Ravizza, Stefan and Atkin, Jason and Burke, Edmund K. (2014) A more realistic approach for airport ground movement optimisation with stand holding. Journal of Scheduling, 17 (5). pp. 507-520. ISSN 1094-6136

Journal of Scheduling manuscript No.
(will be inserted by the editor)

A more realistic approach for airport ground movement optimisation with stand holding

Stefan Ravizza · Jason A. D. Atkin · Edmund K. Burke

Received: 30 October 2011 / Accepted: date

Abstract Despite the requirements to handle ever increasing numbers of aircraft, airports also have to meet environmental targets and regulations. The complexity of the problems increases the closer an airport has to work to its maximal possible capacity. The complexity of the problems also mean that advanced decision support systems are needed to guarantee efficient airside airport operations and to mitigate the environmental impact. This research considers the important problem of getting aircraft from source to destination locations (usually either runways or gates/stands) in as efficient a manner as possible, in terms of time or fuel burn. A new sequential graph-based algorithm is introduced for this important part of the airside operations at an airport, which is usually named the ground movement problem. This algorithm, embedded in a wider operational system, has several advantages over previous approaches in terms of increasing the realism of the modelling and it also utilises a recently developed approach to more accurately estimate taxi times. The algorithm has been configured to absorb as much of the waiting time as possible for departures at the gate/stand, to reduce the fuel burn and the environmental impact. Analysis with data from a European hub airport shows very promising results and gives an indication of both the performance of the system (in comparison to a lower bound on the taxi time) and the limits to the amount of waiting time which could possibly be absorbed as stand hold (without the engines running).

Keywords Ground movement optimisation · Airport operations · Routing · Real world scheduling · Decision support system

Stefan Ravizza
School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK
E-mail: smr@cs.nott.ac.uk

Jason A. D. Atkin
School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK
E-mail: jaa@cs.nott.ac.uk

Edmund K. Burke
Department of Computing and Mathematics, University of Stirling, Cottrell Building, Stirling, FK9 4LA, UK
E-mail: e.k.burke@stir.ac.uk

1 Introduction

European airports face several challenges in the 21st century, including the capacity challenge (with demands for air travel still increasing year on year) and the environmental challenge (ACI EUROPE 2010). To avoid forming huge bottlenecks in the air transportation system, airports have to either be enlarged, or (since enlargement is either not possible or prohibitively expensive in most cases), to utilise the existing resources as efficiently as possible. In addition, the increasing focus upon environmental issues is also likely to further grow over time. As airports work closer to their maximum capacity, airside airport operations become much harder to deal with. As a result, decision support systems have to be increasingly advanced and need to both integrate different airside airport operations with each other and to model each process increasingly realistically.

From an optimisation point of view, ground movement of aircraft can be considered to be one of the most important airside operations at an airport, since it links several other problems together, such as the runway sequencing problems for arrivals and/or departures (Atkin et al. 2007), the stand holding problem (Atkin et al. 2011a) and the gate assignment problem (Dorndorf et al. 2007). For a comprehensive literature review

of ground movement research and the integration with other operations, we point the interested reader to a recently published review (Atkin et al. 2010b).

This paper presents a decision support framework for environmentally friendly ground movement, along with promising experimental results which utilise more realistic taxi time predictions for a European hub airport. A framework is described for integrating a graph-based sequential movement algorithm into a larger decision support system which can also consider the runway sequencing problem and the stand holding problem. A Fuzzy Rule-Based System (FRBS) has been used to more accurately estimate taxi and pushback times for aircraft than a standard lookup table may allow. This utilises the same graph which is used for the ground movement model. This integrated approach allows the effects of ground plan changes to be modelled more accurately, changing both taxi time predictions and routing information. In addition, several concepts have been included in the model which allow airport layouts to be modelled in a more realistic manner, such as restricting certain taxiways to be used only by certain aircraft and coping with the required separations between aircraft. Finally, the absorption of delay at the stand, prior to starting the engines, has been considered. This reduces the waiting times at the runway and is further extending previous stand holding ideas (Atkin et al. 2010a, 2011a; Burgain et al. 2009). The maximal potential benefits of such a system have been quantified.

Section 2 provides a description of the airport ground movement problem and how it can be embedded into the larger combined sequencing/routing/stand holding framework. Details of the dataset which were provided by the airport are then presented in Section 3 together with the method for estimating taxi times. Following this, the sequential ground movement algorithm which has been developed, and was utilised for these experiments, is detailed in Section 4. The results of the application of the algorithm to the dataset are then shown in Section 5; before the paper ends with some conclusions in Section 6.

2 Problem description

The links between the ground movement problem and runway sequencing are considered first in this section, before the ground movement problem itself is discussed in more detail. The section ends with a consideration of the stand holding benefits which can result from the appropriate solution of the ground movement problem.

2.1 The links with runway sequencing

Atkin et al. (2010b) highlighted the importance of integrating the ground movement problem with other airside airport operations, such as the problems of finding good departure and arrival sequences. Supporting controllers in these tasks is a challenge, especially when departures and arrivals have common restrictions and interactions due to the airport layout. For this paper, we assume that the runway sequencing and ground movement problems are solved as two distinct stages. The integrated (departures and arrivals) runway sequencing problem is assumed to be solved in a first stage, then the consequent landing and take-off times are used in the second stage, within the consideration of the ground movement problem. Thus, the wheels-on time at the runway (for arrivals) and the wheel-off time at the runway (for departures) are both assumed to be fixed within the ground movement problem. Issues such as conformance with take-off time slots are assumed to be taken into account by the runway sequencing stage. This decomposition has been found to be effective, but further research will analyse the benefits of providing a feedback loop from the ground movement problem to the integrated runway sequencing problem and of closer integration between the two problems.

2.2 Ground movement problem

The ground movement problem at an airport is a combined routing and scheduling problem. It involves guiding aircraft on the surface of an airport to their destinations in a timely manner, where the goal is to reduce the overall travel time and to enable the target take-off times at the runway to be met. It is important, for reasons of safety that two aircraft never conflict with each other throughout the ground movement process.

In the model which is considered in this paper, the route of the aircraft is not pre-determined (see Figure 2), allowing greater flexibility for solutions; however, the utilised solution method provides the possibility to restrict certain aircraft to specific taxiways and/or to avoid routes which involve tight turns. The airport layout is represented as a directed graph as can be seen in Figure 2, where the edges represent the taxiways and the vertices represent the junctions or intermediate points. Aircraft are considered to occupy edges, and conflicts are avoided by preventing any two aircraft from using the same edge simultaneously.

The times at the runway are assumed to be fixed for departures as well as for arrivals. The sequential approach to ground movement will then minimise the taxi time for each individual aircraft given the planned

movement for the aircraft which have already been routed. Hence, the approach will attempt to absorb as much of the waiting time as possible at the gate/stand, allowing the departures to start their engines as late as possible, reducing fuel burn and environmental impact. Thus, the solution method can be considered to be not only reducing the ground movement time, but also solving the stand holding problem (Atkin et al. 2010a, 2011a; Burgain et al. 2009) for a given runway sequence.

3 Analysed case: Zurich Airport

This analysis utilised data from Zurich Airport (ZRH), which is the largest airport in Switzerland and a hub airport for Swiss International Air Lines. The considered data included information about the airport layout, the positions of stands, runway entrance and exit points, and the layout of all of the taxiways. It also included the real timings for the aircraft using the airport during each day. This information was used to develop a taxi time prediction function, as discussed below, to improve the accuracy of the taxi time predictions which are used in the ground movement model. We had access to data for an entire week's operations between the 27th of June and the 3rd of July 2011. No extraordinary occurrences took place and there were 5613 movements in total (2806 arrivals and 2807 departures).

3.1 Physical layout

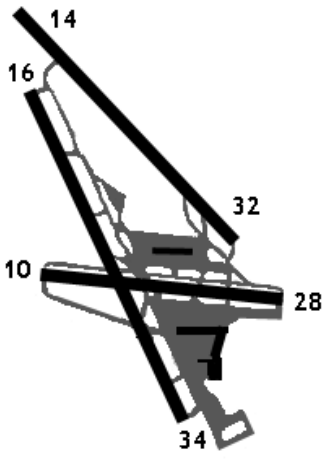

Fig. 1 Sketch of Zurich Airport (ZRH)

The airport has three runways, named 10/28, 14/32 and 16/34, according to their direction of operation, with two runways intersecting each other (see Figure 1). A directed graph model of the airport was developed to represent the airport's layout in Zurich. The full graph has 465 vertices, 553 edges and 119 gates. Figure 2 illustrates a part of the graph. The shortest route between the exit of runway 14 and a gate at pier A is highlighted in blue in this example and some alternative routes are shown in red. When parts of the shortest taxi path are blocked by other aircraft it can be beneficial for an aircraft to use an alternative route, which may be longer but have a shorter taxi time by avoiding the delays.

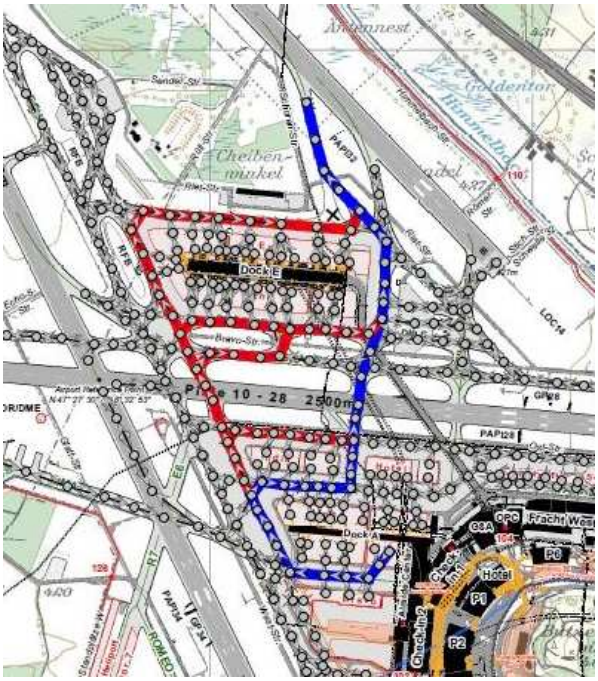

Fig. 2 Different routes from the exit of runway 14 to pier A

3.2 Taxi time prediction

Ground movement models need accurate taxi time predictions, but sufficiently accurate values are rarely available. In previous ground movement research, average taxi speeds have been assumed to be influenced only by the aircraft type, however, taxi time research has shown that the aircraft type is not the most influencing factor upon taxi speeds (Ravizza et al. 2012; Idris et al. 2002; Rappaport et al. 2009). Comparisons between ground movement tool results and the status quo at airports have previously been hard to analyse, due to the need for accurate taxi speed data. The historic data which has to be used usually includes the effects of any delays or re-routing due to conflicts between aircraft, so the effects of taxi time variability and the benefits from the

ground movement decision support system were often intermingled. This research confronts that challenge.

An approach to more accurately predict taxi times for aircraft or, equivalently, their average speeds, was proposed in Ravizza et al. (2012). Multiple linear regression was used to estimate a more accurate taxi speed prediction function and identify the factors which were most related to taxi speeds, such as total distance travelled, total turning angle and the number of other aircraft of different types which were moving around the airport at the time. The aim was to be able to eliminate the effects of factors which represented the actual amount of traffic at the airport (by zeroing the factors related to airport load), with the goal being to predict the taxi times for unimpeded aircraft. These predictions could then be used in a more advanced ground movement decision support system, such as the one described in this paper, which would itself model the effects of the interaction between aircraft (so these should not already be included in the taxi speed data). Chen et al. (2011) recently introduced an alternative fuzzy rule-based system (FRBS) approach to estimate taxi times at airports, using the factors which were identified in Ravizza et al. (2012). Results from the FRBS were found to outperform the multiple linear regression approach when applied to the same airport, thus it was adopted and extended for this research.

It was observed for Zurich that some aircraft have to push back from their allocated gates, taking additional time to do so, whereas other gates allow aircraft to immediately start their engines. The work by Ravizza et al. (2012) was extended to include a pushback duration and the multiple linear regression approach indicated that this factor was significant for Zurich. The resulting taxi time prediction functions by Chen et al. (2011) were therefore further enhanced for this work adding a predicted pushback duration to the taxi time for the first edge for departures where the gate requires it, before being utilised to predict the taxi times.

Finally, depending upon the terminal and the operating mode (which runways are in use), runway crossings may be necessary during the taxi process. For the moment these are included only in the prediction model for taxi times (having influenced the historic data), but we plan to integrate these effects into the combined ground movement and sequencing model later.

4 Ground movement decision support system

Figure 3 provides an overview flowchart describing the ground movement algorithm. Further details are provided later. The aircraft are routed sequentially in this approach. When an aircraft is ready, it has to be routed respecting all previous reservations by other aircraft using the taxiways. The routes which have been previously calculated for other aircraft do not normally change as new aircraft are taken into consideration (the exceptions are discussed in Section 4.9). This has advantages for the dynamic case, where some aircraft will have prior instructions, and acknowledges the difficulty and time costs associated with communicating changes to pilots and reducing the quantity of communication needed between the surface controllers and pilots. The objective for each of the sequential routings is to find the routing with minimal taxi time among all remaining conflict-free routings.

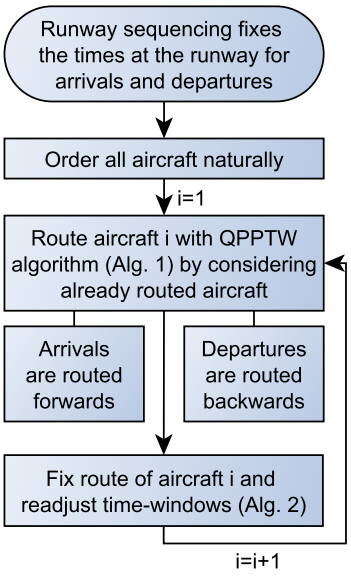

Fig. 3 Flow chart of general concept of the approach

The approach described here is based on research by Gawrilow et al. (2008) and the PhD thesis of Stenzel (2008). Ravizza modified the approach for his Master's dissertation (Ravizza 2009) to label the vertices instead of the edges, to simplify their interpretation. The original aim of this approach was to control automated guided vehicles in container terminals in harbours or in storage areas, but it is here applied instead to routing aircraft. The approach has been further modified for this work, for instance by allowing the approach to work backwards to meet a specified end time rather than starting time. The resulting algorithm is described in this section.

The Quickest Path Problem with Time Windows (QPPTW) algorithm is a generalized vertex-based label-setting algorithm based on Dijkstra's algorithm and can sequentially route aircraft on the airport surface, using

a directed graph model of the airport (see Figure 2). No time discretisation is used in this approach, in contrast to many other ground movement support systems (Balakrishnan and Jung 2007; Marín 2006; Marín and Codina 2008; Roling and Visser 2008). It has similarities to the recently published work by Lesire (2010), which used a sequential A* algorithm, but it provides a better coverage of the solution space, potentially allowing it to find better solutions within comparable execution times - these being short enough for it to be appropriate for real-time decision making. It also provides the possibility to define which edges in the graph are in conflict with each other and cannot be used simultaneously. In addition, for each edge incident to a vertex, the set of valid outgoing edges can be manually defined if desired, or can depend upon information about the aircraft. This enables the decision support system to forbid aircraft from making tight turns or to prevent aircraft from using taxiways for which they are too large. Together, these features enable the approach to more realistically model the airport surface while leaving the routing task itself to the algorithm.

The preprocessing of the algorithm is explained in Section 4.1, then the key concepts are introduced. The QPPTW algorithm is detailed next and the section ends with a discussion about buffer times and the sequence in which aircraft are routed.

4.1 Ground plan preprocessing

It is important to maintain separations between aircraft on the ground. The concept of conflicting edges is introduced here for this reason, so that no two conflicting edges can be occupied simultaneously. The conflicting edges are determined in a preprocessing stage. For this research, we used an approach which assumes straight connecting lines between vertices, since this requires less time in the preprocessing stage and is adequate for the directed graph model which has been used in this research, where the paths are almost straight lines between vertices. Edges in the graph, together with their embedding in the airport plan, are here named segments. In this approach, two segments conflict with each other if they are located closer together than a given threshold distance. To find the minimal Euclidian distance between two segments, the algorithm performs two processing steps. Firstly, it verifies whether the edges are intersecting, then, if they are disjoint, the distance between each end point of one segment and the closest point on the other segment is calculated. The minimum over these four distances corresponds to the minimal distance between the two segments.

4.2 Variable definitions

Definitions of the variables and data structures which are used in the model are given in Table 1.

4.3 Key concepts

The QPPTW algorithm with its expansion steps works in a similar way to Dijkstra's algorithm (Cormen et al. 2001; Dijkstra 1959), however, a label can be expanded several times due to the different time-windows and an additional concept of dominance is needed in order to guarantee a polynomial solution time. It is necessary to define some of the concepts upon which the approach is based. Firstly, the algorithm needs information about the times that each part of the taxiway (edge) is free:

Definition: Set of sorted time-windows
The set $\mathcal{F}(e)$ contains the sorted set of time intervals $F_e^j = [a_e^j, b_e^j]$ which specify the times when the edge e can be used for a new route. This will exclude the times when e, or an edge which conflicts with e, are in use by previously routed aircraft. These are inputs to the routing algorithm for each aircraft.

The use of labels is the essential concept of the QPPTW algorithm:

Definition: Label
A label $L = (v_L, I_L, pred_L)$ *specifies the time period $I_L = [a_L, b_L]$ within which the current aircraft could reach vertex v_L. It includes a reference to the previous label on the route, $pred_L$, and thus implicitly represents a route (with edge traversal timings) from a source vertex to the specified vertex v_L. These labels are generated as the routing algorithm progresses, together specifying the (undominated) time periods (from time a_L to time b_L) when the current aircraft could reach vertex v_L.*

An ordering relation is defined over the intervals of the labels to allow the definitions of dominance:

Definition: Dominance
A label $L = (v_L, I_L, pred_L)$ dominates *a label* $L' = (v_{L'}, I_{L'}, pred_{L'})$ *on vertex $v_L = v_{L'}$ if and only if $I_{L'} \subseteq I_L$ (and there are identical route restrictions on the outgoing edges), which implies $a_L \leq a_{L'}$ and $b_L \geq b_{L'}$.*

Once the routing has been performed by the QPPTW algorithm, the time-windows are readjusted (as discussed in Section 4.6) before the QPPTW algorithm is reapplied to route the next aircraft.

Table 1 Table of definitions

Variable	Explanation
$confl(e)$	The set of edges which conflicts with edge $e \in E$
$F_e^j = [a_e^j, b_e^j]$	jth time-window on edge $e \in E$, from time a_e^j to time b_e^j
$\mathcal{F}(e)$	The sorted set of all the time-windows on edge $e \in E$
$G = (V, E)$	The directed graph representing the airport layout, with vertices $v \in V$ and edges $e \in E$
H	The Fibonacci heap storing the added labels
$I_L = [a_L, b_L]$	The time interval used in a label L
$L = (v_L, I_L, pred_L)$	A label on vertex $v_L \in V$ with time interval I_L and predecessor label $pred_L$
$\mathcal{L}(v)$	The set of all of the labels at vertex $v \in V$
R	A conflict-free route that is being generated
$T = (s, t, time)$	A taxi request to route, from source $s \in V$ at time $time$ to target $t \in V$
w_e	The weight (necessary taxi time) of edge $e \in E$

4.4 QPPTW algorithm

The input of the QPPTW algorithm contains the graph $G = (V, E)$ with its weight function w_e, which corresponds to the taxi times for each edge, estimated using the taxi time estimation method which was described in Section 3. The sorted set of available time-windows $\mathcal{F}(e)$ also has to be provided for each edge e, specifying when the edge is available. A taxi request $T_i = (s_i, t_i, time_i)$ for aircraft i is then a conflict-free route R from the vertices s_i to t_i with minimal taxi time (w.r.t. w_e) that respects the given time-windows.

The pseudocode of the QPPTW algorithm is shown in Algorithm 1 and is a variant of the QPPTW algorithm described by Stenzel (2008). The main difference is that we allocate the labels to vertices, which helps both to model the process more realistically and to more easily understand the algorithm, since it distinguishes between the use of the labels at the vertices and the input time-windows at the edges.

Lines 1 and 2 of Algorithm 1 involve the initialization of the Fibonacci heap and the references to this heap which are stored at each vertex. The use of Fibonacci heaps for this algorithm has the same beneficial effect upon the execution time as it does for Dijkstra's algorithm. The starting label is generated for the source s_i in line 3 and is then inserted into the Fibonacci heap, which is sorted with respect to the earliest possible arrival time (key). A reference is maintained to this label using the $\mathcal{L}(s_i)$ set for each vertex. These references are used as a look-up by the dominance check in lines 23-29, where the algorithm needs fast access to all of the labels associated with a particular vertex.

In each iteration of the while loop, the algorithm checks whether the Fibonacci heap still contains elements. If this is not the case, there is no route which can be enlarged and, therefore, no route from s_i to t_i, starting at $time_i$, exists (line 32). If the Fibonacci heap still contains elements, the algorithm takes a minimal element with respect to the key (line 7), checks whether this label already represents a route to the target t_i (lines 8-10) or, otherwise, tries to expand the associated route.

The route can usually continue along a number of different outgoing edges from any vertex and can potentially use different time-windows on each edge (lines 11 and 12). In order to use an edge there must be a time-window available with an overlapping time interval, as expressed by the conditions on lines 14 and 16. The earliest possible point in time that edge e_L can be left is identified (lines 18 and 19) and the expansion step is executed. When the condition stated in line 20 is true, a new label will be generated (lines 21 and 22). Different cases are possible at this stage. Firstly, the new label may dominate another label (line 27), in which case the dominated label will be erased (lines 28 and 29). Secondly, the new label may be dominated by an older one (line 25), in which case it is not necessary to take this label into account (line 26). The while loop is executed as long as there is a route which can be expanded. Once a route R to the target t_i has been found, the route can be generated by working backwards through the set of labels (line 9) using the references, $pred_L$, to the previous labels.

This generalized vertex-based Dijkstra's algorithm is a variant of that given by Stenzel (2008). His proof that the edge-based algorithm solves the problem in polynomial time (in the number of time-windows) will also hold for this algorithm.

Algorithm 1: Quickest Path Problem with Time Windows (QPPTW)

Input: Graph $G = (V, E)$ with weights w_e for all $e \in E$, the set of sorted time-windows $\mathcal{F}(e)$ for all $e \in E$, a taxi request $T_i = (s_i, t_i, time_i)$ with the source vertex $s_i \in V$, the target vertex $t_i \in V$ and the start time $time_i$.

Output: Conflict-free route R from s_i to t_i with minimal taxi time that starts at the earliest at time $time_i$, respects the given time-windows $\mathcal{F}(e)$ or returns the message that no such route exists.

```
1  Let H = ∅
2  Let L(v) = ∅   ∀v ∈ V
3  Create new label L such that L = (s_i, [time_i, ∞), nil)
4  Insert L into heap H with key time_i
5  Insert L into set L(s_i)
6  while H ≠ ∅ do
7      Let L = H.getMin(), where L = (v_L, I_L, pred_L)
       and I_L = [a_L, b_L]
8      if v_L = t_i then
9          Reconstruct the route R from s_i to t_i by
           working backwards from L
10         return the route R
11     forall the outgoing edges e_L of v_L do
12         foreach F^j_{e_L} ∈ F(e_L), where F^j_{e_L} = [a^j_{e_L}, b^j_{e_L}],
           in increasing order of a^j_{e_L} do
13             /*Expand labels for edges where time
               intervals overlap*/
14             if a^j_{e_L} > b_L then
15                 goto 11 /*consider the next outgoing
                   edge*/
16             if b^j_{e_L} < a_L then
17                 goto 12 /*consider the next
                   time-window*/
18             Let time_in = max(a_L, a^j_{e_L})
               /*a^j_{e_L} > a_L ⇒ waiting*/
19             Let time_out = time_in + w_{e_L}
20             if time_out ≤ b^j_{e_L} then
21                 Let u = head(e_L)
22                 Let L' = (u, [time_out, b^j_{e_L}], L)
23                 /*dominance check*/
24                 foreach L̂ ∈ L(u) do
25                     if L̂ dominates L' then
26                         goto 12 /*next
                           time-window*/
27                     if L' dominates L̂ then
28                         Remove L̂ from H
29                         Remove L̂ from L(u)
30                 Insert L' into heap H with key a_{L'}
31                 Insert L' into set L(u)
32 return "there is no s_i-t_i route"
```

4.5 Modifications to the QPPTW algorithm for airport ground movement

Algorithm 1 is used for arriving aircraft as described above, since their goal is to clear the runway and reach the gate/stand as quickly as possible. In our model, departing aircraft aim to reach the runway at a given time and leave the gate/stand as late as possible in order to do so. This allows for more of the waiting time to be absorbed at the gate/stand when the engines are not running. The same algorithm is used for this purpose, computing the route backwards, with the end time fixed instead of the start time, and with changes to reverse the time-related steps. Since the algorithm logic remains unchanged, this modified algorithm has not been presented here.

In an attempt to further speed up the execution time of the algorithm, we applied goal-oriented search (Sedgewick and Vitter 1986) to the QPPTW algorithm. Two heuristic measures were investigated for estimating lower bounds for the rest of the partial route: firstly the Euclidean distance was used to measure the linear distance to the target, and secondly the remaining time was estimated using Dijkstra's algorithm to compute the time which would be needed ignoring any interference from other aircraft. Unfortunately, neither approach resulted in a valuable speed-up when applied to this problem. This can possibly be explained by the fact that the graph representing the airport layout is sparse (having on average only a few outgoing edges for each vertex) and routes often start on the border of the graph (see Figure 1), so the number of expansions exploring non-promising areas of the airport is relatively small already.

4.6 Readjustment of the time-windows

When an aircraft has been routed, the time-windows have to be readjusted according to the edge utilisation of the adopted route R, and the edges which conflict with these. It is necessary to consider edge conflicts only during this stage and not during the routing process (Algorithm 1).

Algorithm 2 presents the pseudocode for the readjustment of the time-windows. The input consists of the weighted graph $G = (V, E)$, the set of conflicting edges $confl(e)$ for all $e \in E$, the set of sorted time-windows $\mathcal{F}(e)$ for all $e \in E$, and the route R which was found for the most recent aircraft to be routed. The output is the new sorted set of time-windows $\mathcal{F}(e)$, including the reservations of the new route R.

Basically, the algorithm determines which other edges are blocked for each edge of the route R (lines 1 and

Algorithm 2: Readjustment of the time-windows

Input: Graph $G = (V, E)$ with weights w_e for all $e \in E$, the route R with reservations $\left[time_f^{in}, time_f^{out}\right]$ for all $f \in R$, the set of sorted time-windows $\mathcal{F}(e)$ for all $e \in E$ and the set of conflicting edges $confl(e)$ for all $e \in E$.
Output: Sorted set of time-windows $\mathcal{F}(e)$ including the reservations of the route R

```
1  foreach f ∈ R do
2      foreach e ∈ confl(f) do
3          foreach F_e^j = [a_e^j, b_e^j] ∈ F(e) do
4              if time_f^out ≤ a_e^j then
5                  goto 2    /*time-window is too late*/
6              if time_f^in < b_e^j then
7                  /*otherwise time-window is too
                   early*/
8                  if time_f^in < a_e^j + w_e then
9                      if b_e^j − w_e < time_f^out then
10                         Remove F_e^j from F(e)
11                     else
12                         /*shorten start of
                           time-window*/
13                         F_e^j = [time_f^out, b_e^j]
14                 else
15                     if b_e^j − w_e < time_f^out then
16                         /*shorten end of
                           time-window*/
17                         F_e^j = [a_e^j, time_f^in]
18                     else
19                         /*split time-window*/
20                         F_e^j = [a_e^j, time_f^in]
21                         Insert [time_f^out, b_e^j] into set
                           F(e)
```

2). All affected time-windows on these edges are adjusted (lines 3-7) and four different cases then have to be considered, depending upon the relative positions of the time-windows. The remaining time-window may be removed (lines 9-10) if it becomes too short to allow an aircraft to taxi; be shortened at the start (lines 11-13) or shortened at the end (lines 15-17); or it could be split in two smaller windows (lines 18-21).

Once a route has been allocated to an aircraft, some additional waiting times may be required on edges, beyond the time required to traverse the edge as specified by the time intervals on the labels by Algorithm 1. Time intervals on adjacent edges often overlap sufficiently that there is a choice of which edge the wait can be assigned to. In our implementation, the waiting times are forced to be as late in the corresponding part of the route as possible, apart from the initial waiting time for departures, which is allocated so as to maximise the stand hold. Alternative approaches could use this flexibility to select better and smoother speed profiles for the aircraft. Using a similar approach to that used in Lesire (2010), the aim could be to spread the necessary waiting times for an aircraft in such a way that the speed profiles are as "engine friendly" as possible. Although the effects of such postprocessing are not studied within this paper, they are an area which we intend to investigate.

4.7 Buffer times

The solutions of the approach are conflict-free routings, but it is possible for small delays to affect the entire plan. Buffer times would allow small deviations from the taxi times to be absorbed. To achieve such buffer times the label intervals in the algorithm are lengthened in the desired direction (before or after) by a certain amount. To reflect growing uncertainties along the route, the amount of time can be made distance-dependent. Buffer times could also depend upon the expected congestion at the time, being increased when delays were expected to be more likely, although at these times the introduction of a buffer time would be more likely to reduce throughput.

4.8 Initial sequencing of taxiing aircraft

The order in which aircraft are considered by the sequential routing algorithm can potentially affect the efficiency of the routing. The natural sequencing, of considering aircraft in the order in which they become available, has advantages in terms of perceived fairness and has been adopted in the past. A more advanced approach using a concept of collaborative virtual queues was presented in Burgain et al. (2009), with the idea being to limit the number of aircraft which were taxiing on the surface to a specified maximum and maintaining a virtual queue of those waiting to start, forcing them to wait until the count allows them to pushback. The natural ordering (the expected wheel-on time on the runway for arrivals and the expected earliest pushback time at the gate/stand for departures) was adopted by default for this paper, but the potential benefits of using better sequences have also been considered, as explained in the next section.

4.9 Heuristic for finding better aircraft sequences

A number of heuristic improvements to the order in which aircraft are considered were evaluated, as described in Ravizza and Atkin (2011). A simple swap heuristic was found to perform well without prohibitively increasing the execution time of the algorithm, as shown in Section 5.3.

The key idea behind the heuristic is the following: If an aircraft is routed and scheduled using the decision support system, the aircraft either has an uninterrupted planned taxi route or the aircraft has a detour and/or a delay due to the presence of another aircraft. Nothing has to be done in the first case, but in the second case, the heuristic identifies the causer aircraft, the one causing the delay or re-routing. If multiple aircraft are affecting the current aircraft, the one which affects the planned aircraft's route the earliest is used as the causer aircraft. The swap-heuristic then attempts to swap the order of the causer and current aircraft in the aircraft sequence and the new generated routes and times are used if the total taxi time is lower with the swap than without it.

Importantly, the considered sequence stays close to the initial sequence, reducing the number of changes which need to be communicated to pilots in a dynamic situation, keeping the communication to a minimum. To further reduce the schedule/route changes, all of the other aircraft's routes and schedules are fixed. Furthermore, the swap-heuristic does not make changes to the initial sequences or the timings at the runways.

5 Results and discussions

This section starts with a table collating the key results, to ease comparison. The explanation of the results follows. The results of the taxi time estimation which was presented in Section 3 are then discussed. An analysis of the results from the ground movement decision support system, which was described in Section 4, is then provided and followed by more detailed results considering the swap-heuristic.

The relevant results are summarised in Table 2. The first row of results shows the actual total and average taxi times for the supplied dataset, including queuing time at the runway. The taxi time function which was developed was then applied to each aircraft, to estimate the taxi times and the results are shown in the next two rows. In the first case, the function was applied assuming the actual traffic level and we note that the difference between the predicted and actual times is less than 2%. In the second case, the traffic related components of the function were zeroed, to estimate the taxi times if there had been no delays due to other aircraft, and the difference illustrates the amount of the taxi time which was a result of such delays. The unimpeded taxi times were then used within the QPPTW algorithm based on FCFS ordering of the aircraft and the total and mean resulting taxi times are shown in the table. These results are analysed and explained further in the following two sections.

5.1 Analysis of taxi time estimation

Once the pushback duration had been included in the Mamdani fuzzy rule-based system (see Section 3.2), the coefficient of determination R^2 of 94.15% showed that the FRBS was able to explain the variability of the taxi time data very well for the real world Zurich dataset.

The fitted FRBS model was then used to predict a taxi time for each aircraft in the dataset, with and without the factors which represented the effects of the delays due to other aircraft (see Section 3.2). The results can be seen in Table 2. The model predicts that 31.4% of the taxi time was related to delays due to other aircraft, including delays in queues behind other aircraft at the runway. There would be an average saving of 137.7s per aircraft if these delays could be eliminated. The influence of the interactions between the aircraft which lead to the waiting times is analysed in the next section.

5.2 Experimental details using the QPPTW algorithm

The framework was programmed in Java as a single-threaded application and executed on a personal computer (Intel Core 2 Duo, 3GHz, 2GB RAM). In these experiments, all aircraft were allowed to use all of the taxiways and only intersecting and adjacent edges were considered to be in conflict and were, therefore, not allowed to be used by two aircraft simultaneously. The buffer time (Section 4.7) was set to zero. Analysis of different buffer times showed that the taxi time would have been enlarged by only a linear factor of the buffer time. Similar results were also found in Ravizza (2009).

Extensive analysis was performed using the QPPTW algorithm, with FCFS consideration sequence for aircraft, to solve the ground movement problem using the data from and layout of Zurich Airport. The aircraft were routed sequentially using the taxi speed estimations from the fuzzy rule-based system which was discussed in Sections 3, 3.2 and 5.1. The resulting total taxi times can be found in Table 2, where the taxi times used were those which were estimated for unimpeded aircraft

Table 2 Summary of the results

	Total taxi time [s]	Average taxi time per aircraft [s]
Actual total taxi time	2489262.0	443.5
Fuzzy rule-based system		
Total taxi time estimation	2458400.4	438.0
Total taxi time estimation (unimpeded)	1685798.5	300.3
QPPTW algorithm with FCFS		
Using unimpeded taxi time estimates	1736020.9	309.3

(ignoring the influence of factors related to other aircraft on the surface), the average taxi time (including re-routing and waiting delays) was 309.3s per aircraft.

The estimations of the unimpeded taxi times from the FRBS prediction approach provide a lower bound for the taxi times, since they assume no re-routing delays or queuing behind other aircraft. The QPPTW algorithm is designed to predict the delays which are actually necessary due to the interactions between aircraft for the specific routings and timings which the algorithm assigns to aircraft. Comparison of the resulting taxi times from the QPPTW algorithm against the lower bound reveals an increase in the taxi time from 1685798.5 to 1736020.9 seconds, showing that the additional taxi times for the re-routing and waiting summed to 50222.4s over the entire week, an increase of around 3% in the total taxi time. The 3% increase over the lower bound (rather than optimal) times indicates that its use as a ground movement decision support system seems very promising for this problem.

It is also interesting to compare the approach described here against the actual performance of the airport on this particular week of operation. Data from Zurich Airport reports a total taxi time of 2489262.0s. Comparison with the results for the QPPTW algorithm with unimpeded taxi time estimation highlights savings of about 30.3% or an average of 134.2s per aircraft. Obviously, this only indicates an upper bound for the potential savings, since the real times will include some slack time for the departures at the runway to ensure a high runway throughput.

The solution time to solve the entire week of operation with 5613 aircraft was 216887ms, an average solution time of 39ms per aircraft. This supports the potential use of the algorithm in an online decision support system. No infeasible solution occurred within any of the executions of the experiments. These findings are consistent with earlier work by Atkin et al. (2011b), using another dataset from Zurich Airport (from 2007) and taxi times which were generated from the linear regression approach.

5.3 Studies of a swap-heuristic

Table 3 provides a comparison of the routing and scheduling algorithm with and without the swap-heuristic. The different columns represent the different days in the dataset and the total for the entire week. The first three rows of the table report the number of aircraft movements during each day and it can be seen that at the weekend (day 6 and day 7), the airport has lighter traffic. Rows two and three differentiate between departures (DEP) and arrivals (ARR). The second block shows the results of the QPPTW algorithm with the FCFS order (without the swap-heuristic) and the third block shows the results with the swap-heuristic. The lower bound was computed using the estimated taxi times but with each aircraft routed in isolation, so no waiting times or detours were included. The following block shows the absolute gap between the lower bound and the results for the FCFS and the swap-heuristic, respectively. The reduction in the gap is the relative improvement from using the swap-heuristic compared with the FCFS ordering.

The results were similar for the different days and the total taxi times were approximately double for departures compared to arrivals, independent of the sequencing method. Obviously, the more advanced swap-heuristic increased the solution time per aircraft, however, the algorithm is still fast enough to be used in an online environment. The approach would also be fast enough to respond to unforeseen delays of aircraft and could instantaneously adjust a schedule and the choice of the route, if needed.

The swap-heuristic based sequencing method was able to reduce the gap between the routing which was found and the lower bound by 30% on average over the entire week, with a bigger reduction rate for departing aircraft (33%) than arriving aircraft (25%).

The sorted individual delays for the aircraft which resulted from the analysis with and without the swap-heuristic are shown in Figure 4. In both cases, at least the first 4578 (out of 5613) aircraft had no delays in their planned schedules and are not included in the fig-

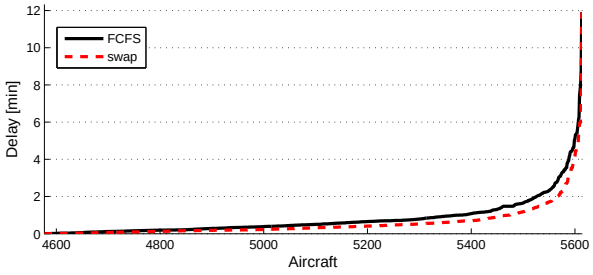

Fig. 4 Sorted delay for each aircraft with and without swap-heuristic

ure. The delays from Figure 4 are summarised in Table 4, showing the percentage of aircraft which have more than a certain amount of delay. The swap-heuristic was able to improve most of the percentages by almost a factor of 2. Again, these results are consistent with earlier work by Ravizza and Atkin (2011) which was based on an older dataset from the same airport.

Table 4 Percentage of aircraft with more than a certain amount of delay

	Without swap-heuristic	With swap-heuristic
Have a delay	18.47%	18.31%
More than 1min	4.22%	2.51%
More than 2min	1.57%	0.86%
More than 3min	0.80%	0.48%
More than 4min	0.45%	0.27%
More than 5min	0.27%	0.12%

5.4 Scenarios with more ground traffic

New scenarios were generated based on the data from summer 2011, simulating more ground traffic at Zurich Airport. The analysis focused upon Monday as a representative day. Each movement of an aircraft was duplicated and the copy was shifted by 30 minutes to generate the scenario with 200% ground traffic. For the 300% scenario each movement was duplicated twice and one copy was shifted by 15 minutes and the other copy by 30 minutes. The scenarios for the settings with 120%, 140%, 160% and 180% were generated by randomly removing some of the duplicated aircraft movements from the 200% case and the scenarios between 200% and 300% were created by randomly removing movements from the second duplication. It has to be noted that within this analysis the focus was entirely upon analysing the ground movement problem with more ground traffic and, obviously, separations and deadlines were considered for neither taking-off nor landing (since the runway throughput would not be achievable), nor was it guaranteed that no overlaps occurred in the gate allocations. The aim is to consider only whether the algorithm can cope with increased traffic load, and if so what the consequent delays are which would be allocated to aircraft.

Table 5 shows the results of the analysis. Each column represents a scenario with the appropriate amount of ground traffic related to the actual setting. The table is structured similarly to Table 3 to ease comparison. It can be seen that the lower bound increases linearly which is due to the construction of the problems. The numbers also show an approximately linear increase of the approach which was based on the FCFS consideration of aircraft until the ground traffic reached the 240% level. After that the gap between the QPPTW algorithm without the swap-heuristic increased from values between 3% to 9% before that to values between 16% and 27% after it. The swap-heuristic achieved an average of a 22% reduction in the gap between the lower bound and the QPPTW algorithm with FCFS ordering. This was relatively consistent for the scenarios with lower traffic and higher traffic levels. The only exception was the 240% scenario, where the reported reduction of the gap was only 4%. The implementation of the swap heuristic was, therefore, generally worthwhile.

5.5 Further use for simulations

The main purpose of this paper is to enhance decision support systems which can be used in control towers. Nevertheless, a prototype of this approach can also be used for simulations of management or operational strategies. From an airport point of view several kinds of analysis would be possible. A taxiway layout could be analysed to highlight where the bottlenecks are and by how much the operations are restricted if a part of the network is blocked, such as for maintenance requirements. Airports often have a concept of where certain aircraft should be routed and variations of such concepts could also be tested by either restricting certain combinations of taxiway parts or by favouring certain combinations. Furthermore, a ground movement simulation could be integrated with runway sequencing or gate assignment to perform a broader analysis.

Airlines could also use simulations to better understand the situation at an airport, to improve their own operations. For instance, they could be used to identify which times of the day are less likely to cause waiting times. Airlines could then adjust their schedules to improve the operational performance, assuming that the other carriers maintain their existing schedules. A good

example of such a study is "Delta's Operation Clockwork" (Petroccione 2007). De-peaking of their operations at Hartsfield-Jackson International Airport was able to save waiting times for aircraft equivalent to adding nineteen aircraft into the fleet, which were then re-inserted into the system to provide more connections. A test phase confirmed the findings of the analysis, but the airline decided to revert to the old schedule afterwards due to reductions in revenue from reduced passenger demand.

Simulation has been widely used by research groups and software vendors to get insights into airport operations and to evaluate the impacts of uncertainties. Rosenberger et al. (2000, 2002) presented a stochastic model for airline operations within the SIMAIR project, with the primary purpose being to evaluate crew scheduling plans and recovery policies. Simulation tools for airport and airspace operations, such as SIMMOD from the Federal Aviation Administration (FAA), RAMS from Eurocontrol, DPM from Sabre and TAAM from the Preston Group, can model existing and planned operations very well, but may lack in the area of automatically improving operations which can be performed with optimisation systems.

5.6 Impact of results

This section highlights the possible savings in fuel costs of the introduced algorithm by using the same approach as in the analysis by Brinton et al. (2011). An average aircraft used 306.6 seconds of fuel burn in our analysis with the integration of taxi time estimation, the QPPTW algorithm and the swap-heuristic, instead of 443.5 seconds as was reported from the historic data. The saving of 136.9 seconds per aircraft movement accumulates to around 637000 minutes per year based on 279000 movements as it was reported at Zurich Airport in 2011. Brinton et al. (2011) based their calculations on a jet aircraft using 25 pounds of fuel per minute while taxiing, which fits the guidelines from ICAO for the settings of a "Single Aisle Jet". With an assumed $4 US per gallon of fuel, the annual cost savings in fuel at Zurich Airport would be approximately $9.6 million. However, it should be noted that other sources question the actual fuel rate for taxiing, which is possibly slightly overestimated by ICAO (Morris 2005; Kim and Rachami 2008).

6 Conclusion

This paper described a more realistic and potentially more environmentally friendly ground movement decision support system, compared to previous approaches. The overall framework is designed to combine the runway sequencing problem and ground movement problem, aiming for better global solutions, although only the ground movement element was considered in this paper. This work extends the basic ground movement problem of minimising the travel times to include the concept of absorbing possible waiting times for departures at the gate/stand, to reduce the fuel burn and environmental impact. The sequential QPPTW algorithm which was described here is based on graph theoretical concepts and can include restrictions such as limitations upon which taxiways aircraft can use, which taxiways block which and when, and any turning limitations at taxiway junctions. In addition, the algorithm provides the opportunity to add buffer times for blocking the reserved taxiways for longer than expected, to absorb small delays and schedule disturbances.

Experiments used data for an entire week of operations at Zurich Airport, the largest hub airport in Switzerland. This data was used to generate more accurate taxi time estimations for each aircraft, using a taxi time prediction function which was generated from an extensive statistical analysis and a fuzzy rule-based system, applied to the same dataset. These taxi time estimations were then utilised within the QPPTW algorithm to route and schedule the ground movement. The results are very promising and show potential maximum savings in total taxi time from using the decision support system described here, together with the taxi time prediction system, of about 30.3%, compared to the actual performance at the airport. Further research is necessary to determine the amount of buffer time and runway delay which should be utilised to account for any remaining taxi time uncertainty and to avoid starving the runways.

The experimental results of the developed decision support approach show average solution times of only a few milliseconds per aircraft, and are, therefore, adequate for the implementation of such a system for real time use at airports.

We intend to investigate various extensions of this work in future, in addition to the combination of the ground movement problem with the runway sequencing problem. Firstly, the QPPTW algorithm enables the possible waiting times to be spread in different ways. In this paper, they were allocated so as to maximise the stand hold time and to better adapt to schedule disturbances, but an alternative approach would be to develop smoother speed profiles for aircraft, using the engine in a more efficient and environmentally friendly way. Secondly, we would like to perform a similar analysis for different airport layouts, to better understand the

effects of the layout upon the best solution approach, but it will be necessary to obtain more data and support from other airports in order to do so.

Acknowledgements The authors wish to thank the Engineering and Physical Sciences Research Council (EPSRC) for providing the funding which made this research possible. We would also like to thank Flughafen Zürich AG who provided the real dataset and especially Giovanni Russo for his continuous support and Daniele Gullo for valuable feedback and suggestions. Moreover, the authors thank the anonymous reviewers who have helped to improve this paper.

References

ACI EUROPE (2010). An outlook for Europe's airports - Facing the challenges of the 21st century. *Technical Report*, Airports Council International Europe, www.aci-europe.org.

Atkin JAD, Burke EK, Greenwood JS & Reeson D (2007). Hybrid metaheuristics to aid runway scheduling at London Heathrow Airport. *Transportation Science* 41(1):90–106, DOI 10.1287/trsc.1060.0163.

Atkin JAD, Burke EK & Greenwood JS (2010a). TSAT allocation at London Heathrow: the relationship between slot compliance, throughput and equity. *Public Transport* 2(3):173–198, DOI 10.1007/s12469-010-0029-2.

Atkin JAD, Burke EK & Ravizza S (2010b). The airport ground movement problem: Past and current research and future directions. In: *Proceedings of the 4th International Conference on Research in Air Transportation (ICRAT 2010)*, Budapest, Hungary, pp 131–138.

Atkin JAD, Burke EK & Greenwood JS (2011a). A comparison of two methods for reducing take-off delay at London Heathrow Airport. *Journal of Scheduling* 14(5):409–421, DOI 10.1007/s10951-011-0228-y.

Atkin JAD, Burke EK & Ravizza S (2011b). A more realistic approach for airport ground movement optimisation with stand holding. In: *Proceedings of the 5th Multidisciplinary International Scheduling Conference (MISTA 2011)*, Phoenix, Arizona, USA.

Balakrishnan H & Jung Y (2007). A framework for coordinated surface operations planning at Dallas-Fort Worth International Airport. In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, USA.

Brinton C, Provan C, Lent S, Prevost T & Passmore S (2011). Collaborative departure queue management: An example of airport collaborative decision making in the United States. In: *Proceedings of the 9th USA/Europe Air Traffic Management Research and Development Seminar*, Berlin, Germany.

Burgain P, Feron E & Clarke JP (2009). Collaborative virtual queue: Benefit analysis of a collaborative decision making concept applied to congested airport departure operations. *Air Traffic Control Quarterly* 17(2):195–222.

Chen J, Ravizza S, Atkin JAD & Stewart P (2011). On the utilisation of fuzzy rule-based systems for taxi time estimations at airports. In: *Proceedings of the 11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2011)*, Saarbrücken, Germany, pp 134–145, DOI 10.4230/OASIcs.ATMOS.2011.134.

Cormen TH, Leiserson CE, Rivest RL & Stein C (2001). *Introduction to Algorithms*, 2nd edn. MIT Press and McGraw-Hill.

Dijkstra EW (1959). A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271, DOI 10.1007/BF01386390.

Dorndorf U, Drexl A, Nikulin Y & Pesch E (2007). Flight gate scheduling: State-of-the-art and recent developments. *Omega* 35(3):326–334, DOI 10.1016/j.omega.2005.07.001.

Gawrilow E, Köhler E, Möhring R & Stenzel B (2008). Dynamic routing of automated guided vehicles in real-time. In: *Mathematics - key technology for the future*, Springer, pp 165–178.

Idris HR, Clarke JP, Bhuva R & Kang L (2002). Queuing model for taxi-out time estimation. *Air Traffic Control Quarterly* 10(1):1–22.

Kim B & Rachami J (2008). Aircraft emissions modeling under low power conditions. In: *Proceedings of the A&WMA's 101st Annual Conference and Exhibition*, Portland, Oregon, USA.

Lesire C (2010). Iterative planning of airport ground movements. In: *Proceedings of the 4th International Conference on Research in Air Transportation (ICRAT 2010)*, Budapest, Hungary, pp 147–154.

Marín Á (2006). Airport management: Taxi planning. *Annals of Operations Research* 143(1):191–202, DOI 10.1007/s10479-006-7381-2.

Marín Á & Codina E (2008). Network design: Taxi planning. *Annals of Operations Research* 157(1):135–151, DOI 10.1007/s10479-007-0194-0.

Morris KM (2005). Results from a number of surveys of power settings used during taxi operations. *Technical Report*, ENV/KMM/1126/14.8, British Airways, www.britishairways.com/cms/global/pdfs/csr/PSDH_Technical_Reports.pdf.

Petroccione L (2007). Delta's operation clockwork, transforming the fundamentals of an airline. *Principal Transportation & Logistics Practice, Decision Strategies, Inc.*.

Rappaport DB, Yu P, Griffin K & Daviau C (2009). Quantitative analysis of uncertainty in airport surface operations. In: *Proceedings of the AIAA Aviation Technology, Integration, and Operations Conference*.

Ravizza S (2009). Control of automated guided vehicles (AGVs). *Master's thesis*, ETH Zurich, Switzerland.

Ravizza S & Atkin JAD (2011). Exploration of the ordering for a sequential airport ground movement algorithm. *Technical Report*, 1543, University of Nottingham, http://eprints.nottingham.ac.uk/1543/.

Ravizza S, Atkin JAD, Maathuis MH & Burke EK (2012). A statistical approach for improving taxi time estimations at airports. *Journal of the Operational Research Society* (advance online publication), DOI 10.1057/jors.2012.123.

Roling PC & Visser HG (2008). Optimal airport surface traffic planning using mixed-integer linear programming. *International Journal of Aerospace Engineering* 2008(1):1–11, DOI 10.1155/2008/732828.

Chen J, Ravizza S, Atkin JAD & Stewart P (2011). On the utilisation of fuzzy rule-based systems for taxi time estimations at airports. In: *Proceedings of the 11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2011)*, Saarbrücken, Germany, pp 134–145, DOI 10.4230/OASIcs.ATMOS.2011.134.

Rosenberger JM, Schaefer AJ, Goldsman D, Johnson EL, Kleywegt AJ, Nemhauser & George L(2000). Air transportation simulation: SimAir: a stochastic model of airline operations. In: *Proceedings of the 32nd conference on Winter simulation*, San Diego, CA, USA.

Rosenberger JM, Schaefer AJ, Goldsman D, Johnson EL, Kleywegt AJ, Nemhauser & George L(2002). A Stochastic Model of Airline Operations. *Transportation Science* 36(4):357–377, DOI 10.1287/trsc.36.4.357.551.

Sedgewick R & Vitter JS (1986). Shortest paths in euclidean graphs. *Algorithmica* 1(1):31–48, DOI 10.1007/BF01840435.

Stenzel B (2008). Online disjoint vehicle routing with application to AGV routing. *Phd thesis*, TU Berlin, Germany.

Table 3 Analysis of the routing and scheduling algorithm with and without swap-heuristic over one week

		Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total
# Aircraft	Total	818	806	781	839	825	757	787	5613
	DEP	407	405	392	416	421	379	387	2807
	ARR	411	401	389	423	404	378	400	2806
FCFS	Total taxi time [s]	251231.3	248751.7	244449.9	256497.1	256662.5	234632.2	243796.2	1736020.9
	Total taxi time DEP [s]	168731.8	168503.6	172718.4	172070.7	183031.6	163713.7	168059.2	1196829.0
	Total taxi time ARR [s]	82499.5	80248.1	71731.5	84426.4	73630.8	70918.5	75737.0	539191.9
	Solution time [ms]	33640	32562	28765	33078	30483	28859	29500	216887
	Solution time per aircraft [ms]	41	40	37	39	37	38	37	39
Swap-heuristic	Total taxi time [s]	249355.0	246128.0	242097.1	253869.1	254924.1	233204.3	241216.3	1720793.8
	Total taxi time DEP [s]	167382.5	166201.3	171258.8	170221.0	181683.2	162652.0	166168.5	1185567.4
	Total taxi time ARR [s]	81972.5	79926.7	70838.3	83648.1	73240.8	70552.2	75047.7	535226.4
	Solution time [ms]	123919	109248	110685	117701	101373	89311	101248	753485
	Solution time per aircraft [ms]	151	136	142	140	123	118	129	134
Lower bound	Total taxi time [s]	243699.3	240061.6	237926.9	248121.3	250165.6	228864.3	236911.6	1685750.5
	Total taxi time DEP [s]	163890.8	161979.0	169068.7	166165.2	178146.0	159547.7	163469.9	1162267.3
	Total taxi time ARR [s]	79808.5	78082.7	68858.1	81956.1	72019.5	69316.6	73441.6	523483.2
FCFS gap	Total taxi time [s]	7532.1	8690.1	6523.0	8375.8	6496.9	5767.9	6884.7	50270.4
	Total taxi time DEP [s]	4841.0	6524.7	3649.7	5905.5	4885.6	4166.0	4589.3	34561.7
	Total taxi time ARR [s]	2691.0	2165.4	2873.4	2470.3	1611.3	1601.9	2295.4	15708.7
Swap-heuristic gap	Total taxi time [s]	5655.7	6066.4	4170.2	5747.8	4758.5	4340.0	4304.7	35043.3
	Total taxi time DEP [s]	3491.7	4222.3	2190.1	4055.8	3537.2	3104.4	2698.6	23300.1
	Total taxi time ARR [s]	2164.0	1844.1	1980.2	1692.0	1221.3	1235.6	1606.1	11743.2
Reduction of gap	Total taxi time	25%	30%	36%	31%	27%	25%	37%	30%
	Total taxi time DEP	28%	35%	40%	31%	28%	25%	41%	33%
	Total taxi time ARR	20%	15%	31%	32%	24%	23%	30%	25%

Table 5 Analysis of the routing and scheduling algorithm with and without swap-heuristic with artificially more ground traffic

		100%	120%	140%	160%	180%	200%	220%	240%	260%	280%	300%
FCFS	Total taxi time [s]	251231	304523	354337	406332	463119	522593	581120	640659	736357	858409	929010
	Total taxi time DEP [s]	168732	204889	235528	267120	306219	350662	390517	432088	506596	607601	657527
	Total taxi time ARR [s]	82500	99634	118809	139212	156900	171931	190604	208571	229761	250808	271483
Swap-heuristic	Total taxi time [s]	249355	301591	349673	401258	456006	513862	570208	638429	715518	827778	887346
	Total taxi time DEP [s]	167383	202594	232713	264669	302068	345455	384747	435042	492151	585126	624224
	Total taxi time ARR [s]	81973	98997	116961	136589	153938	168407	185461	203387	223367	242652	263122
Lower bound	Total taxi time [s]	243699	293209	339086	385438	435692	487399	537638	586888	633717	682267	731098
	Total taxi time DEP [s]	163891	197666	226342	254709	289150	327782	362454	397385	427691	459642	491672
	Total taxi time ARR [s]	79808	95543	112744	130729	146542	159617	175184	189504	206026	222626	239425
FCFS gap	Total taxi time	3%	4%	4%	5%	6%	7%	8%	9%	16%	26%	27%
Swap-heuristic gap	Total taxi time	2%	3%	3%	4%	5%	5%	6%	9%	13%	21%	21%
Reduction of gap	Total taxi time	25%	26%	31%	24%	26%	25%	25%	4%	20%	17%	21%