the effect this has on the rest of the schedule. To avoid this, a period of flexibility is introduced whereby, for a given number of seconds, F_H, after arrival at the holding point, any CTOT miss by the aircraft will be less severely penalised. W_7 is set to a relatively low value so that CTOT misses within this period soon after arrival at the holding point are penalised less.

Terms (3d), (3e), and (3f) relate to aircraft that are scheduled to depart correctly within their allocated CTOT. By determining a minimum number of seconds of flexibility, F_L, which should be allowed between any aircraft's scheduled take-off time and the end of its CTOT slot, a penalty can be assigned to ensure that there is some leeway to account for possible delays and ensure that aircraft still meet their CTOT slot, term (3d). Similarly, a minimum number of seconds of flexibility, F_B, can be assigned to aircraft scheduled at the start of their CTOT slot to ensure that the schedule can take account of opportunities for earlier take-off if they become available. The constants W_8 and W_9 are set to low values because moving aircraft further into their CTOT period is desirable, but not essential if it would mean increased delay in the schedule or another aircraft missing their CTOT slot. For any aircraft, i, without a CTOT, b_i and l_i were chosen to ensure that term (3e) applies and CTOT restrictions do not contribute to the cost.

4.6. The Objective Function

As described in §4.3 above, the objective to reduce delay will move problematic separations later in the schedule, where they are more likely to be utilised by aircraft entering the system later. However, some aircraft are almost always associated with larger separations, before or after them. For example, because there are so few light aircraft flying from Heathrow, light aircraft will invariably be forced to follow heavier-weight-category aircraft, therefore requiring a larger separation to be enforced, regardless of how long they are delayed. The function $P(c_i, a_i, v_i, s_i)$ exists to ensure that where the characteristics of an aircraft will always mean there is a larger separation before or after it, schedules that delay the aircraft are penalised. Delaying these aircraft merely constrains the holding-point reordering by congesting the holding point unnecessarily. The CTOT slots and holding-point structure, combined with the positional delay measures, also help avoid the undue penalising of certain aircraft.

The final objective function, Formula 3, is a weighted sum of the delay, reordering cost, CTOT cost as given by function $C(d_i, b_i, l_i, h_i)$, and the additional penalty cost, $P(i, a_i, v_i, s_i)$. The weights W_1 and W_2 are used to ensure that CTOT compliance is of primary concern, minimising delay is secondary, and avoiding reordering is tertiary:

$$\text{Minimise} \sum_{i=1}^{n}(W_1(d_i - h_i) + W_2(\max(0, c_i - a_i))^2 + C(d_i, b_i, l_i, h_i) + P(c_i, a_i, v_i, s_i)). \quad (3)$$

In this paper, we used $W_1 = 1$, $W_2 = 3$, $W_3 = 10{,}000$, $W_4 = 10{,}000{,}000$, $W_5 = 1$, $W_6 = 100{,}000$, $W_7 = 2{,}000$, $W_8 = 5$, $W_9 = 10$, $F_L = 120$, $F_B = 60$, and $F_H = 120$. The exact values used can be modified to change the relative priorities of the different objectives. The important factors are that W_3 and W_4 should be extremely high, and W_6 and W_7 should be very high compared with the other weights. By choosing a very low value for W_5 we are indicating that if an aircraft is arriving too late for its CTOT, it is not so important where in the extension it takes off because putting too much importance on this may overly restrict the rest of the schedule.

5. Results

To evaluate the effectiveness of the different search techniques, an automated testing system was used to simulate the operation of a live system. The aim of the real system is to get the best schedule for the entire data period, even though the system is only aware of a subset of the aircraft at any time. Section 3 presents a solution for the problem of scheduling a known set of aircraft. The objective function in the model in §4 is designed to ensure that scheduling the aircraft that are in the system at the time will not overly constrain the scheduling of aircraft that will arrive later.

The tests moved through the data set in 15-second intervals, from the start time until the last aircraft had been scheduled for take-off. At each step, aircraft were added if they had left their stands, and removed if they had taken off and could no longer affect the schedule. At each step, one search was performed to find a good order for take-off and the best schedule found was adopted, giving little opportunity for later searches to find schedules missed by earlier searches. A live system would be running continuously, re-solving the static problem every second as the situation changes and taking advantage of any reduced separations the controller may allow.

Aircraft can usually traverse the holding point in less than one minute, but a two-minute minimum traversal time was used to allow for some uncertainty in holding-point arrival time, delay in traversal time, and to allow extra time for the line up for take-off. It should be noted that this is a minimum traversal time and that, due to queueing time, most aircraft will actually take much longer to traverse the holding point. The heuristic holding-point traversal path assignment ensures that the aircraft with the longer paths through the holding point will be those that are overtaken within the holding point and therefore have more time available to traverse the holding point.

This test is deliberately overly restrictive in what can be performed—for example, in fixing the holding-point traversal path as soon as the aircraft reaches the holding point, only using the simpler paths and using a restrictive holding-point network model. The intention is to ensure that the tests do not find schedules that are prohibitively difficult to enact in practice. The early fixing of the holding-point traversal path and the restriction to only use good paths both limit the reordering that can be performed to that which is easy rather than that which is possible. In a live decision-support system with a controller present, more flexibility could be allowed and the schedules produced may be even better.

5.1. Test Data

Real, historic data, provided by National Air Traffic Services (www.nats.co.uk), was used for the testing. The test data covered six time periods, each for a single runway configuration. Table 1 gives information about the test sets and the holding-point configuration they were for. Test data included details of the weight class, SID route, speed group, and CTOT for

Table 1 Details of the Data Sets

Data set	Holding point	Aircraft	CTOTs	Light	Medium	Heavy
1	27R	329	172	0	235	94
2	27L	304	90	0	205	100
3	27L	341	158	0	245	96
4	27L	330	100	1	239	90
5	27R	345	98	1	244	100
6	27L	259	42	2	193	64

every aircraft with the times the aircraft left the stand, arrived at the holding point, and actually took off.

The holding-point configuration is important because it can greatly change the constraints upon reordering. There are three different categories of aircraft for the purpose of wake vortex departure separations: light, medium, and heavy. The table shows that there are very few light aircraft and that most aircraft are in the medium weight category.

5.2. Test Results

Tables 2 to 7 present the results of the tests that have been performed using each of the six data sets. Five sets of results are provided for each data set; these are explained below. In each case, we show the results when the schedule containing all of the aircraft in the data set is evaluated as a whole.

The CTOT column shows the total number of CTOT slots that were missed. The primary aim of the search process is to ensure CTOT compliance. The delay column gives the total holding-point delay, in seconds, suffered by the aircraft in the schedule. The cost column shows the cost of the final schedule, calculated using the model presented earlier. This is not very meaningful for the manual schedules, but it gives an idea of the worth the model attributes to each of the automatically generated schedules.

The results for the real, manually produced schedule are given for two methods of evaluation. The "real order, actual times" results are for the real take-off order, using the real take-off times. The "real order, predicted times" are the results for the real take-off order, using the model to predict take-off times for the aircraft. The "first-come-first-served" (FCFS) schedule is produced by assuming that no reordering is performed within the holding point. Aircraft are assumed to take off in the order they arrived at the holding point, and the model is used to predict the take-off times.

Table 2 Results for Data Set 1

Search	CTOTs missed	Total cost	Total delay
Real order, actual times	10	13,823,992	99,805
Real order, predicted times	13	26,950,174	110,248
FCFS order	94	1,697,393,350	413,350
Tabu search order	4 (0)	452,017 (6)	83,913 (3)
SAES order	4	453,261	85,079

Table 3 Results for Data Set 2

Search	CTOTs missed	Total cost	Total delay
Real order, actual times	9	12,025,641	127,891
Real order, predicted times	9	11,171,470	121,734
FCFS order	68	1,607,102,025	671,926
Tabu search	4 (0)	510,041 (19)	88,019 (28)
SAES order	4	511,183	89,239

Table 4 Results for Data Set 3

Search	CTOTs missed	Total cost	Total delay
Real order, actual times	6	82,627,619	117,894
Real order, predicted times	11	89,398,804	121,037
FCFS order	94	1,477,545,168	421,168
Tabu search order	3 (0)	56,085,290 (144)	93,508 (121)
SAES order	4	57,444,798	95,248

Table 5 Results for Data Set 4

Search	CTOTs missed	Total cost	Total delay
Real order, actual times	5	9,560,770	120,893
Real order, predicted times	13	22,876,359	124,510
FCFS order	55	1,435,684,737	602,737
Tabu search order	3 (0)	94,736 (24)	92,719 (29)
SAES order	3	98,553	96,609

Because the tabu search has a stochastic element, it was performed 20 times for each data set. The mean result is presented in the table, and the standard deviation is given in parentheses. Finally, within the time allowed to the tabu search algorithm, an exhaustive reordering of the first seven aircraft in the schedule would be possible. We call this the *seven-aircraft exhaustive search* (SAES), and the results of doing this are presented in the tables and discussed below.

5.3. Real and First-Come-First-Served Schedules

The accuracy of the take-off time predictions can be evaluated by comparing the real take-off times against the predicted take-off times for each schedule. The predictions for the take-off times have been observed to be slightly pessimistic at busy times and slightly optimistic at quiet times, so that the predicted schedule will lag slightly behind the actual schedule at busy times and lead it at quiet times.

The predicted times work on the assumption that it is imperative for aircraft to take off as early as

Table 6 Results for Data Set 5

Search	CTOTs missed	Total cost	Total delay
Real order, actual times	5	2,558,965	107,786
Real order, predicted times	5	2,537,579	112,400
FCFS order	53	1,266,005,125	647,125
Tabu search order	1 (0)	84,225 (3)	82,958 (16)
SAES order	1	89,450	88,097

Table 7 Results for Data Set 6

Search	CTOTs missed	Total cost	Total delay
Real order, actual times	4	33,617,393	96,235
Real order, predicted times	8	33,120,710	131,261
FCFS order	33	1,019,792,389	572,389
Tabu search order	4 (0)	26,575,020 (26)	73,581 (47)
SAES order	4	26,581,742	80,160

possible. The accuracy of these predictions is very good at busy times when there are a number of aircraft waiting for take-off. At quiet times, the system acts slightly differently to the model because the time between the aircraft entering the holding point and taking off becomes increasingly unpredictable when there is less pressure to keep the throughput high.

Conversely, in some cases controllers can safely reduce certain separations, provided the aircraft involved have visual contact. In these cases, aircraft can sometimes take off earlier than predicted, reducing the delay for any aircraft following it. In some cases, smaller aircraft can enter the runway at intermediate points rather than at the end of the runway via the holding point. Our generated schedules do not allow this at present, but when this was actually done in the real schedule it would mean that the time the aircraft reached the runway, and the consequent take-off time, could be earlier than predicted by the model.

A real system would have constant feedback from the real-world situation to correct for slight deviations between the predicted and actual take-off times. This feedback is absent from our automated tests, so the cumulative discrepancies often increase through the busy times until they realign at periods of relative quiet. Importantly, the model will usually overestimate the delay in a schedule. The times when it does not do so are the quieter times, i.e., those times when there is no necessity to do so. We can have confidence, therefore, that the results using the predicted take-off times are pessimistic rather than optimistic, and that a real system would be able to perform at least as well as the simulations predict.

The FCFS results are provided for comparison with the manually produced schedules to see the effects of not having a controller performing the reordering. It is obvious from the results in the table that the controllers perform an important job both in reducing delay at the holding point and in complying with CTOTs. If the controller did not reorder the aircraft, a very large increase in the holding-point delay for the aircraft in the schedule would occur. Cumulative delays mean that a large number of CTOT misses should be expected for this schedule.

A value of only 30 seconds was used for the minimum holding-point traversal time for the evaluation of the manual schedule. This is closer to the times that are seen in practice and gives better predictions of the take-off times. When evaluating automatically generated schedules, it is important to ensure that the schedule is not only achievable, but also easy to achieve, so a two-minute minimum traversal time is used to ensure this. When examining the real take-off schedule, the feasibility is already known, so the lower, more realistic traversal time can be used. We will show later that the increased minimum traversal time makes the predicted delay higher, not lower, and so is not the reason for the good results obtained.

5.4. Automatically Generated Schedules

The tabu search always performed better than the seven-aircraft exhaustive search (SAES). In one case, it even obtained a better CTOT compliance than the SAES search, while simultaneously reducing the delay for aircraft. The only advantage that the tabu search has over SAES is that it can consider more than just the first seven aircraft. We therefore conclude that this added flexibility, being able to reorder later aircraft, shows that the metaheuristic search is worthwhile.

Both the tabu search and SAES performed better than the real controllers in all cases, even though the automated searches are more constrained in what they can do in a holding point than the real controllers. Due to the heuristic allocation of paths through the holding point, only good paths are ever used. Furthermore, the network model of the holding point is deliberately overly restrictive to ensure that schedules adopted are easy to achieve. Some of the reordering that is currently rejected may be acceptable to some controllers in some circumstances. Therefore, it is envisaged that even better results could be obtained when combining the system with a real controller.

Any decision-support system should allow input from the controllers—for instance, to use reduced separation rules. A live decision-support system should also be able to take into account the real times of take-off and adjust the schedule forwards or backwards to take account of these.

The schedules produced by the tabu search were evaluated using a two-minute minimum holding-point traversal time. Table 8 shows the results for a minimum traversal time of one minute. Again, the mean number of CTOTs missed and the total delay for aircraft is given, with the standard deviation in parentheses.

It is obvious from Table 8 that reducing the minimum holding-point traversal time to 60 seconds can considerably decrease the total delay at the holding point. The main reason for the decrease in the delay is due to new schedules becoming available. In some cases, it is possible to fit an aircraft into an earlier gap

Table 8 **Tabu Search, 60-Second Minimum Traversal Time**

Data set	CTOTs missed	Seconds delay
1	2 (0)	68,248 (164)
2	4 (0)	73,491 (66)
3	3 (0)	78,520 (93)
4	3 (0)	75,313 (110)
5	1 (0)	66,704 (13)
6	3 (0)	59,388 (28)

in the schedule, with a corresponding effect upon the departure times of a number of aircraft.

In practice, it is wise, in general, to assume a two-minute minimum traversal time to allow for delays, enable the schedule to slip forward without rescheduling aircraft, make it easy for an aircraft to reach the runway in time, and allow some leeway for errors in predicted holding-point arrival times for aircraft not yet at the holding point. Examination of where the one-minute and two-minute schedules differ implies that it would be wise, in a live implementation of the system, to allow the controller to specify a lower minimum traversal time for some aircraft—for instance, those with tight CTOTs.

5.4.1. The Performance of the Controllers. Runway controllers generally have time only to consider the aircraft actually in the holding point. We performed the SAES considering only the aircraft in the holding point rather than also including those on the taxiways, and the results were much more comparable to those obtained by the controllers.

To control their workload, controllers give conditional clearances to aircraft, giving pilots early information about which aircraft they will follow for take-off and, hence, fixing the take-off order for those aircraft. We performed the SAES, assuming that the next six minutes of departures, roughly six aircraft at busy times, were fixed. In this case, due to the overly constrained nature of the rest of the model, the search failed to create schedules that were as good as those that the controllers created.

The results show that the controllers are performing very well, given the time constraints and workload conditions under which they are working. The results also show that the gain that the controllers would see from a decision-support system is due to the advance notice of future aircraft that such a system could give.

5.4.2. Workload and Dissatisfaction. Controlling the amount of work for the controller and pilot was key, as was avoiding undue dissatisfaction. We control workload by allowing only good holding-point traversal paths to be used and by allowing any aircraft at least two minutes to reach the runway. To evaluate the equity of delay, we examined the amount of positional delay and overtaking that takes place in the generated schedules.

The maximum delay of the most delayed aircraft is slightly higher in the automatically generated schedules. However, in all cases these aircraft were delayed to achieve their CTOT slots. Where aircraft were delayed to wait for the start of their CTOT slots, controllers usually let the aircraft take off as soon as possible. The automatic searches, however, favoured scheduling the aircraft further into the take-off slot because the objective function allows for slight movement of the schedule forward over time.

The automatically generated schedules positionally delayed fewer aircraft. Additionally, the total number of places by which aircraft were delayed was lower for the automatically generated schedules. The improved results for the generated schedules were therefore due to identifying the correct aircraft to delay rather than increasing the reordering. The automatically generated schedules produce less positional delay on average. However, they penalise specific aircraft more than the real controllers do. This suggests that real controllers value equity of delay more highly than the objective function used by the search algorithms does.

The total number of aircraft overtaking is higher for the automatically generated schedule, but each overtakes fewer aircraft than in the manual schedule. This is not surprising because the overly restrictive holding-point model ensures that only a limited number of aircraft can be overtaken at any time.

5.4.3. Schedule Evolution. Over the duration of each test, the simulation generates a number of static problems for the decision-support system to solve. Examining the solutions to each of the static problems is useful in understanding exactly what decision support would have been given to controllers and how the suggestions changed over time. In particular, it is important that at least the early part of the schedule is stable over time, rather than constantly fluctuating between wildly different schedules of similar cost. The early part of the schedule is important because these are usually the aircraft in the holding point, under the control of the runway controller, so they are the only aircraft that would have been given instructions.

During the execution of the tests, we observe that the schedule does not usually change until a new aircraft enters the system, at which point it changes to accommodate the new aircraft. In the majority of cases, new aircraft leaving their stands do not overtake more than a couple of aircraft to reach a good take-off position, and these overtaken aircraft are usually those still on the taxiways, so most of the time only the last few aircraft in the schedule are affected.

The most excessive change to the schedule is observed when aircraft with tight CTOTs leave their stands. At this point, the system will try to schedule them within CTOT if possible, often requiring overtaking a number of other aircraft to do so. This is the case that is seen in Data Set 3, where the tabu search improves on the CTOT compliance of the SAES search. In this case, the aircraft needed to overtake more than six other aircraft to hit the CTOT.

Even in the case of a tight CTOT, the constraints that we imposed upon rescheduling within the holding point and upon late rescheduling act to limit changes. For instance, as the first two minutes of the schedule are fixed, no new aircraft can overtake these aircraft. Similarly, if an aircraft in the holding point was not already being overtaken, then it will have been assigned a direct path through the holding point. In this case, there is no way for a later arrival at the same holding-point entrance to overtake it.

The relative stability of the early part of the schedule is not unexpected because the delay-based objective function penalises larger separations much more if they are nearer the start of the schedule. The objective function, therefore, puts much more value on getting the first part of the schedule right. This means that the search is much more likely to find solutions that schedule the first few aircraft well than those that schedule the last few aircraft well. Additionally, the fact that each subsequent search is seeded with the resulting schedule from the previous iteration helps to ensure that the best schedule found in the previous search will be rediscovered in this search, so if there is a good schedule that is similar to the previous iteration, it is more likely to be found than a good schedule that is very different.

6. Conclusions

The job of the runway controller is complex, and involves a high workload. With all of the communication involved in the job, there are limits to how much consideration a controller can give to the reordering of the aircraft at the holding points. The complexity of the reordering task, the physical constraints of the holding point, and the limited time that controllers have available mean that they have an extraordinarily difficult task to perform. The successful development of a decision-support system could alleviate this difficulty to some extent.

The presentation of the take-off scheduling problem in the academic literature usually fails to take account of the physical constraints on the scheduling problem at airports such as Heathrow. Much of the previous research has been concerned with the arrivals problem, but there are considerable differences between the arrival and departure problems. Where research has looked specifically at the departure problem it is usually assumed that reordering can take place easily, and so assumes that the physical holding-point structures are not relevant to the problem. However, at London Heathrow the most practical place to reorder departures is within the holding points at the end of the runway.

This paper presented a hybrid metaheuristic and a heuristic approach to this reordering of aircraft that can take account of the physical holding-point structure, presents only good orders of take-off to the controller, and does so very quickly. Real-world constraints, such as partially fixed schedules and fixed routes through the holding points, have been accounted for in addition to the usual constraints, such as maintaining required separations and meeting calculated time of take-off slots.

Because this is a real-world problem, the objective function is not simple, but is key to obtaining relevant results. It has components to avoid overconstraining future searches, as well as to obtain the best delay and CTOT compliance at the time of the search. The desirable schedules include a degree of leeway to allow for natural movement of a schedule forward or backward in response to the real-world situation. The presented results show that, even though at each iteration of the test the system has knowledge of only a subset of the aircraft, good overall schedules can be achieved.

A decision-support system can consider many more aircraft than a human controller can and hence help to avoid future problems from aircraft not currently under the control of the runway controller. The presented results show that if the controller had more information about the aircraft taxiing around the runway, the CTOT compliance and delays at the holding point could be improved. We have also shown that the controllers do a very good job, given the information they currently have and that any improvement would come as a result of the increased knowledge of aircraft taxiing toward the holding points. Even though a human controller could not reasonably be expected to handle the increased amount of data, we have presented a hybrid metaheuristic approach that can.

If a computerised system is to be of any use to the runway controller in ordering the aircraft at the holding points, then it must be able to find good take-off schedules in real time. The system should only present schedules that are easy to achieve, rather than requiring complex manoeuvring and complicated instructions. Our decomposition of the problem with heuristic allocation of holding-point paths, ensures that any reordering can be easily achieved and that the aircraft that have longer routes through the holding point are the ones that are overtaken and hence have more time in which to traverse it. Analysis of the results shows that the automated searches find schedules that do not involve either excessive positional delay or reordering when compared with the manually produced schedules.

The evaluation of our system was deliberately overly restrictive to ensure that the results for the automated searches were pessimistic. With a real controller, we would expect the system to perform better than these results imply.

First, we have shown that a decision-support system could help at Heathrow because the increased search power could improve the holding-point delay and CTOT compliance. Second, we have shown that with the presented decomposition of the problem schedules can be found fast enough for use in a real-time system. Having developed a methodology that has produced successful results on real-world data, we have a number of future research directions that we would like to explore.

We will extend our work by explicitly considering the effects of the uncertainty implicit in the system—for instance, in the taxi times and traversal times. We aim to develop the system to cope with uncertainty—for example, by enabling fail-over schedules and producing schedules that are less sensitive to uncertainty. Successfully achieving these goals should allow us to improve upon the current results.

We also plan to improve the simulation part to predict holding-point positions of aircraft and update them as the simulation progresses. We then aim to use the simulation to evaluate the effects of other influences such as holding-point structure changes or decisions by the ground movement planner, as well as to evaluate how well the system can cope with unforeseen circumstances—for example, a situation in which the next aircraft planned to take off is suddenly unable to do so.

Finally, we intend to look at push-back times and stand holding. Push-back times are defined to be the time when aircraft push back from the stands onto the taxiways. At the moment, due to the large degree of uncertainty involved in predicting push-back times, we do not even include knowledge of aircraft at the stands, never mind predicting push-back times for them. The scheduling is currently performed at the holding point, as that is where the greatest amount of knowledge is available with the least amount of uncertainty. The inclusion of the taxiing aircraft provides sufficient information about the future aircraft that will need to be scheduled to ensure that good schedules can be obtained by reordering aircraft in the holding point. However, scheduling at the stands, if the difficulties with doing so can be overcome, facilitates the possibility of holding aircraft at the stands for longer, without the engines running, rather than queueing at the holding point. So even if the total take-off times are no earlier, the fuel burn should be significantly less.

Acknowledgments

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

References

Abela, J., D. Abramson, M. Krishnamoorthy, A. de Silva, G. Mills. 1993. Computing optimal schedules for landing aircraft. *Twelfth National Conf. Australian Soc. Oper. Res.*, Adelaide, Australia.

Anagnostakis, I., J.-P. Clarke. 2002. Runway operations planning, a two-stage heuristic algorithm. *AIAA Aircraft, Tech., Integration Oper. Forum*, Los Angeles, CA.

Anagnostakis, I., J.-P. Clarke. 2003. Runway operations planning: A two-stage solution methodology. *Thirty Sixth Hawaii Internat. Conf. System Sciences* (*HICSS-36*), HI.

Anagnostakis, I., J.-P. Clarke, D. Böhme, U. Völckers. 2001. Runway operations planning and control, sequencing and scheduling. *Thirty Fourth Hawaii Internat. Conf. System Sciences* (*HICSS-34*), Maui, Hawaii.

Anagnostakis, I., H. R. Idris, J.-P. Clarke, E. Feron, R. J. Hansman, A. R. Odoni, W. D. Hall. 2000. Computing optimal schedules for landing aircraft. *Third FAA/Eurocontrol Internat. Air Traffic Management R&D Seminar* (*ATM2000*), Naples, Italy.

Atkin, J. A. D., E. K. Burke, J. S. Greenwood, D. Reeson. 2004. A meta-heuristic approach to departure scheduling at London Heathrow Airport. *Ninth Internat. Conf. Comput.-Aided Scheduling Public Transport* (*CASPT2004*), San Diego, CA.

Beasley, J. E., J. Sonander, P. Havelock. 2001. Scheduling aircraft landings at London Heathrow using a population heuristic. *J. Oper. Res. Soc.* **52** 483–493.

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

Bianco, L., P. Dell'Olma, S. Giordani. 1999. Minimizing total completion time subject to release dates and sequence-dependent processing times. *Ann. Oper. Res.* **86** 393–415.

Burke, E. K., G. Kendall, eds. 2005. *Search Methodologies.* Springer, New York.

Craig, A., R. Ketzscer, R. A. Leese, S. D. Noble, K. Parrott, J. Preater, R. E. Wilson, D. A. Wood. 2001. The sequencing of aircraft departures. *Fortieth Eur. Study Group with Indust.*, Keele, UK.

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

Gendreau, M., J.-Y. Potvin. 2005. Tabu search. E. K. Burke, G. Kendall, eds. *Search Methodologies*, Ch. 6. Springer, New York, 165–186.

Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13** 533–549.

Glover, F., G. Kochenberger, eds. 2003. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston/Dordrecht/London.

Glover, F., M. Laguna. 1997. *Tabu Search*. Kluwer Academic Publishers, Boston/Dordrecht/London.

Idris, H. R., J.-P. Clarke, R. Bhuva, K. Kang. 2002. Queuing model for taxi-out time estimation. *Air Traffic Control Quart.* **10** 1–21.

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

Idris, H. R., B. Delcaire, I. Anagnostakis, W. D. Hall, N. Pujet, E. Feron, R. J. Hansman, J.-P. Clarke, A. R. Odoni. 1999. Observations of departure processes at Logan Airport to support the development of departure planning tools. *Air Traffic Control Quart.* **7** 229–257.

Newell, G. F. 1979. Airport capacity and delays. *Transportation Sci.* **13** 201–241.

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

van Leeuwen, P., H. Hesselink, J. Rohling. 2002. Scheduling aircraft using constraint satisfaction. *Electronic Notes in Theoret. Comput. Sci.* **76**.