Optimization of Taxiway Routing and Runway Scheduling

Gillian Clare* and Arthur Richards†

University of Bristol, Bristol, UK

Abstract

This paper describes a Mixed Integer Linear Programming (MILP) optimization method for the coupled problems of airport taxiway routing and runway scheduling. The Receding Horizon (RH) formulation and use of iteration in the avoidance constraints allows the scalability of the baseline algorithm presented to be illustrated, with examples based on Heathrow airport containing up to 240 aircraft. The results show that average taxi times can be reduced by 55% when compared to a First Come First Serve (FCFS) approach. The main advantage is seen with the departure aircraft flow. Comparative testing demonstrates that iteration reduces the computational demand of the required separation constraints whilst introducing no loss in performance. RH formulation enables near real-time operation as the computation is spread between horizons preventing unnecessarily detailed plans being calculated for the distant future. The conversion to RH does however result in an approximation to the globally optimal solution, although this is shown to be relatively small.

Manuscript received

* Ph.D. Candidate, Department of Aerospace Engineering, University of Bristol, Queens Building, University Walk, Bristol BS8 1TR; gk2661@bristol.ac.uk, Student Member AIAA.
† Lecturer, Department of Aerospace Engineering, University of Bristol, Queens Building, University Walk, Bristol BS8 1TR; Arthur.Richards@bristol.ac.uk, Member AIAA.
**Nomenclature**

\(a, b, c\) Identify Aircraft

\(a_L(n)\) The last aircraft to visit node \(n\) prior to current horizon

\(a_x\) Auxiliary aircraft

\(\mathcal{A}\) Set of all active aircraft

\(\mathcal{A}_{arr}\) Set of all arriving aircraft

\(\mathcal{A}_{dep}\) Set of all departing aircraft

\(\mathcal{A}_x\) Set of all auxiliary aircraft

\(C(n,m)\) Absolute binary connectivity matrix

\(C_a(n,m)\) Binary connectivity matrix for aircraft \(a\)

\(d_t(a,b,n)\) Temporal separation required between aircraft \(a\) and \(b\) at node \(n\)

\(D_{SE}\) Binary Variable

\(D_{OT}\) Binary Variable

\(D_{HO}\) Binary Variable

\(FCFS\) First Come First Serve

\(g_{SE}\) Binary Ordering Switch for Separation conflict

\(g_{CO}\) Binary Ordering Switch for Cross-Over conflict

\(g_{RO}\) Binary Ordering Switch for Runway conflict

\(k, j\) Identify planning periods

\(\mathcal{G}\) Set of all gate nodes

\(L(n,m)\) Topology distances
\( L_{\text{taxi}}(a) \) Distance taxied previous to current problem

\( M \) Large number

\( n, m \) Identify nodes (intersections) on the taxiways

\( n_0(a) \) Origin node of aircraft \( a \)

\( n_D(a) \) Destination node of aircraft \( a \)

\( n_F(a) \) Final node in aircraft \( a \)'s detailed plan

\( n_{Ra} \) Arrival runway node

\( n_{Rd} \) Departure runway node

\( n_v(a) \) Virtual node introduced by aircraft \( a \)

\( N \) Number of aircraft in the problem

\( N_n \) Number of nodes in the airport graph structure

\( N_k \) Number of planning periods in the problem

\( N \) Set of all nodes in the airport graph structure

\( CO, CO, CO_{\text{det}} \) Set of crossover conflicts: all, those enforced in iteration \( i \), those detected

\( r_{ST}(n,m) \) Shortest path through the airport graph structure between nodes \( n \) and \( m \)

\( R(a,k) \) Routing variable recording the node at which aircraft \( a \) begins planning period \( k \)

\( RH \) Receding Horizon

\( RCO \) Set of all possible departure runway conflicts

\( SE, SE, SE_{\text{det}} \) Set of separation conflicts: all, those enforced in iteration \( i \), those detected

\( t_D(a) \) Destination time for aircraft \( a \)'s

\( t_{\text{start}}(a) \) Universal time at which aircraft \( a \) started taxiiing

\( t_0(a) \) Origin time of aircraft \( a \)

\( T(a,k) \) Time at which aircraft \( a \) passes the node in it's \( k \)th planning period
$T_L(n)$ Time at which node $n$ was last visited prior to current horizon

$T_t(a)$ Time aircraft $a$ taxied prior to current horizon

$V(a)$ Maximum speed of aircraft $a$

$w_i$ Weightings in the objective function

$X(a,n,m,p)$ Binary representation of aircraft movement plans

## 1 Introduction

Meeting the increasing air traffic demand, which is expected to more than double between 2005 and 2020 [1], within the current air traffic system requires improvements in all areas of Air Traffic Management (ATM). Significant improvements in en-route ATM are shifting the system bottleneck to the limited airport capacity. Congestion on the airport surface is a major constraint to the available capacity of the air transport system. Economically, congestion reduces the turnaround efficiency, while environmentally the increase in both air pollutant and noise emissions impact negatively upon the local region. Furthermore, congestion causes concerns for controller workload [2] and increased risk of runway incursion [3]. The practical difficulties of increasing capacity through airport expansion introduces the desire for enhanced airport ground movement efficiency by intelligent use of the existing resources. Projects such as Eurocontrol’s Advanced-Surface Movement, Guidance and Control Systems (A-SMGCS) [4], and the European Commission’s European airport Movement Management by A-SMGCS (EMMA) [5] have begun the development of new technology and systems of operation to improve upon current practices.

Optimization of airport ground operations is an important topic in modern transportation
research and several approaches have been investigated. Surface movement research has mainly focussed on subsets of the problem which can be classified as: arrival and departure management, including runway sequencing and scheduling [6–9]; stand allocation [10]; and taxi planning, including route allocation followed by local deconfliction [11], and route scheduling [12]. Ref.13 introduces three separate yet interacting schedulers for the runway, ramp and taxiway system. Ref.14 performs both routing and timing optimization, although using discrete time steps and not including take-off separation on the runway. In contrast this paper considers the problem in continuous time, which prevents conservative rounding, considered to be beneficial [15] and incorporates departure runway scheduling, as this has been identified as the key flow constraint to surface movement [16].

This paper describes an automated tool for airport taxiway routing and runway scheduling which applies non-convex optimization to the surface movement problem. The key features of this work are coupled routing and timing optimization, incorporation of runway scheduling, and the implementation in continuous time. This combines discrete decisions, choosing among predetermined taxiways, with continuous decisions, relating to the timing of the planned movement. The method in this paper adopts Mixed Integer Linear Programming (MILP) optimization, known to be suited to such hybrid problems [17]. MILP has previously been successfully used in Air Traffic Management optimizers for both en-route [18] and ground [11,12] traffic. This type of formulation can be solved using efficient, commercial software, such as CPLEX [19], generating globally optimal solutions.

Airport surface movement optimization is NP-hard [20] due to the dynamic traffic assignment when the coupled non-convex decisions of routing and timing are simultaneously optimized. There are many individual constraints such as push-back times, taxiway layouts, and runway separations all requiring consideration. The runway operations alone are constrained by both wake vortex
separations and downstream flight path, or Standard Instrument Departure (SID) separations [21].

This level of complexity results in computationally demanding optimizations. The dynamic nature of the problem, with aircraft entering and leaving, and the possibility of unexpected events, requires plans to be updated regularly. These factors motivate a formulation with reduced computational demand. This was achieved by the RH formulation and use of iteration in the avoidance constraints in the baseline algorithm presented.

RH schemes have been used extensively before in trajectory planning [22,23] as the framework is well suited to such problems. The optimization combines detailed near-term planning over a finite period called the planning horizon with coarse far-term approximations beyond this; this plan is then implemented up to a shorter ‘execution’ horizon, before re-planning occurs, with inputs from the state that has been reached during execution. The continuous nature of traffic at airports is well modelled by the continuation of one horizon to the next, as this allows for inclusion of a rolling window [9,12]. Practically this introduced scalability, as the optimizer considers only a ‘window’ of the aircraft from the entire problem in and given horizon, with aircraft able to enter and leave the problem between the optimizations for each horizon. As a result, computation is spread between horizons preventing unnecessarily detailed plans being calculated for the distant future. However, this approach results in an approximation to the overall optimization problem solution.

Iteration has previously been demonstrated to be an effective tool for reducing computation with no loss in performance when many constraints are redundant [24,25]. In this work it is done with respect to the avoidance constraints. Instead of solving the full problem from the beginning, all of the avoidance constraints are initially relaxed. When a solution is found, avoidance violations are identified, and constraints are re-applied to the problem and the MILP is solved again. This process is repeated until a solution with no constraint violation is found.
The structure of the paper is as follows: Section 2 presents the statement of the problem to be tackled, including the iteration, Section 3 describes the formulation in MILP. The results of a large scale problem are presented in Section 4.2, and comparative testing in Section 4.3 demonstrates how the RH and iterative features of the baseline algorithm contribute to the computation. Finally, performance comparisons from the comparative testing are discussed in Section 4.4.

2 General Problem Statement

This section presents the general problem formulation, beginning with the optimization structure and progressing through the basic parameters, constraints and objective. Further details of the MILP implementation of equations is presented in Section 3. The notation is developed in the text, but to improve readability, the following standards for indices are used throughout: $a$, $b$ and $c$ represent aircraft; $n$ and $m$ represent nodes; and $i$, $j$ and $k$ represent planning periods.

2.1 Optimization Structure

The optimization scheme is formulated in a Receding Horizon (RH) framework, meaning that a single large planning problem is approximated as a sequence of smaller problems. The active taxiing aircraft plan for a number, $N_k$, of planning periods, or planning horizon, this plan is then carried out over a shorter execution horizon before replanning occurs. As the aircraft may not reach their destination nodes in the finite planning horizon, an approximate cost-to-go for the remaining travel to their destination nodes is incorporated. This allows computation to be spread over time, and ensures that computational effort is not wasted on detailed plans for the distant future, which are likely to be revised multiple times before execution. It also results in an approximation to
the overall problem solution. However, our previous results have shown that the effects of this approximation are insignificant whereas the computation saving is large [26].

Planning for problem instances which span periods of time much longer than the planning and execution horizons are naturally handled with a rolling window of aircraft. The idea is that aircraft are able to enter and leave the problem between plans, meaning the optimization is only ever looking at a ‘window’ of the total number of aircraft in the global problem. This is especially well suited to the continuous nature of traffic at airports, with aircraft continuously arriving and departing. Aircraft are considered to be ‘active’ in the optimization for a horizon if they are already taxiing or their earliest possible taxi start time is within the execution horizon. When the aircraft take off, they are removed from the active set before the next horizon is optimized.

The planning solution for each horizon is provided by an optimizer which iterates on the conflict prevention constraints, this idea is similar to that presented by Earl and d’Andrea [24]. Initially no constraints are included to prevent conflict on the airport surface. Once the problem is solved, the solution is checked for any conflicts between aircraft. If conflicts are detected, constraints to
prevent these are added to the problem and it is re-solved. This is repeated until the solution is conflict free, as shown by the system diagram in Fig. 1. The formulation of this iterative optimizer is described in the remainder of this section. A simulator is used to predict the new state of each aircraft at the end of the current execution horizon, defining the inputs for the optimization at the next planning step. In this paper the simulator assumes that all aircraft behave perfectly, and no variability is introduced. The RH formulation introduces feedback, meaning that the method can in principal compensate for uncertainty.

2.2 Input parameters

2.2.1 Airport

The airport surface is modelled as a graph (see Fig. 3) containing a set of $N_n$ nodes denoted $\mathcal{N}$, each representing a junction or intersection of taxiways. A base binary connectivity matrix $C$ is defined such that $C(n,m) = 1$ if and only if there is a direct taxiway connection from node $n$ to node $m$ on the airport surface. A similarly defined aircraft specific connectivity matrix $C_a$ allows for variations in allowable taxiways between aircraft. The distance matrix $L$ is defined such that $L(n,m)$ represents the distance along the taxiway between adjacent nodes $n$ and $m$. The shortest taxiway distance from node $n$ to any other node $m$ is defined as $r_{ST}(n,m)$. This quantity is pre-calculated outside the optimization using a rapid graph search algorithm. A set of ‘gate entry’ nodes $\mathcal{G} \subset \{n : n \in \mathcal{N}\}$ are defined, these represent gateways beyond which are apron areas with the parking positions: we do not model the apron area in detail. Designate the departure runway node as $n_{Rd} \in \mathcal{N}$ and the arrival runway node as $n_{Ra} \in \mathcal{N}$. At present mixed mode runways (including the runway crossing mode) are not being considered.
2.2.2 Aircraft

The set $\mathcal{A}$ contains all $N_a$ active aircraft in the problem, as explained, those which can taxi within the execution horizon. $\mathcal{A}$ is composed of an active arriving aircraft set $\mathcal{A}_{\text{arr}} \subset \mathcal{A}$, and an active departing aircraft set $\mathcal{A}_{\text{dep}} \subset \mathcal{A}$ where $\mathcal{A}_{\text{arr}} = \mathcal{A} \setminus (\mathcal{A}_{\text{dep}})$. Active aircraft are modelled as points moving along the arcs, subject to a maximum speed limit $V(a)$. Each active aircraft $a \in \mathcal{A}$ is required to move from its specified origin node $n_0(a) \in \mathcal{N}$ for the horizon to its final destination node $n_D(a) \in \mathcal{N}$. No aircraft is permitted to move until after its origin time $t_0(a)$. In their initial horizon this is earliest push-back for departing aircraft and landing time for arriving aircraft, whereas in any subsequent horizon the horizon start time is used. Consideration is not given in this paper to scheduling of the arrival runway, hence the landing times of arriving aircraft are currently assumed to be fixed inputs.

2.2.3 Separation Rules

The matrix $d_t \in \mathbb{R}^{N_a \times N_a \times N_n}$ encodes the temporal separation limits during taxiing and on the departure runway, such that the time from aircraft $a$ to that of aircraft $b$ at a node $n$ must be no less than $d_t(a, b, n)$. For the runway this includes wake vortex separation limits and the downstream Standard Instrument Departure (SID) routes separation requirements [21]. Some work [12, 27] on the timing of predetermined routes have encoded the temporal separation as a variable, depending on the speeds of the aircraft to be separated. In this paper we consider it to depend only on the weight classes of the aircraft, their planned flight paths, and whether the node in question is a runway.
2.3 Decision Variables

The optimization divides the movement of all aircraft into a sequence of $N_k$ moves. Note that the actual time length of these vary from one planning period to another, and also between aircraft, so moves are not required to be synchronised. This variability means that in a planning horizon which consists of $N_k$ planning periods, each aircraft’s plan is for a different length of time. This introduces challenges in setting an execution horizon that is lower than the planning horizons of each aircraft, in this paper we consider a small fixed execution horizon of 40 seconds, although investigation of variable execution horizons is of interest for future work.

The variable $R \in \mathbb{R}^{N_a \times N_k}$ defines the routing of each aircraft $a$. $R(a, k)$ is the node at which aircraft $a$ begins its $k$’th move. The second key decision variable $T \in \mathbb{R}^{N_a \times N_k}$ is then defined such that $T(a, k)$ is the time at which aircraft $a$ starts its $k$’th move. Together, $R$ and $T$ completely specify each aircraft’s motion from start to finish.

2.4 Constraints

2.4.1 Initial Conditions

Constraints on the initial planning step of the optimization are:

$$R(a, 1) = n_0(a) \quad \forall a \in A$$

(1)

$$T(a, 1) \geq t_0(a) \quad \forall a \in A$$

(2)

Equation (1) ensures that every aircraft’s initial position is its origin node for the current horizon. Equation (2) ensures that the all aircraft do not plan to begin moving before their origin time in any given horizon. As we are assuming fixed landing times, Equation (3) is needed to ensure
that the arriving aircraft comply with their given origin time in their initial horizon:

\[
\forall a \in A_{\text{arr}} \quad T(a, 1) = t_0(a) \tag{3}
\]

Note that terminal conditions, defining where the aircraft should end up and when, are not enforced as constraints. Instead, these are represented by the terms of the objective function explained later in this section.

### 2.4.2 Virtual Nodes

The formulation requires all aircraft \( a \) to have an origin node, \( n_0(a) \) within each horizon. As within the RH environment aircraft may end an execution horizon between two nodes \( n \) and \( m \) (see Fig. 2-A), virtual nodes \( n_v(a) \) are employed. These are nodes which are temporarily introduced into the airport topology model at the terminal locations of aircraft from the preceding execution horizon, as in Fig. 2-B. The virtual node \( n_v(a) \) introduced by aircraft \( a \) is then its origin node \( n_0(a) \) in the next planning horizon.

The aircraft specific connectivity matrices, \( C_a(n, m) \) incorporate the virtual nodes in each horizon, before being restored to the base matrix \( C(n, m) \) after the optimization. In order to prevent reversing immediately occurring in the next horizons plan, the link directly behind each aircraft \( a \) is considered to be one-way in the individual connectivity matrix \( C_a(n, m) \) of that aircraft, as shown in Fig. 2-C.

In some cases a virtual node is very close to the next node, as demonstrated by the distance \( f \) from node \( m \) in Fig. 2-D. It is considered that any aircraft at a virtual node \( n_v \) within this distance
Figure 2: Virtual Nodes: A) A case where a virtual node is required B) How the virtual node amends the graph C) Directional arcs to prevent immediate reversing in proceeding horizon D) Virtual node placement too close to original node causing a 'blockage'

would block the original node $m$. In this situation the aircraft is considered to have carried out its original plan through to the next planned node $m$, and this is set as the origin $n_0$ in the next horizon with it’s earliest movement time $t_0$ restricted to be after the planned time at node $m$ from the previous horizon.

2.4.3 Routing Constraints

The following constraint ensures that the routes recorded by the variable $R$ are valid for the taxiway structures defined in connectivity matrices $C_a$.

$$C_a(R(a, k - 1), R(a, k)) = 1 \quad \forall a \in A, k \in \{2, \ldots, N_k\} \quad (4)$$

Note all nodes apart from destination nodes are not considered to be linked to themselves, preventing repeated nodes in the plan. Repeats of the destination node are allowed if required to fill the planning periods, the associated times are constrained by Equation (5) to being the same, preventing any confusion for runway scheduling constraints. Doubling back, for example the route
\( n, m, n \), is also not permitted as it is assumed that the aircraft can only travel forwards. This ‘reversing’ can occasionally become a favourable option in order to alter the take off order and therefore is constrained against by Equation (6).

\[
\forall a \in \mathcal{A}, j \in \{2, \ldots, N_k - 1\}, R(a, j) = R(a, N_k) :
T(a, N_k) \leq T(a, j)
\] (5)

\[
\forall a \in \mathcal{A}, j, k \in \{2, \ldots, N_k - 1\}, j \neq k, R(a, k) = R(a, j) :
R(a, (k + 1)) \neq R(a, (j - 1))
\] (6)

### 2.4.4 Taxi Timing Constraints: Speed

The following constraint links the routing and timing variables, ensuring that the movements are consistent with the aircraft speed capabilities,

\[
\forall a \in \mathcal{A}, k \in \{2, \ldots, N_k - 1\} :
T(a, k) + \frac{L(R(a, k), R(a, k + 1))}{V(a)} \leq T(a, k + 1)
\] (7)

in which \( \frac{L(n, m)}{V(a)} \) represents the minimum time in which aircraft \( a \) can move from node \( n \) to node \( m \).

### 2.4.5 Taxi Timing Constraints: Conflict

This subsection describes the constraints on movement timing which ensure aircraft conflicts are prevented. The number of possible conflicts is high in problems of interest \( (N_a(N_a - 1)N_n) \), and as
a result fully constrained problems are computationally demanding. However the number of actual conflicts which would occur if not constrained against is typically much lower. Conflict constraints are therefore well suited to an iterative framework which aims to improve solution efficiency by solving a series of less-constrained problems instead of the full problem [24].

\[ SE = \{ (a, b, n) : a, b \in 1 \ldots N_n, a \neq b, n \in N \} \]

is a set of all instances of possible separation conflict between two aircraft \( a \) and \( b \) at a node \( n \). Similarly \( CO = \{ (a, b, n, m) : a, b \in 1 \ldots N_n, a \neq b, n, m \in N, n \neq m \} \) contains the lists of all possible crossover (overtaking and head-on) conflicts between two aircraft \( a \) and \( b \) between two nodes \( n \) and \( m \).

Subsets of these: \( SE_i \subseteq SE \) and \( CO_i \subseteq CO \) are needed for the iterative procedure, these are augmented from one iteration, \( i \) to the next, \( i + 1 \). The conflict constraints (11-17) are initially fully relaxed by including no instances of possible conflict in the subsets:

\[ SE_0 = CO_0 = \emptyset \] 

Note for solution of the fully constrained problem it is considered that in the initial iteration the subset represents the full set, \( SE_0 = SE \) and \( CO_0 = CO \). After each iterations MILP solution the plan is checked for conflicts, if any occur they are added to the detected sets for that iteration \( SE_{det}(i) \) or \( CO_{det}(i) \) which are then used to augment the subsets, using Equations (9-10), and the problem is re-solved. This process is repeated until no conflicts are present in the solution.

\[ SE_i = SE_{i-1} \cup SE_{det}(i - 1) \] 
\[ CO_i = CO_{i-1} \cup CO_{det}(i - 1) \] 

This iterative procedure still results in the globally optimal solution whilst reducing computational
effort by avoiding the application of redundant constraints. The rationale behind this comes from
the non-linear increase in solve times with problem size, meaning solving four problems of size 1
can be quicker than solving one problem of size 4.

Separation is required between an aircraft and another using the same intersection on the
taxiway, when this type of conflict is detected it is generally enforced at each node by (11) or (12). The MILP implementation of this choice is detailed in Section 3.

\[ \forall j, k \in \{1, \ldots, N_k - 1\}, \{a, b, n\} \in SE_i : \]

\[ T(a, j) \geq T(b, k) + d_t(b, a, n) \quad \text{or} \quad (11) \]
\[ T(a, j) \leq T(b, k) - d_t(a, b, n) \quad (12) \]

Note that a ‘first is first’ separation constraint is also needed to ensure that aircraft already
taxiing at the beginning of the problem horizon are the first to visit their respective origin nodes. Equations (13) and (14) enforce this special case separation constraint at origin nodes.

\[ \forall j, k \in \{1, \ldots, N_k - 1\}, \{a, b, n\} \in SE_i : \]

\[ \{n = n_0(a)\} \implies T(a, 1) \geq T(b, k) \quad (13) \]
\[ \{n = n_0(b)\} \implies T(a, j) \leq T(b, 1) \quad (14) \]

It is also important in the Receding Horizon framework to ensure that an aircraft planning to
use a given node in the current horizon is adequately separated from aircraft that used the same
node in the previous horizon. This is accounted for by Equation (15):

\[ \forall a \in \{1, \ldots, N_a \}, k \in \{1, \ldots, N_k \} : \]

\[ T(a, k) > T_L(R(a, k)) + d_t(a, a_L(R(a, k)), R(a, k)) \]  \hspace{1cm} (15)

where \( T_L(n) \) is the last time that node \( n \) was visited prior to the current horizon, and \( a_L(n) \) is the last aircraft to use the node \( n \). Since separation is only enforced at nodes, it is necessary to be able to constrain against crossover conflicts occurring on the arcs in between nodes, such as overtaking and head on situations. Equations (16)-(17) constrain against detected conflicts of this type.

\[ \forall k, j \in \{1, \ldots, N_k - 1\}, \{a, b, n, m\} \in \mathcal{C}_i : \]

\[ \begin{cases} T(a, j) \geq T(b, k) + d_t(b, a, n) \quad \text{and} \\ T(a, j + 1) \geq T(b, k + 1) + d_t(b, a, m) \end{cases} \]  \hspace{1cm} (16)

or

\[ \begin{cases} T(a, j) \leq T(b, k) - d_t(a, b, n) \quad \text{and} \\ T(a, j + 1) \leq T(b, k + 1) - d_t(a, b, m) \end{cases} \]  \hspace{1cm} (17)

2.4.6 Cost-to-go

A cost-to-go is used to represent the approximate plan beyond the detailed planning horizon discussed in Section 2.1. This is based on the shortest taxiway distance, \( r_{ST}(n_F, n_D) \), from the terminal node \( n_F \) of the detailed plan to the destination node \( n_D \) (runway or gate entry), traversed
at the maximum speed of the aircraft, $V_a$.

$$\forall a \in A :$$

$$t_D(a) = T(a, N_k) + \frac{r_{ST}(n_F(a), n_D(a))}{V_a}$$

Equation (18) ensures that the auxiliary variable $t_D(a)$, encoding the absolute planned time at which aircraft $a$ reaches its destination node, complies with the approximate plan from the planned terminal position.

### 2.4.7 Runway Scheduling

Studies have shown that the runway is a major bottle neck in the airport surface [16], and Atkin at al [6] identified that the downstream separation constraints beyond the runway were of great importance. This motivates the inclusion of runway scheduling in a ground movement optimizer.

As mentioned already, arrival runway scheduling is not being considered in this paper, and the predetermined landing times and order are fixed. This leaves the departure runway scheduling. As the aircraft are not guaranteed to reach their destination in the planning horizon the runway scheduling has to be dealt with in a different way to the general separation. Each aircraft $a$ has an absolute planned destination (or take-off for departures) time $t_D(a)$ which is used in (19) or (20).
to create a valid schedule which meets the separation constraints.

\[
\forall a, b \in \mathcal{A}_{dep} \cup \mathcal{A}_x : \\
\begin{align*}
    t_D(a) &\geq t_D(b) + d_t(b, a, n_R) \quad \text{or} \\
    t_D(a) &\leq t_D(b) - d_t(a, b, n_R)
\end{align*}
\]  

where the separation limits on the runway, represented by \(d_t\), include both the wake vortex and downstream SID separation requirements. Also \(\mathcal{A}_x\) is a set of auxiliary aircraft, a concept discussed in the next section. Note that the last take off time from executed horizons is also recorded and used to constrain the first take off in the current plan.

2.4.8 Departure Runway Planning Foresight

As discussed, it is natural to operate the RH planner in a ‘Rolling Window’ fashion, considering only a subset of the aircraft. Under this formulation it was observed that the introduction of new aircraft could cause considerable shuffling in the take-off queue. This leads to undesirable behaviour, as taxiing aircraft may slow down considerably, lengthening the taxi time, and also increasing the need for acceleration and deceleration. This effect is similar to ‘churning’ seen in other assignment and routing optimizers [28].

In order to combat this effect, the scheduler is allowed greater ‘foresight’, in terms of departing aircraft. The set, \(\mathcal{A}_x\) of auxiliary departure aircraft, \(a_x\) is defined. These aircraft are those whose earliest push-back time \(t_0(a_x)\) is within the foresight horizon \(T_f\) beyond the current execution horizon. This set of aircraft is not included in the routing optimizer, and they are used for the
purposes of runway scheduling only. Auxiliary aircraft do not have a detailed plan: instead the shortest path approximation is used to determine the time at which they are expected to start taxiing $t_{\text{start}}$, and a take off time on the runway $t_D$ only.

The lists of departing aircraft, $A_{\text{dep}}$ and auxiliary input aircraft, $A_x$ are combined for use in (20). The following equations are the equivalent to (18) for auxiliary aircraft, the distinction being the use of a variable value of $t_{\text{start}}$. The shortest taxiway distance to the destination node $r_{ST}(n_0(a_x), n_D(a_x))$ for auxiliary input aircraft is also included, measured from the origin node, $n_0$ due to their lack of detailed plan.

$$\forall a_x \in A_x :$$
$$t_D(a_x) = t_{\text{start}}(a_x) + \frac{r_{ST}(n_0(a_x), n_D(a_x))}{V_{a_x}}$$

$$t_{\text{start}}(a_x) \geq t_o(a_x)$$

Equation (21) ensures that estimated take off times comply with the rough plan to the runway from the origin position. Equation (22) ensures that auxiliary aircraft $a_x$ does not plan to start taxiing before its earliest push-back time. The foresight is set to 300 seconds, preliminary experiments suggested that a foresight horizon of around 400 seconds could offer a reduction in the overall taxied time of approximately 2% over the case where the optimizer has no foresight. Larger values of foresight inevitably result in higher computation times, hence 300 seconds was used in this case, reducing computation whilst still accessing the benefits of foresight. Readers are directed to [26] for thorough discussion.
2.5 Objective Function

The objective that is minimized in a given horizon, $h$, is a weighted combination of the last time on the take-off runway amongst all active and auxiliary aircraft, the total taxi time of all the active aircraft and their total taxi distance,

$$J_h = w_0 t_{\text{end}}$$

$$+ w_1 \sum_{a=1}^{N_a} (t_D(a) - t_{\text{start}}(a))$$

$$+ w_2 \sum_{a=1}^{N_a} \sum_{k=2}^{N_k} L(R(a, k-1), R(a, k)) + r_{ST}(n_F(a), n_D(a))$$

The coefficients $w_0$, $w_1$ and $w_2$ are non-negative weightings with initial values in this paper of 25, 5 and 3 respectively, and $t_{\text{end}}$ represents the last time on the take off runway motivating throughput.

$$t_{\text{end}} \geq T_D(a) \quad \forall a \in \mathcal{A} \cup \mathcal{A}_x$$

The absolute time that aircraft $a$ starts to taxi is represented by $t_{\text{start}}(a)$. This value is fixed for aircraft already taxiing at the beginning of the problem, but variable for those who have yet to begin taxiing, being equal to $T(a, 1)$ in this case. Beyond the horizon, the cost-to-go assumes the aircraft use the shortest possible path to the destination $r_{ST}(n_F(a), n_D)$, where $n_F(a)$ is the final node in aircraft $a$’s detailed plan.

The inclusion of the approximation values beyond the detailed plan acts as a terminal penalty. Also aircraft are prevented from ‘putting off’ making a detailed plan in favour of a rough repre-
sentation of their whole plan by only allowing destination nodes to be repeated in the detailed plan.

3 MILP Implementation

3.1 Introduction

This Section describes the MILP formulation of the constraints presented in Section 2. A linear program is a mathematical formulation of a minimization or maximization of a linear function, $f(x)$, subject to linear constraints. A Mixed Integer Linear Programming (MILP) problem is a subset of linear programs in which a specified subset of the variables are required to take on integer values as in (25), meaning that both discrete and continuous decisions can be represented. Solving such problems is NP-hard.

\[
\text{Min : } f(x) = c^T x + d^T y \\
\text{Subject to : } \\
Ax + By = b \\
Gx + Fy \leq g \\
x \in \mathbb{Z}
\]  

(25)

Representation of the taxiway routing and runway scheduling problem defined in Section 2 as a MILP problem requires the introduction of binary decision variables, and reformulation of constraints. The remainder of this Section presents these amendments.
3.2 Binary Decision Variables

The routing of each aircraft, previously described by the $R$ variable is encoded within the MILP by the binary decision variable $X \in \{0, 1\}^{N_a \times N_n \times N_n \times N_k}$. It is defined such that $X(a, n, m, k) = 1$ if and only if aircraft $a$ is routed from node $n$ to node $m$ during planning period $k$, this draws on the work of Marin [14]. The previously described decision variable representing timing of the plans $T$ is retained. In addition to $X$ and $T$, several auxiliary decision variables are necessary to formulate the problem in MILP form:

- Binary variable $g_{SE}(\{a,b,n\}) = 1$ if and only if aircraft $a$ is routed before aircraft $b$ into node $n$, $\forall\{a,b,n\} \in SE_i$ where $SE_i \subseteq SE$.

- Binary variable $g_{CO}(\{a,b,n,m\}) = 1$ if and only if aircraft $a$ is routed before aircraft $b$ through the arc between nodes $n$ and $m$, $\forall\{a,b,n,m\} \in CO_i$ where $CO_i \subseteq CO$.

- Binary variable $g_{RO}(a,b) = 1$ if and only if aircraft $a$ is routed before aircraft $b$ through the departure runway node, $\forall a,b \in A_{dep}$.

- Binary variable $D_{SE}(\{a,b,n\}, k, j) = 1$ if and only if aircraft $a$ is routed into node $n$ in planning period $j$ and aircraft $b$ is routed into node $n$ in planning period $k$, $\forall\{a,b,n\} \in SE_i$.

- Binary variable $D_{OT}(\{a,b,n,m\}, k, j) = 1$ if and only if aircraft $a$ and $b$ are routed from node $n$ to node $m$ in respective planning periods $j$ and $k$, $\forall\{a,b,n,m\} \in OT_i$.

- Binary variable $D_{HO}(\{a,b,n,m\}, k, j) = 1$ if and only if aircraft $a$ is routed from node $n$ to node $m$ in planning period $j$ and aircraft $b$ is routed from node $m$ to node $n$ in planning period $k$, $\forall\{a,b,n,m\} \in HO_i$. 

23
Binary variables $D_{SE}$, $D_{OT}$ and $D_{HO}$ are products of the binary variable $X$ as shown by Equations (26 - 28). Note $D_{OT}$ and $D_{HO}$ represent possible crossover conflicts as two aircraft plans share an arc, $D_{SE}$ represents possible separation conflict as two aircraft plans share a node.

∀$j,k \in \{1, \ldots, (N_k - 1)\}$ : 

$$D_{SE}(\{a,b,n\}, k, j) = \sum_{m=1}^{N_n} X(a,m,n,j) \sum_{m=1}^{N_n} X(b,m,n,k) \quad \forall\{a,b,n\} \in SE_i$$ (26)

$$D_{OT}(\{a,b,n,m\}, k, j) = X(a,m,n,j)X(b,m,n,k) \quad \forall\{a,b,n,m\} \in OT_i$$ (27)

$$D_{HO}(\{a,b,n,m\}, k, j) = X(a,m,n,j)X(b,n,m,k) \quad \forall\{a,b,n,m\} \in HO_i$$ (28)

These binary products are converted into linear inequalities using the method developed by Bemporad and Morari in Ref. 17. This allows the product of two binary variables, $\delta_1$ and $\delta_2$, to be represented by a third binary variable, $\delta_3$, via a series of linear inequalities as described by Equation 29.

$$\delta_3 = \delta_1 \delta_2 \quad \text{is equivalent to} \quad \begin{cases} -\delta_1 + \delta_3 \leq 0 \\ -\delta_2 + \delta_3 \leq 0 \\ \delta_1 + \delta_2 - \delta_3 \leq 1 \end{cases}$$ (29)

### 3.3 Constraint Formulations

Constraints which previously required the use of the routing variable $R$ are reformulated for use in the MILP with the binary variable $X$. These reformulations and additional constraints required to maintain integrity are presented in the remainder of this section. Within the MILP formulation the first planning period is a ‘dummy’ step, this is due to the formulation needs of some of the
constraints which require summations in the binary routing variable $X$. This first step is used to enforce the initial position and time constraints:

$$X(a, n_0(a), n_0(a), 1) = 1 \quad \forall a \in A \quad (30)$$

$$T(a, 1) = T(a, 2) \quad \forall a \in A \quad (31)$$

The following constraints are introduced to ensure that the variable $X$ encodes a valid routing through the taxiways:

$$\forall a \in A, n \in N :$$

$$X(a, n, m, k) \leq C(n, m) \quad \forall m \in N, k \in \{2, \ldots, N_k - 1\} \quad (32)$$

$$\sum_{m=1}^{N_n} X(a, n, m, k) = \sum_{m=1}^{N_n} X(a, m, n, k - 1) \quad \forall k \in \{2, \ldots, N_k - 1\} \quad (33)$$

$$\sum_{n=1}^{N_n} \sum_{m=1}^{N_n} X(a, n, m, k) = 1 \quad \forall k \in \{1, \ldots, N_k - 1\} \quad (34)$$

Equation (32) constrains the aircraft’s movements to follow the rules of connectivity of the graph structure. Continuity is assured by Equation (33), meaning that if an aircraft moves to node $n$ in period $k - 1$, it must move from node $n$ in period $k$. Equation (34) forces each aircraft to have a plan for each period.

Equations (5) and (6) are respectively emulated in MILP by the following Equations:
∀a ∈ A, j ∈ \{2, \ldots, N_k - 1\} :
\[ T(a, N_k) \leq T(a, j) + M \cdot (1 - \sum_{m=1}^{N_n} X(a, m, nD(a), (j - 1))) \] (35)

∀a ∈ A, n ∈ \{1, \ldots, N_n - 1\}, m ∈ \{(n + 1), \ldots, N_n\} :
\[ \sum_{k=1}^{(N_k - 1)} X(a, n, m, k) + \sum_{j=1}^{(N_k - 1)} X(a, m, n, j) \leq 1 \] (36)

where \( M \) is an arbitrary large number greater than the largest scale in the problem. The speed constraint is translated into that shown in Equation (37)

∀a ∈ A, k ∈ \{2, \ldots, N_k - 1\} :
\[ T(a, k) + \frac{\sum_{n=1}^{N_n} \sum_{m=1}^{N_n} L(n, m) X(a, n, m, k)}{V(a)} \leq T(a, k + 1) \] (37)

The iterated constraints on movement timing which ensure aircraft conflicts are prevented are reformulated as follows:

∀\{a, b, n\} ∈ S\mathcal{E}_i, n ≠ n_0(a) \cup n_0(b), k ∈ \{2, \ldots, N_k - 1\} :
\[ g_{SE}(\{a, b, n\}) \geq -M (1 - D_{SE}(\{a, b, n\}, j, 1)) + 0.5 \] (38)
\[ g_{SE}(\{a, b, n\}) \leq +M (1 - D_{SE}(\{a, b, n\}, 1, k)) + 0.5 \] (39)

Equations (38) and (39) represent the first is first rule, ensuring that if a node, apart from the
arrival runway node \( n_{Ra} \), is the first node in a taxiing aircraft’s planned route, then that aircraft is the first aircraft to visit that node.

\[
\forall \{a, b, n\} \in SE, k \in \{1, \ldots, N_k - 1\}, j \in \{1, \ldots, N_k - 1\} : \\
D_{SE}(\{a, b, n\}, k, j) = 1 \implies \\
T(a, (j + 1)) \geq T(b, (k + 1)) + d_t(b, a, n) - M_{S_E}(\{a, b, n\}) \\
T(b, (k + 1)) \geq T(a, (j + 1)) + d_t(a, b, n) - M(1 - g_{SE}(\{a, b, n\}))
\]

where \( \implies \) is the ‘implies’ operator which can be efficiently formulated for CPLEX optimization (see Implementation Section below) via indicator constraints. By identifying a binary variable to control the activity of specified linear constraints, indicator constraints can replace some Big-M formulations creating more numerically robust and accurate problems. Experience suggests significant computational improvements as a result of the use of indicator constraints. The scheduling constraints on the departure runway are reformulated in a similar way to the separation constraints by using the binary runway ordering switch \( g_{RO}(a, b) \). Equations (40) and (41) formulate the separation constraints defined in Equations (11) and (12).

\[
\forall a \in A, n, m \in N, k \in \{3, \ldots, N_k - 1\} : \\
X(a, n, m, k) = 1 \implies T(a, k) \geq T_L(n) + d_t(a, a_L(n), n)
\]
∀a ∈ A_{dep} : \begin{align*}
\text{if } T_t > 0 \text{ then } & T(a, 1) \geq T_L(n_0(a)) + d_t(a, a_L(n_0(a)), n_0(a)) \\
\text{if } T_t = 0 \text{ then } & T(a, 1) \geq 0
\end{align*} (43)

The consideration of aircraft from previous horizons demonstrated by Equation (15), is split into the two Equations (42-43), respectively dealing with the departure runway / taxiways and the aircraft departing their gates. $T_t$ is the time an aircraft has already spent taxiing prior to the current horizon, and indicates whether it is leaving from the gate, or a position already on the taxiways. The choice in Equation (43) can be formulated as $T_t$ is an input parameter.

∀\{a, b, n, m\} ∈ CO, k ∈ \{1, \ldots, N_k - 1\}, j ∈ \{1, \ldots, N_k - 1\} :

\begin{align*}
D_{OT}(\{a, b, n, m\}, k, j) \cup D_{HO}(\{a, b, n, m\}, k, j) = 1 \implies \\
T(a, j) + d_t(a, b, n) & \leq T(b, k) + M(1 - g_{CO}(\{a, b, n, m\})) \quad (44) \\
T(a, (j + 1)) + d_t(a, b, m) & \leq T(b, (k + 1)) + M(1 - g_{CO}(\{a, b, n, m\})) \quad (45) \\
T(b, k) + d_t(b, a, n) & \leq T(a, j) + M_{gCO}(\{a, b, n, m\}) \quad (46) \\
T(b, (k + 1)) + d_t(b, a, m) & \leq T(a, (j + 1)) + M_{gCO}(\{a, b, n, m\}) \quad (47)
\end{align*}

Equations (44 - 47) prevent overtaking on the taxiways and head-on conflicts on the taxiways,
emulating Equations (16) and (17). To summarize the objective is formulated as:

\[ J_h = w_0 t_{\text{end}} \]

\[ + w_1 \sum_{a=1}^{N_a} \left( t_D(a) - t_{\text{start}}(a) \right) \]

\[ + w_2 \sum_{a=1}^{N_a} \sum_{k=2}^{N_k} \sum_{n=1}^{N_n} \sum_{m=1}^{N_n} L(n,m)X(a,n,m,k) + r_{ST}(m,a)X(a,n,m,(N_k - 1)) \]

\[ \text{Distance travelled by } a \]

this is to be minimized subject to Equations (2,3,18,21,22, and 30-47).

4 Results

4.1 Implementation

The optimization was translated into the AMPL modelling language [29]. An AMPL model file contains the constraint forms for all instances, while the data is written to an AMPL data file by a Matlab script. CPLEX 11.1.0 parallel optimization software [19] is used on a 3GHz dual core PC with 2GB of RAM to solve all problems.

4.2 Large-Scale Heathrow Problem

4.2.1 Problem Set-Up

To illustrate the potential of the approach the baseline optimization scheme developed was used on a large-scale problem using the Heathrow layout represented by the 126-node graph structure shown in Fig. 3(a). Basic departure and arrival board data collected from the Heathrow website [30] on Wednesday 26th August 2009 between 9 am and 12 noon, was used. This included 240 aircraft,
Figure 3: Time frames of RH large-scale problem solution
122 of which were arrivals. Gate assignments, SID routes and aircraft weight classes have been assigned arbitrarily as these details were not publicly available. The graph structure used does not incorporate Terminal 4, as very few flights used this terminal. Terminal 4 arrivals are assumed to taxi south of the arrival runway and hence do not form part of the problem. Terminal 4 departures were assumed to enter the problem at the bottom right-hand node, having already passed to the East of the arrival runway.

The RH framework had a planning horizon of $N_k = 6$ planning periods with replanning occurring every 40 seconds. The departure foresight window used was 300 seconds. The weightings used in the objective (48), were set as follows: $w_0 = 25$, $w_1 = 5$, $w_3 = 3$.

4.2.2 FCFS Comparisons and Analysis

The problem described above was solved using both the RH optimization scheme and a FCFS approach. The resulting RH plan was found to improve upon FCFS by 55% when comparing the total taxiing time across all aircraft, going from an average of 13.2 minutes per aircraft to 5.89 minutes per aircraft. Fig. 4 shows the direct comparison of taxi times for each arrival and departure aircraft in the problem. This shows a distinct difference between arrivals and departures, indicating the MILP optimization offers little improvement over FCFS for arrivals, whereas significant improvements can be made in the departure plans. This may be due to the lack of downstream constraints on arrivals as actions beyond the gate entry are not considered. It is also noted that many arrivals had short journeys to Terminal 5, which were largely unimpeded by departure aircraft flow, and that due to the layout of Heathrow, all aircraft follow the same general circulation pattern across the surface (North-East in the example presented). Although slight, the deviations that some arrivals do show from FCFS indicate interactions with departures, with the slightly longer taxi times
suggesting they may have accepted some delay to increase global gains.

It was observed that some departure aircraft wait at their gate in order to begin their taxi later. This was seen to be a good strategy in recent work by Simaiakis and Balakrishnan [31] in which it was demonstrated that a saturation limit to the number of departure aircraft taxiing exists. Below the limit, increasing the number of departing aircraft increases take off rate; beyond it, the runway becomes the defining capacity constraint, and increasing the number of departure aircraft simply adds to congestion on the surface. However, in the solution produced by the RH scheme, 10% of the departure aircraft and 50% of the arrival aircraft also took routes which were different to the shortest paths used in FCFS, and as the simple demonstrations in our previous work [32] showed, allowing re-routing can be beneficial to the problem solution.

Fig. 5(a) shows a comparison of the departure aircraft take-off times between the FCFS and RH solutions, with the dashed, light and heavier lines respectively highlighting an earlier take-off, no change, and a later take-off under the RH scheme. It can be seen from Fig. 5(a) that the majority of aircraft have earlier take-off times under the RH optimization. Also, although considerable reordering takes place, it occurs in a relatively localised fashion, meaning no one
Figure 5: Take-off Schedule Comparisons between RH and FCFS
aircraft is unreasonably delayed for the global good. By observing the change in the distribution of separation times between take offs, shown in Fig. 5(b), it is clear that the throughput capacity of the departure runway would be improved by the RH scheme, as it creates a 17 minute gap in runway usage in which up to 17 additional aircraft could potentially take off.

Stills from the RH solution are shown in Fig. 3. Within these shaded aircraft represent departures and unshaded arrivals. Interactions between the arrival and departure paths around the airport surface can be seen clearly in still (a), still (b) and still (c). Note that in still (c), departure aircraft number 107 departs from the gate in a timely fashion allowing it to slip into the departure queue between aircraft number 90 and aircraft number 88, as shown by still (d). The departure runway order and spacing are seen to be well formed on the taxiways in still (e) and still (f).

The complete simulation took just under 4 hours to run. Fig. 6 shows the solve times of each individual horizon within the problem, from this it can be seen that all of the horizons solved within 120 seconds with many horizons solving in under a minute. Although slightly slower than
real time, as the replanning occurs every 40 seconds, the difference is small enough to suggest that with increased computing power, real time operation will be practical. The issue of computational efficiency is discussed further in the next section.

4.3 Computation

In order to investigate the efficiency of the baseline algorithm presented here, we consider the effect of moving away from it by first the removing the RH framework and then also removing the constraint iteration.

Solving airport problems of the scale of that described in Section 4.2 without a receding horizon means expecting all aircraft to plan their entire route, to allow computation of the global optimal. This would require a longer planning horizon of around 30 planning periods, as the largest value of $R_{ST}$ from gate to runway in the graph model of Heathrow is 25 nodes in length. Even considering a rolling window in aircraft memory issues become a problem. In a typical day more than 1200 aircraft pass through Heathrow [30]. The RH element of the baseline scheme is necessary to allow solution of problems of this scale, although it does introduce some approximations into the solution.

We have found that it is possible, without RH, to solve small instances of the taxi problem on a small section of the Heathrow lay-out for up to 8 aircraft taxiing simultaneously where the aircraft only need to plan up to 10 moves. Therefore, comparison tests were completed using a 19-node section representing the area around the holding point of Heathrow’s North runway. Only departing aircraft were considered and 64 different instances of a 3 aircraft problem were posed. The same was done for cases involving 4, 5, 6, 7 and 8 aircraft. The weights used in the objective function for these problems were also: $w_0 = 25$, $w_1 = 5$ and $w_2 = 3$.

The problems were solved using the baseline algorithm: as presented - ‘RH’; without the RH
Figure 7: Solve Time Comparison of 3, 4, 5, 6, 7 and 8 aircraft problems, 64 permutations of each, solved using Fully Constrained, Iterative, and Receding-Horizon methods framework - ‘Iterative’; and without RH or constraint iteration - ‘Fully Constrained’. Fig. 7 shows the solve times of these problems, including both the full problem solved by the baseline RH scheme, ‘RH - Total Problem’, and a single horizon within that, ‘RH - Single Horizon’. Note the lack of results for the fully constrained case for problems involving more than 5 aircraft. Beyond this the algorithm consistently ran out of memory to complete the optimization, or reached the 10 hour restriction on solve time without finding an integer solution.

The results show that constraint iteration reduces the solve time considerably, although the solve times still increase at an exponential rate with respect to problem size. The spread of Iterative solve times also increases significantly with problem size, consistent with the results of previous works [24]. The more constraints that have to be added back into the problem, the longer the iterative method takes. However in all cases presented here the solve time saving is significant due
to the large number of inactive constraints, and no approximations are introduced so the solution is still the global optimal.

Considering the RH framework, it can be seen that the single horizon computation times are at the lowest end of the range of Iterative solve times if not below. The solve time saving available from adopting the RH approach becomes more pronounced as the number of aircraft in the problem increases. The ‘RH - Total Problem’ solve time is a summation of the solve times of the individual horizons needed to complete the entire problem. This shows much less variability than iterative scheme for larger problems and is observed to be much more scalable in the number of aircraft, being almost linear, suggesting that for bigger problems the computational saving will become even more significant. The ability of the RH, rolling window scheme to spread the computation between horizons, and hence over time, also becomes more significant in larger problems which span longer periods of time on the airport surface (≥ 1 hour), allowing the optimizer to concentrate on the near future, using approximations for the distant future.

4.4 Performance

As already stated constraint iteration does not effect the solution quality, meaning that the Iterative solution is identical to that of the Fully-Constrained problem. However, the introduction of the RH framework results in a plan which is an approximation to the global optimal solution. Figure 8 shows comparisons of RH solution performance to both the non-RH (Iterative/Fully Constrained) and FCFS with points representing each of the problems described above and a line indicating the 1:1. Figures 8(a) and 8(b) show comparisons of the total taxi time performance of RH against Iterative/Fully Constrained and FCFS respectively. From these figures it can be seen that, as would be expected, in general the RH-scheme has equivalent or longer taxi times than the Iterative/Fully
Figure 8: Performance Comparisons
Constrained schemes, and shorter taxi times than the FCFS approach. Figs. 8(c) and 8(d) show the same comparison for the total taxi distance performance. The RH scheme plans are shown to have similar total taxi distances to the Iterative/Fully Constrained plans. As would be expected due to the FCFS approach’s use of the shortest path to runway, the RH plans are shown to have longer total taxi distances than the FCFS, due to the key re-routing behaviour of the coupled routing-timing scheme.

Figs. 8(e) and 8(f) show comparisons of the last time on runway or $t_{end}$ performance, a measure of throughput, of RH against Iterative/Fully Constrained and FCFS respectively. The RH scheme is shown in many cases to match the Iterative/Fully Constrained last time on runway (note it never improves on this due to the relatively high weighting on destination times $t_D$), and improve upon or match FCFS. In some cases the RH plan has a higher $t_{end}$, than both the non-RH and the FCFS approaches. This is a symptom of the close proximity of the problem within the 19-node structure. The ‘rough plan’ part of the RH scheme does not take into account physical infeasibility of reordering when computing the runway schedule. In larger problems this is not prevalent, although further refinement of the rough plan is a consideration for future work.

5 Conclusions and Further Work

In this paper we have presented a novel method combining in one optimization the taxiway operations and runway scheduling elements of airport surface movement in a continuous time environment. Additionally this is implemented in a receding horizon, iterative manner to improve scalability of computation.

The potential of the algorithm has been illustrated using a 240-aircraft example of ground
movement operations at Heathrow airport. Results demonstrated that the optimization scheme presented offers the potential for significant savings (over 50% on average) in total taxiing time over a FCFS approach.

Our simulation shows that the behaviour of arrivals differs little between FCFS and optimized operations. No significant improvement in taxiing time was observed in any arrival aircraft. It is speculated that this differentiation in the improvement between arrivals and departures is due to the lack of downstream constraints on arrivals.

Comparative tests broke down the contributions of each feature of the baseline algorithm to the computation achieved. We have shown that use of iteration of the conflict constraints can reduce solve times significantly, while maintaining globally optimal solutions. The RH formulation has been shown to significantly increases the scope of problems which can be handled by the optimizer. This is because not only does it allow faster computation of the same problems, but also allows easy transition to the continuous solution of problems in a rolling window sense.

Future work will concentrate on further studies of the large-scale heathrow airport problem in order to learn more about how the arrival and departure aircraft interact on the airport surface. This should include restricting the taxiway system, and introduction of enforced push-back times. Also comparative results with heuristic approaches to taxi planning are of interest.

Acknowledgements

This research is supported by Airbus Operations Ltd, the Knowledge Transfer Network for Industrial Mathematics (managed by the Smith Institute) and the Engineering and Physical Sciences
Research Council (EPSRC). The support of Sanjiv Sharma and Mick Dunne as well as useful discussions with Jason Atkin and Edmund Burke are gratefully acknowledged.

References


