A Fuzzy Approach to Addressing Uncertainty in Airport Ground Movement Optimisation

Alexander E. I. Brownlee[a], Michal Weiszer[b], Jun Chen[b], Stefan Ravizza[c], John R. Woodward[d], Edmund K. Burke[d]

[a]*Division of Computing Science & Mathematics, University of Stirling, UK; sbr@cs.stir.ac.uk*
[b]*School of Engineering and Materials Science, Queen Mary University of London, UK*
[c]*IBM Global Business Services, Zurich, Switzerland*
[d]*School of Electronic Engineering and Computer Science, Queen Mary University of London, UK*

Abstract

Allocating efficient routes to taxiing aircraft, known as the Ground Movement problem, is increasingly important as air traffic levels continue to increase. If taxiways cannot be reliably traversed quickly, aircraft can miss valuable assigned slots at the runway or can waste fuel waiting for other aircraft to clear. Efficient algorithms for this problem have been proposed, but little work has considered the uncertainties inherent in the domain. This paper proposes an adaptive Mamdani fuzzy rule based system to estimate taxi times and their uncertainties. Furthermore, the existing Quickest Path Problem with Time Windows (QPPTW) algorithm is adapted to use fuzzy taxi time estimates. Experiments with simulated taxi movements at Manchester Airport, the third-busiest in the UK, show the new approach produces routes that are more robust, reducing delays due to uncertain taxi times by 10-20% over the original QPPTW.

Keywords: Routing, Scheduling, Airport Operations, Optimization, Taxiing, Ground Movement, Uncertainty

1. Introduction

The aviation industry is experiencing sustained and long-term growth. It is estimated that air traffic within the European Union will reach 1.5x 2012 levels by 2035 [1]. As a result, many airports are operating near capacity, and the European Commission has recognised [2] the need to use existing infrastructure more efficiently as well as increasing capacity. Thus, there is increasing interest in better-performing decision support systems to optimise various airport operations[3]. Such systems need to cope well with the complex, integrated nature of airports, and model the processes realistically with minimal simplification of the constraints or uncertainties.

At many airports, a major bottleneck is the system of taxiways between the runways and stands. Optimisation of the Ground Movement of aircraft on

the taxiways is a critical problem [4]. It directly links other problems such as runway sequencing [5, 6] and stand/gate allocation [7, 8]. Furthermore, while only a fraction of the total journey consists of Ground Movement, it makes a large contribution to the running cost and emissions of an aircraft. Jet-engines are designed to operate optimally at cruising speed in the air, and are considerably inefficient while propelling an aircraft at low speed on the ground. It is estimated that fuel burn during taxiing alone represents up to 6% of airline fleet fuel consumption for short-haul flights with single-aisle aircraft from congested airports, resulting in 5m tonnes of fuel burnt globally[9], with reduced taxi delays offering potential savings of one third of that [10]. Reviews of Ground Movement research can be found in [3] and [4], with work published since these reviews including [11–29].

A critical issue remaining largely unaddressed by existing research (notable exceptions being [13, 17]) is the uncertainty inherent in Ground Movement. In particular, it is hard to accurately predict the time taken to travel between the runways and stands. This can be affected by slope, turning angle, other aircraft, runway crossings and simply the speed set by the flight crew. Existing approaches to Ground Movement optimisation typically assume that the taxi times are fixed. This can lead to a lack of robustness: an aircraft arriving at a point before or after the expected time can cause conflicts with other aircraft, leading to delays. Therefore, a decision support system which accommodates uncertainty has the potential to produce tighter, more efficient, and more robust taxiing schedules that bring an airport closer to its maximum capacity.

Some previous work has touched on this: [16] explored a number of different modelling approaches, with the aim of reducing the error in estimated times. That paper found that, depending on the model, 3.2-5.7% of all flights were incorrectly estimated by over 3 minutes, with the lower end of that range (the best estimates) from a fuzzy rule based system. A common means of tackling this problem is adding time buffers to absorb uncertainty in the taxi times [13]. This paper presents a more sophisticated approach. The uncertainty in the taxi times is represented using fuzzy membership functions, which come directly from an adaptive Mamdani fuzzy model of the taxi times. The Quickest Path Problem with Time Windows (QPPTW) algorithm [30] is extended to use these fuzzy times, generating multiple routes for different levels of uncertainty. This allows the decision support system to find a route assignment that is robust in a range of situations, yet still uses a minimal time to complete the movement.

As far as we are aware, no other research has applied fuzzy systems to handling uncertainty in Ground Movement, though fuzzy approaches have been applied to handling uncertainty in other transportation problems [31, 32] and are well established in more general scheduling problems [33–35].

Thus, the major contributions of this paper are: an adaptive Mamdani fuzzy rule based system (FRBS) from [16] is improved and extended to estimate taxi times and their uncertainties; and Fuzzy-QPPTW, an algorithm to allocate taxi routes to aircraft that are robust to taxi time uncertainty, is proposed. The new approach is demonstrated through the use of simulation to reduce delays caused by uncertainties in aircraft movements by 10-20% for higher levels of uncertainty.

This has the potential to reduce fuel burned by stopping and starting aircraft, and make better use of congested taxiways at busy airports.

The rest of this paper is structured as follows. We begin in Section 2 with a review of existing work in airport Ground Movement and uncertainty. In Section 3 we fully define the problem, and then describe the FRBS and new algorithm Fuzzy-QPPTW in Section 4. In Sections 5 and 6 we detail our case study, centred around Manchester Airport, and the simulator used to compare the different algorithms. We present and discuss experimental results in Section 7 and finally in Section 8 we draw our conclusions.

2. Related work

2.1. Ground Movement

Airport Ground Movement is a difficult problem which has been the focus of extensive research over the past couple of decades. Comprehensive reviews of this area are [3] and [4].

Early work in this area [36–38] used a list of routes that were either human-designed or generated before the algorithm was run using a shortest path algorithm. Heuristic search algorithms, such as genetic algorithms, selected an appropriate route and wait points for each aircraft. More recently, genetic algorithms were used to evolve the routes rather than choosing predefined ones [11]. Alternative efforts including [12, 21, 22, 39], [40] formulated Ground Movement as a mixed-integer linear programming problem. Ravizza et al. [30] describe the QPPTW algorithm, an adaptation of Dijkstra's shortest path algorithm that accounts for the movements of previously-allocated aircraft. Rather than optimisation of routes, congestion on the airport surface has also been reduced by management of the speeds, time slots, landing runway, and pushback delay [41]. Optimising gate allocations can also target reduced taxi times [25]. Often the focus is on optimising taxi times, but other objectives have attracted some attention, particularly reducing aircraft emissions and fuel consumption due to taxiing [22, 42–44], [45]. Integrated approaches to optimising Ground Movement with gate/stand allocation [24, 46] and runway sequencing [26, 47] have also been shown to provide more airport-wide improvements.

Most of the above methods assume fixed start or end times and taxi speeds. Some research has also attempted to account for the inherent uncertainty in this problem, which includes variations in push-back times, landing times and taxi speeds. Such uncertainty has been modelled as a fixed percentage of the initially defined taxi speed [37], with an aircraft occupying multiple positions on the taxiway graph simultaneously. An alternative is to use a planning horizon [39], so that aircraft routes were only determined up to a fixed time, and were then completed in subsequent iterations. Lesire et al. [48] used an increased temporal separation between aircraft to cope with uncertainty, and Koeners et al. [13] made use of time-margins around aircraft trajectories. The similar concept of buffering was discussed in [30]. Pfeil et al. [14] addressed the issue of rerouting aircraft to accommodate forecast bad weather. Furthermore, the

Ground Movement problem has been tackled as a network congestion control problem [15], making use of a probabilistic model based on Erlang random variables for taxi-out times at Boston Logan Airport.

There have been some attempts to reduce uncertainty by building more accurate models of taxi time using data provided by airports or airlines. Several papers describe models to accurately estimate taxi times for different aircraft movements. Balakrishna et al. [49] used reinforcement learning to predict taxi times at Tampa International Airport. The similarity of Ground Movement to road traffic flows was considered by Yang et al [29], in proposing a modelling approach based on the cell transmission model for simulating the evolution of flow and congestion on taxiways. Ravizza et al. [50] used a multiple linear regression to estimate taxi time at Zurich and Stockholm airports. In [51], an adaptive Mamdani Fuzzy Rule-Based Systems were used to estimate taxi times at Zurich Airport. A comparison of the latter two and several other modelling approaches [16] found the fuzzy approaches to yield the most accurate modelling. A regression model was used in [18] to estimate taxi times at Newark Liberty International Airport; a log-linear regression analysis to estimate taxi time was demonstrated by [52]. Truong [19] developed a probability distribution function to model taxi times for JFK Airport. More realistic timings can also be produced by redefining the taxiway model from a pure graph of nodes and edges to a zone based partition of the taxiways [23]. This achieves more realistic trajectory modelling through curves and intersections. Speed profiles [42] and Active Routing [44] also represent a path to more realistic routing. Khadilkar et al [53] used a dynamic programming approach combined with a model drawing on taxi times, arrival airspace and departures to determine the optimal push-back times for aircraft waiting on their stands at Boston Logan International Airport. This meant that aircraft could delay starting their engines to save fuel, mitigating the effect of delays in taxiing. A similar approach for delaying pushbacks Heathrow airport was described in [54].

Some work has also looked at the impact of uncertainty on Ground Movement. Lee et al. [17] described the use of a simulation of aircraft movements, given their route allocations and variations in speed profile, to measure the impact of uncertainty from various sources at Detroit International Airport. They confirmed that ground delay increases as uncertainty increases for most scenarios. Furthermore, they found that while routing allocations based on deterministic models can still be robust to certain types of uncertainties (such as runway exit times for arrivals), variations in taxi speeds resulted in significant increases in ground delay for departures. However, they did not propose an algorithm to explicitly handle uncertainty. A simulation was also used by [13] to explore the effect that the size of buffers around taxiing aircraft has upon throughput and robustness, finding that taxi time uncertainty is a major factor preventing the optimal use of the taxiways and runways. More detailed analyses of airport surface movements was conducted by [20] and [55], also reflecting the strong impact of taxi time uncertainty on the airport's efficiency. Morris et al [27] analysed various sub-problems within airport surface operations, and suggested that uncertainty in executing allocated taxi movements could be handled by

'migrating' part of the uncertainty into probabilistic or flexible predictive models. Stergianos et al [28] investigated how arriving aircraft can affect the routing process and whether pushbacks can result into different types of delays. They showed that arriving aircraft can indeed produce a lot of delay to overall taxi movements, and they noted the importance of having an accurate model for the pushback process. Lie et al. [56] gave a comprehensive assessment of the predictability impacts of airport surface automation. A wide range of the impacts is considered, which includes variability in taxi-out time, predictability of take-off time and take-off sequence, entropy of the airfield state, and perceived predictability from users. It has also been suggested that punctuality figures should be adjusted to use take-off times rather than push-back times, so that uncertainty in taxi times can be explicitly recognised and so tackled [21].

In summary, there has been considerable research into Ground Movement and some attempts to quantify the impact of uncertainty. While it has been established that taxi time uncertainty can cause delays, and has a large impact on the efficient use of taxiways and runways, only preliminary work has been conducted into generating taxi routes that are robust to uncertainty. This paper addresses this problem.

2.2. Stochastic routing

Closely related to the present work is the body of research in stochastic routing [57, 58]. This encompasses a family of related vehicle routing problems (VRP) with an uncertain element, which can be in the demands [59], travel times/costs [60], time-windows or other aspects of the problem. Exact and heuristic approaches have been applied to stochastic routing problems with uncertain travel costs. These are often termed *robust optimisation*, seeking to find a solution that is invariant under uncertainty such as different delay patterns.

One of the closest works to the present study is [61]. This proposed a Dijkstra-based routing algorithm applied to road and rail movements, but without time-windows. Link costs were either dynamic or stochastic, with labels on nodes being distributions, and travel times dependent on time of day. Rather than our approach, which considers the full range of uncertain times on each edge, their algorithm made comparisons and operations on labels using percentiles, taking scalar values from the distributions representing a fixed variation on the expected time. [62] considered a VRP with soft time-windows and stochastic travel times. The goal was minimising transportation cost, using another label setting algorithm to compute shortest paths. Rather than considering travel time, costs for comparing labels were associated with missed time-windows (which had stochastic start and end times). Again, the shortest path algorithm worked on the resulting crisp, scalar, costs.

Also close to the present study is [63], which focused on the stochastic orienteering problem with time-windows. This is essentially a combination of the knapsack problem and travelling salesman problem, with a reward for visiting each node, meaning that decisions must be made on which nodes to visit and in what order. Vertices have time-windows representing the expected arrival times

to qualify for the reward. [63] described a variant of the Dijkstra-based algorithm from [61] with node comparisons at the 99th centile to generate instances from which time-independent stochastic travel times could be computed. That work used the Ant Colony Optimisation metaheuristic as the main solution method, applied to graphs having up to 100 vertices. They used a Monte-Carlo simulation to test performance, similar to the approach we take. The method was shown to produce better solutions than those obtained for the regular orienteering problem when evaluated in a stochastic environment. This is also similar to our method of evaluating crisp and fuzzy QPPTW, both under uncertainty.

Recently, [64] looked at the related probabilistic orienteering problem, where the nodes were only available for service with a certain probability. A matheuristic algorithm based on the branch-and-cut scheme was described, with heuristic rules to reduce the solution space.

Surveying the literature in this area, we can make two important distinctions between our work and stochastic routing problems. In a general sense, Ground Movement with uncertain taxi times might be viewed as a VRP with hard but uncertain time-windows (i.e. hard constraints for missing time-windows), although the time-windows are on the edges rather than the nodes. In contrast to VRPs with Ground Movement very few nodes *must* be visited (in fact, usually just start and end). However, for each aircraft we are attempting to find a conflict-free path, so there are still time-windows on the intermediate edges to avoid aircraft conflicts. These time-windows are updated for every aircraft that is routed, so a dependency exists between vehicles not usually present in VRPs or SVRPs. Our algorithm also considers the taxi-time uncertainty in a tunable (via α-cuts) range of uncertainty levels, rather than only a centile taken from the distribution. Furthermore, rather than a probabilistic distribution, representing times as fuzzy membership functions means we can capture the uncertainties without needing to specify particular distributions for each taxi time, using a model that can be parameterised with qualitative metrics, e.g. route direction, traffic conditions etc.

3. Problem description

We now define underlying concepts that are key to this work. First, we discuss the links between Ground Movement and other airport optimisation problems. This is followed by a formal definition of the Ground Movement problem.

3.1. Links with other airport operations

Operations at an airport are highly complex and interrelated. Atkin et al. [65] noted the importance of integrating Ground Movements with other airport operations, such as gate allocation and finding efficient arrival and departure sequences. At many airports (including our case study, Manchester), a key issue is the crossing of active runways by taxiing aircraft. Arriving or departing aircraft are always given priority, effectively blocking the taxiway at regular

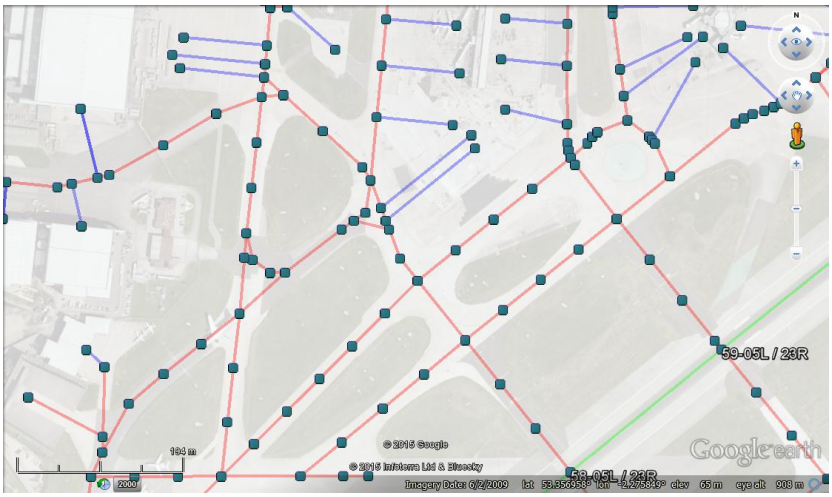


Figure 1: Part of the undirected graph representing Manchester Airport. Nodes represent the locations of stands, intersections, or intermediate points on taxiways used to maintain separation of aircraft. Edges are coloured blue to indicate stands, red to indicate taxiways and green to indicate runways.

intervals. In [30], while the taxi time model incorporated likely delays due to such crossings, the routing algorithm itself did not consider them. We take the same approach for the present work, except that the simulator used to recreate aircraft movements under uncertainty also includes runway crossings: the appropriate taxiways are blocked for an appropriate interval during landings and take-offs. We also follow [30] in solving the Ground Movement problem separate to the runway sequencing and gate allocation problems. Both runway sequencing and gate allocations are solved first, then the resulting take-off and landing times, and allocated gates, are considered to be fixed when solving the Ground Movement problem. Changes to runway sequence or gate allocations can be accommodated by rerunning the routing algorithm, with the routes of already-taxiing aircraft kept fixed. A study of tighter integration of runway sequencing and Ground Movement forms part of a separate piece of work [47] and more recently has been considered in [24, 25].

3.2. Ground Movement

Airport Ground Movement is a combined routing and scheduling problem [4]. Time-efficient routes must be allocated to aircraft seeking to traverse the taxiways between the runways and stands. Routes must respect allocated runway times, route restrictions, and safety constraints on the proximity of other aircraft. At less busy airports, where only a couple of aircraft are moving at any one time, it would be possible to assign routes using shortest path algorithms like Dijkstra's or A*. However, interactions between moving aircraft mean that a more sophisticated approach is required at busier airports.

Our approach does not use pre-determined routes, allowing for greater flexibility. As noted earlier, we assume that the runway time for an aircraft is fixed for departures and arrivals. The objective for arrivals is simply to find the quickest route to the stand. For departures, it is to find the quickest route from the stand that finishes at the runway at the allocated time. This approach means that the aircraft can be held on the stand as long as possible, delaying the engine start-up time and reducing fuel burn and emissions.

The airport layout is represented as an undirected graph $G = (V, E)$ (e.g. Fig. 1). Edges E represent taxiways and vertices V represent stands, junctions and intermediate points. An edge $e \in E$ has a set of weights T_e, the times to traverse e. Specifically which taxi time t_e in T_e applies for a particular aircraft depends on the previous edge traversed, airport operating mode (i.e. which runways are in use) and aircraft type (i.e. arrival or departure). Each e can only have one aircraft a_i on it at any one time. This is enforced by each e having an associated list of *time-windows* $\mathcal{F}_e$. The $\mathcal{F}_e$ specify the times that e is available to be used as part of a route. An aircraft will only be allocated a route for which there is a chain of time-windows along its entire length. This way, we ensure that the route is conflict free. Additionally, aircraft have a minimum separation distance of 60m at all times. To ensure this, G is pre-processed before routing to find the conflicting edges $conf(e)$ for each e, these being any edge sharing a vertex with e or passing within 60m of e. When an aircraft is present on any edge e, the time-windows of $conf(e)$ must be updated to prevent other aircraft coming in to conflict with it. Long edges are divided into lengths of no more than 60m by intermediate points to accommodate separation of consecutive aircraft on the same taxiway. This and subsequent notation are summarised in Table A.4 in the Appendix.

4. Adaptive Mamdani FRBS and Fuzzy-QPPTW

In this section we describe Fuzzy-QPPTW, an algorithm to find the quickest route for taxiing aircraft, given the existing routes of other aircraft and fuzzy estimates of taxi times for edges in the taxiway graph. Fuzzy-QPPTW extends the existing QPPTW algorithm [30]. First we describe the estimation of fuzzy taxi times for edges and introducing some key concepts in fuzzy sets and arithmetic. We then replicate the relevant details of the original QPPTW algorithm for convenience. Finally, we describe Fuzzy-QPPTW, in terms of the required changes to the original QPPTW. These are grouped into changes to the core algorithm, methods for calculating and comparing fuzzy taxi times, the operation to update time-windows, and an alternative approach to backwards routing for departures. Fig. A.14 in the Appendix gives an overview of how the components fit together to route a set of aircraft.

4.1. Adaptive Mamdani fuzzy rule-based system

Earlier work with QPPTW [30] considered taxi times for traversing edges as crisp real values. In the present work, we use fuzzy time $\tilde{t}$ that a given aircraft will take to traverse an edge e based on historical aircraft movements.

Given that nonlinearity is inherent in the Ground Movement problem, nonlinear modeling approaches, like fuzzy rule-based systems (FRBSs) with proven ability to approximate any real continuous function [66], are very suitable for predicting taxi times [51]. In particular, a Mamdani FRBS that outputs a fuzzy set is ideal as it can produce not only taxi time predictions, but also a membership function that quantifies uncertainty. However, as noted in [51], a standard Mamdani FRBS is highly dependent on human expertise and not suitable for making accurate predictions when: 1) such expertise is only partially or not available, 2) the problem is complex, with high dimensions in its explanatory variables. Given the abundance of historical aircraft movements, a data-driven fuzzy approach affords more accurate predictions, in turn leading to more accurate uncertainty quantification. Therefore, an adaptive Mamdani FRBS with a bespoke back-error propagation algorithm [51] was developed. Good performance compared to other methods, including linear and support vector regression, for predicting crisp taxi times has been shown [16]. While, in [16], a TSK FRBS was best performing, this type of FRBS only outputs a single value rather than a fuzzy set. The Mamdani FRBS represents a small sacrifice in accuracy to gain the benefit of uncertainty quantification. In this study, we extend the work in [16] to quantify uncertainty via the overall implied fuzzy set.

A typical Mamdani FRBS is defined by a number of fuzzy if-then rules in the following form:

$$\begin{aligned} \text{If } x_1 \text{ is } H_i^1 \text{ and } x_2 \text{ is } H_i^2, \ldots, \text{ and } x_j \text{ is } H_i^j, \ldots, \\ \ldots \text{ and } x_n \text{ is } H_i^n \text{ Then } y_i = Z_i, \end{aligned}$$

where x_j is the value of the j-th explanatory variable ($j = 1, 2, \ldots, n$), y_i is the output of the i-th rule, H_i^j is the fuzzy set (a linguistic value) for the j-th explanatory variable of the i-th rule and Z_i is the consequent of the i-th rule, and is defined as the fuzzy set B_i in a Mamdani FRBS. The original Mamdani FRBS is based on the so-called "sup-star compositional rule of inference" (as defined in (1)–(3)) and the overall implied fuzzy set $\hat{B}$ (as defined in (3)) [67].

$$\mu_{\hat{B}_i}(y) = \mu_i(X) * \mu_{B_i}(y), \tag{1}$$

$$\mu_i(X) = \mu_{H_i^1}(x_1) \cdot \mu_{H_i^2}(x_2) \cdot \ldots \cdot \mu_{H_i^j}(x_j) \cdot \ldots \cdot \mu_{H_i^n}(x_n), \tag{2}$$

$$\mu_{\hat{B}}(y) = \mu_{\hat{B}_1}(y) \oplus \mu_{\hat{B}_2}(y) \oplus \ldots \oplus \mu_{\hat{B}_i}(y), \quad i = 1, 2, \ldots, r, \tag{3}$$

where, r is the number of fuzzy rules in the rule-base. $\oplus$ and $*$ correspond to "sup" and "star", and are maximum and minimum respectively in the original Mamdani implementation. Note that the centre of average defuzzification was applied on $\hat{B}$ in order to derive a crisp output, which leads to two problems as mentioned in [67]: 1) $\hat{B}$ is itself difficult to compute; and 2) the defuzzification techniques based on $\hat{B}$ are also difficult to compute.

Therefore, in this work, the centre of gravity defuzzification is applied on the individual implied fuzzy set $\hat{B}_i$ as defined in (1). Instead of using minimum

and maximum, "product" is used for $*$ and "plus" is used for $\oplus$. In order to arrive at a differentiable analytical form after defuzzification, we further choose Gaussian membership functions for all of the explanatory variables and bell-shaped membership functions for the consequents, as defined in (4) and (5) respectively:

$$\mu_{H_i^j}(x_j) = \exp[-\frac{1}{2} \cdot (\frac{x_j - c_i^j}{\sigma_i^j})^2], \tag{4}$$

$$\mu_{B_i}(y) = \frac{1}{1 + (\frac{y - c_i^y}{\sigma_i^y})^2}. \tag{5}$$

where c_i^j and σ_i^j are the centre and the spread of the i-th membership function of the input. c_i^y and σ_i^y are the centre and spread of the i-th membership function of the output. Using the above modifications, a defuzzified output y^{crisp} of the FRBS for input X can be formulated as follows:

$$\begin{aligned} y^{crisp} &= \frac{\sum_{i=1}^{r} coa_i \cdot \int_y \mu_{\hat{B}_i}(y)\,\mathrm{d}y}{\sum_{i=1}^{r} \int_y \mu_{\hat{B}_i}(y)\,\mathrm{d}y} \\ &= \frac{\sum_{i=1}^{r} coa_i \cdot \mu_i(X) \cdot \int_y \mu_{B_i}(y)\,\mathrm{d}y}{\sum_{i=1}^{r} \mu_i(X) \cdot \int_y \mu_{B_i}(y)\,\mathrm{d}y} \stackrel{\text{def}}{=} y^{crisp}(X|\theta), \end{aligned} \tag{6}$$

where coa_i is the centre of area of $\mu_{B_i}(y)$ and is the peak, i.e. c_i^y, if $\mu_{B_i}(y)$ is symmetric. $\int_y \mu_{B_i}(y)\,\mathrm{d}y$ denotes the area under $\mu_{B_i}(y)$ over the output interval $y : [y_l, y_u]$ and is calculated using (7).

$$\int_y \mu_{B_i}(y)\,\mathrm{d}y = \sigma_i^y[\arctan(\frac{y_u - coa_i}{\sigma_i^y}) - \arctan(\frac{y_l - coa_i}{\sigma_i^y})] \tag{7}$$

In order to obtain a good estimation, i.e. y^{crisp}, the parameter vector $\theta = (c_i^j, \sigma_i^j, c_i^y, \sigma_i^y)$ needs to be chosen appropriately. The initial values of θ are derived by applying a clustering algorithm [68] on historical data containing aircraft movements. This vector is subject to further fine-tuning with a bespoke back-error propagation (BEP) algorithm [69] in a bid to improve the estimation accuracy of the FRBS, leading to an adaptive Mamdani FRBS. For details of the updating laws based on the BEP algorithm, readers are referred to [69].

It is well known that a large number of factors influence taxi times [49, 50, 55, 70]. The adaptive Mamdani FRBS uses explanatory variables identified in [50] as inputs for accurate prediction of taxi times, i.e. the output. The explanatory variables include airport operating mode; whether an aircraft is departing or arriving; total taxi distance; total turning angle; whether a push-back manoeuvre was performed; and the number of other moving aircraft divided into groups of arrivals and departures. Aircraft type (wake vortex category and number of engines) was excluded in line with the finding of [50, 70] that this had a poor correlation with taxi time. For the purpose of routing, all factors related to

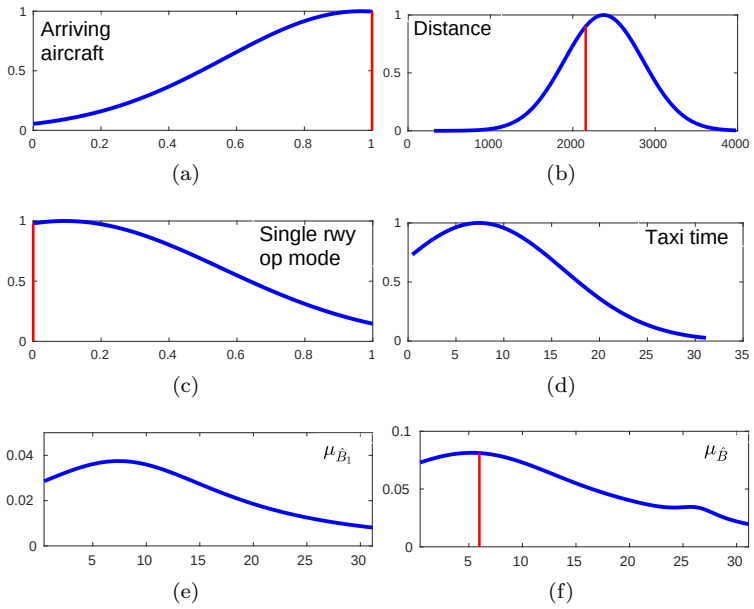


Figure 2: An example rule; (a)–(c) membership functions of explanatory variables and the values of the variables for a single example; (d) the membership function for the output; (e) the implied fuzzy set of this rule; (f) the overall implied fuzzy set with the indicated prediction.

other moving aircraft are set to 0, allowing estimation of unimpeded taxi times.

As an example, Fig. 2a–f illustrates an example rule of the Adaptive Mamdani FRBS, which has the highest firing strength with respect to the given inputs among all rules, as part of the inference system to derive taxi times. Fig. 2a–c depict membership functions of some explanatory variables and the values taken as the given inputs. Other explanatory variables are omitted for simplicity. Fig. 2d shows the membership function of the output pertaining to this particular rule. Fig 2e shows the implied fuzzy set of this rule. The rule can be interpreted using subjective abstractions: If the aircraft is arriving and the operating mode is in single runway and the distance is medium, then taxi time is short. Note that the choice of membership functions is quite subjective and does not come from randomness, nor should they follow probability theory, which deals with objective treatment of random phenomena [71]. Fig 2f shows the overall implied fuzzy set with the indicated prediction, which is the result of several fired rules working concomitantly within the FRBS. Readers interested in the primary difference between the study of fuzzy theory and probability theory are referred to [72].

As the elicited FRBS models historical aircraft movements accurately, in this paper, we further exploit the overall implied fuzzy set $\hat{B}$ as defined in (3) for accurate uncertainty quantification. Note that for a certain airport operational scenario as described by a combination of the explanatory variables, $\hat{B}$ effec-

tively gives a fuzzy time for traversing an edge on the graph. In order to use fuzzy arithmetic introduced later in Section 4.2, we use triangular membership functions of a fuzzy set $\tilde{t} = (l, b, u)$ as defined in (8) to approximate those of the resulting $\hat{B}$:

$$\mu_{\tilde{t}}(t) \begin{cases} 0 & t \leq l \\ \frac{t-l}{b-l} & l < t \leq b \\ \frac{u-t}{u-b} & b < t < u \\ 0 & t \geq u \end{cases} \tag{8}$$

where, b represents the modal, with l and u representing the lower and upper bounds respectively. The membership value of $t \in \mathbb{R}^+$ represents how likely t is to occur given $\tilde{t}$. Fig. 3a illustrates this approximation process. The intersection of the predicted taxi time and the membership function of $\hat{B}$ represents the modal of $\mu_{\tilde{t}}(t)$. The bounds l and u are extreme values defined by the membership function of $\hat{B}$. As l can take small values close or equal to 0, which is not a realistic speed for aircraft to taxi at, a limit is imposed to l such that the maximum speed of aircraft is 30 m/s. For u, a limit equal to the maximum taxi time in the historic data is applied.

4.2. Fuzzy arithmetic

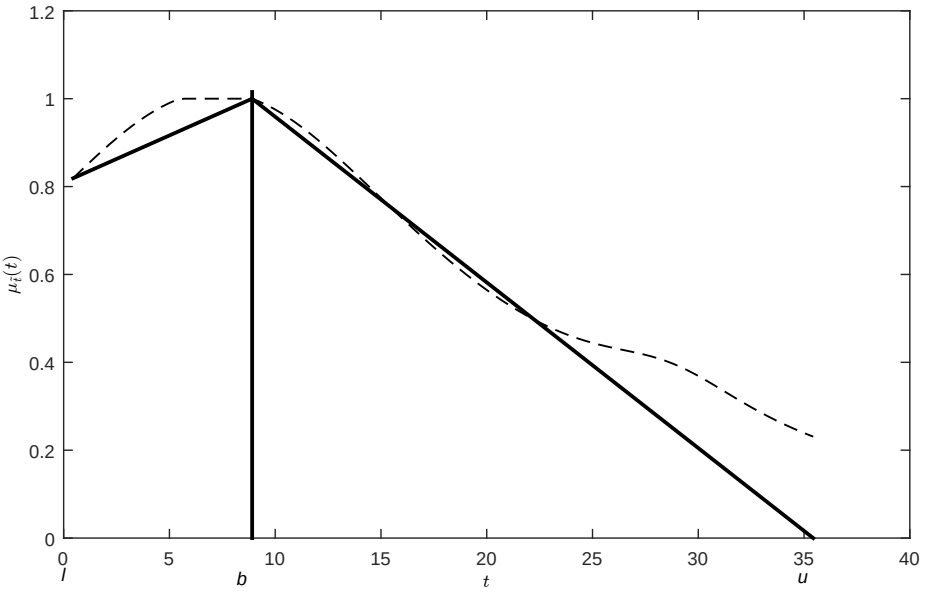
In order to adapt QPPTW to consider fuzzy times, we make use of several existing concepts. Our approach is broadly inspired by the algorithm for solving flow shop scheduling problems with fuzzy processing times in [35]. That was an extension of an earlier work by McCahon and Lee [73], which adapted Johnsons' exact algorithm [74] for flow shop scheduling. McCahon and Lee's idea is that the times, represented by triangular fuzzy membership functions $\tilde{t} = (l, b, u)$, were ranked by their Generalized Mean Values ($GMV(\tilde{t}) = (l + b + u)/3$). Here we make a few definitions which will be referred to in the description of the algorithm.

An important concept that we use is the set of α-cuts of a fuzzy set $\tilde{t}$. An α-cut of $\tilde{t}$, denoted by t^α, is a crisp set comprised of the elements in $\tilde{t}$ with a membership greater than α. More formally,

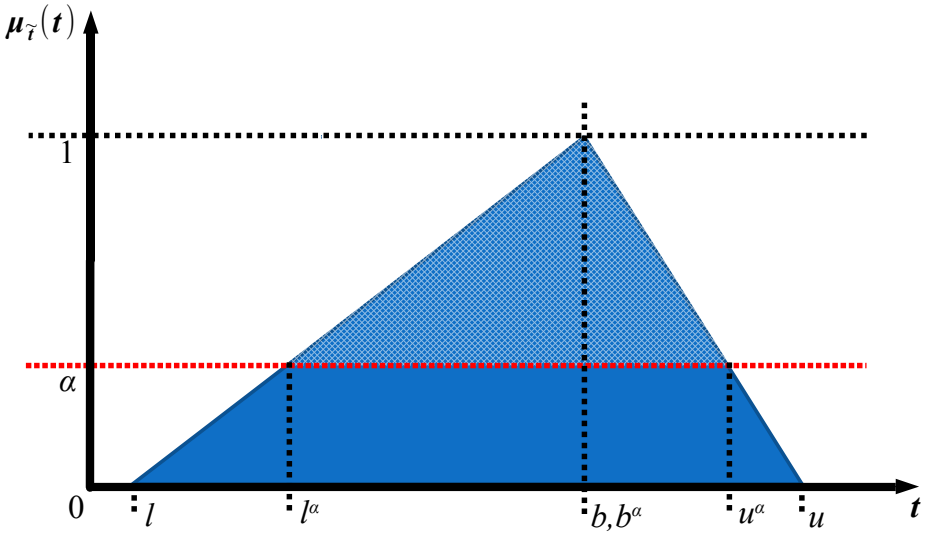
$$t^\alpha = \{x \mid \mu_{\tilde{t}}(x) \geq \alpha\}, \quad 0 \leq \alpha \leq 1. \tag{9}$$

An α-cut of t is illustrated in Fig. 3b, where t^α has the modal value $b^\alpha = b$, lower bound l^α and upper bound u^α.

In fuzzy set theory, the representation theorem states that a fuzzy set $\tilde{t}$ can be decomposed into a series of its α-cuts [75]. This in turn means that any fuzzy set can be derived from a family of nested sets: if $\alpha_1 > \alpha_2$, then $t^{\alpha_1} \subset t^{\alpha_2}$. A problem formulated in terms of fuzzy sets can be solved by decomposing them into their families of α-cuts. In order to compute the fuzzy taxi time for an aircraft along a series of edges, given fuzzy times on each edge and fuzzy time-windows, addition, minimum and maximum operations are required. As per [35] and [73], each operation can be decomposed into corresponding interval operations, conducted at each α-cut:



(a)



(b)

Figure 3: Triangular membership function (a) constructed from the membership function of $\hat{B}$ of the FRBS; (b) α-cut of the triangular membership function (shown by the hatched area).

Add Given two intervals $[l_1, u_1]$ and $[l_2, u_2]$,
then $[l_1, u_1] + [l_2, u_2] = [l_1 + l_2, u_1 + u_2]$

Min Given two intervals $[l_1, u_1]$ and $[l_2, u_2]$,
then $min([l_1, u_1], [l_2, u_2]) = [min(l_1, l_2), min(u_1, u_2)]$

Max Given two intervals $[l_1, u_1]$ and $[l_2, u_2]$,
then $max([l_1, u_1], [l_2, u_2]) = [max(l_1, l_2), max(u_1, u_2)]$.

For one or a combination of these operations, the partial results obtained for all α-cuts, $\alpha \in (0, 1)$, can be combined to yield a solution to the original fuzzy-set formulated problem. The resulting taxi time is not necessarily a triangular function, and in practice is highly unlikely to be so when calculated over an entire route. However, for convenience it can be represented as a triple $\tilde{t} = (l, b, u)$. We use the popular Yager's index [76] to scalarise the membership function for total taxi time:

$$Y(\tilde{t}) = \frac{1}{2} \int_0^1 (t_l^\alpha + t_u^\alpha) d\alpha \tag{10}$$

In comparing routes, we use the centre of gravity of $\tilde{t}$, which is simply the weighted mean of all the values in $\tilde{t}$

$$COG(\tilde{t}) = \frac{\sum_{x=l}^{u} \mu_t(x) x}{\sum_{x=l}^{u} \mu_t(x)} \tag{11}$$

As noted above, our method for adapting QPPTW to use fuzzy taxi times is based on the approach of [35]. It is noted in [35] that the approach of scalarising the fuzzy values (in that paper, using Generalized Mean Values / GMVs) for comparisons could miss optimal schedules because possibly useful information describing how times overlap is lost (see Fig. 4 for an example). For this reason, they took the approach of repeating the algorithm over several α-cuts, calculating a schedule for each α-cut, and finally choosing the best one overall. The final comparison between these alternatives is made using the fuzzy makespan computed using the full membership functions (rather than at each α-cut). In effect, the concept is to explore several alternative solutions for increasing levels of uncertainty ($\alpha \to 0$), but choosing the one that performs best under maximal uncertainty.

In our case, we repeat QPPTW for values of α from 0 to 1 in increments of 0.1. In each run, operations associated with edge weights and checks against time-windows are conducted over the interval of times at the value of α. The final comparisons of routes, calculation of taxi times and updating of time-windows all use the full membership functions (i.e. $\alpha = 0$).

4.3. QPPTW

Our work extends the QPPTW algorithm [30]. This algorithm resembles Dijkstra's shortest path algorithm, but also considers time reservations on the graph edges. Aircraft are routed sequentially, each being given the optimal route

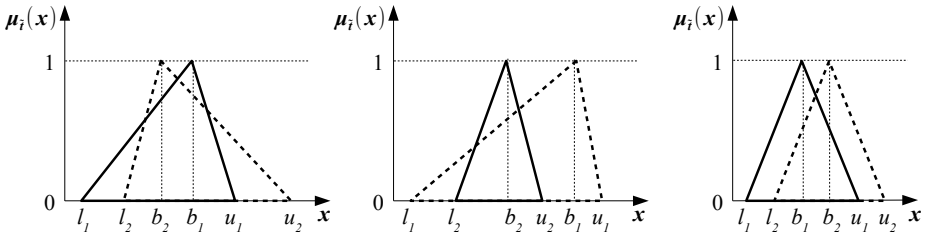


Figure 4: Three pairs of overlapping membership functions. In each, a simple scalarisation of the functions (e.g. GMV) results in the solid-line function being the minimal one. However, at high α values, the dashed-line one has the minimal value.

considering those aircraft that have already been routed. The overall framework proceeds as in Algorithm 2.

For convenience, the original QPPTW algorithm from [30] is replicated in Algorithm 1. For a given taxi request $Q_i = (v_{start}, v_{end}, \tau_{rw})$ for aircraft a_i, this finds the conflict-free route R_i from vertex v_{start} to v_{end} over G with the minimal taxi time $\mathcal{T}$ that respects the time-windows in E. τ_{rw} is the allocated runway time for the aircraft: the time to leave v_{start} or to arrive at v_{end} for arrivals and departures respectively.

Two key concepts in QPPTW are *time-windows* and *labels*. Each e has a sorted set of time windows $\mathcal{F}(e)$, representing the times e can be used. These exclude periods where e or an edge in $conf(e)$ are occupied by previously routed aircraft. A label $L = (v_L, I_L, pred_L)$ specifies the time period $I_L = [a_L, b_L]$ within which the current aircraft being routed could reach vertex v_L, given the previous label $pred_L$ in the route.

Once a route R_i to the destination vertex is found, the algorithm terminates. R_i is allocated to aircraft a_i, and the $\mathcal{F}(e)$ are trimmed, split or deleted to reflect times that a_i is present on each e. As long as a complete path of edges exists between v_{start} and v_{end} on G, QPPTW will always find a route. Any delay caused by conflicts with other aircraft will simply make the time of arrival at the stand later (for arrivals) or the required pushback time earlier (for departures, whose routes are computed backwards from the runway).

We have made two changes to the QPPTW algorithm presented in [30]. Firstly, in our approach, we use an undirected graph rather than a directed graph, to better reflect the operations at Manchester Airport. Secondly, we have removed the swap heuristic to save CPU time (it did not appear to make a large difference to taxi times in this work). It was noted by [30] that, according to [77], QPPTW will solve the problem in polynomial time in the number of time-windows: $O(|\mathcal{F}|^3 \log |\mathcal{F}|)$. More details of each step in the algorithm will be discussed in Section 4.4.

4.4. Modified steps of QPPTW main loop

The core of QPPTW has several operations involving times. We now discuss how each is modified to handle fuzzy values. We start with the operations for

Algorithm 1 The QPPTW algorithm, replicated for convenience

1: Let $H = \emptyset$
2: Let $\mathcal{L}(v) = \emptyset \ \ \forall v \in V$
3: Create new label L such that $L = (v_{start}, \tau_{rw}, nil)$
4: Insert L into heap H with key τ_{rw}
5: Insert L into set $\mathcal{L}(v_{start})$
6: **while** $H \neq \emptyset$ **do**
7: Let $L = H$.getMin(), where $L = (v_L, I_L, pred_L)$ and $I_L = [a_L, b_L]$
8: **if** $v_L = \tau_i$ **then**
9: From L, rebuild route R from v_{start} to v_{end}
10: **return** the route R
11: **end if**
12: **for all** outgoing edges e_L of v_L **do**
13: **for all** $F^j_{e_L} \in \mathcal{F}(e_L)$, where $F^j_{e_L} = [a^j_{e_L}, b^j_{e_L}]$, in increasing order of $a^j_{e_L}$ **do**
14: ▷ Expand labels for edges where time intervals overlap:
15: **if** $a^j_{e_L} > b_L$ **then**
16: **next** e_L ▷ Consider next outgoing edge (line 12)
17: **end if**
18: **if** $b^j_{e_L} < a_L$ **then**
19: **next** $F^j_{e_L}$ ▷ Consider next time-window (line 13)
20: **end if**
21: Let $\tau_{in} = max(a_L, a^j_{e_L})$
22: Let $\tau_{out} = \tau_{in} + \tau_{e_L}$
23: **if** $\tau_{out} \leq b^j_{e_L}$ **then**
24: Let $u = head(e_L)$
25: Let $L' = (u, [\tau_{out}, b_{e_L}], L)$
26: **for all** $\hat{L} \in \mathcal{L}(u)$ **do** ▷ Dominance check
27: **if** $\hat{L}$ dominates L' **then**
28: **next** $F^j_{e_L}$ ▷ Next time-window (line 13)
29: **end if**
30: **if** L' dominates $\hat{L}$ **then**
31: Remove $\hat{L}$ from H
32: Remove $\hat{L}$ from $\mathcal{L}(u)$
33: **end if**
34: **end for**
35: Insert L' into heap H with key $a_L{}'$
36: Insert L' into set $\mathcal{L}(u)$
37: **end if**
38: **end for**
39: **end for**
40: **end while**
41: **return** "there is no v_{start} to v_{end} route"

Algorithm 2 The overall QPPTW framework

1: Runway times are fixed for each aircraft
2: Aircraft are sorted to natural ordering (arrivals before departures, then by runway time)
3: **for all** Aircraft i **do**
4: Route i using QPPTW (Algorithm 1), considering already routed aircraft
5: Fix route of i and re-adjust time-windows
6: **end for**

computing edge entry and exit times as these are referred to by other steps, then proceed through the remaining parts of the algorithm in order.

4.4.1. Edge entry and exit times

Steps 21 and 22 calculate the aircraft's entry time τ_{in} and exit time τ_{out} on the edge.

$\tilde{\tau}_{in}$ is the latest of the arrival time at the start of the edge and the beginning of the current time-window (the aircraft cannot start to move along the edge until the time-window, marking the edge as available, has begun).

The start $\tilde{a^j_{e_L}}$ and end $\tilde{b^j_{e_L}}$ of the time-window are represented by fuzzy membership functions (Fig. 5a). The aircraft's arrival time $\tilde{a_L}$ at the start of the edge is represented by a third membership function (Fig. 5b). The start time for the edge traversal $\tilde{\tau}_{in}$ is the maximum of $\tilde{a_L}$ and $\tilde{a^j_{e_L}}$ (Fig. 5c).

The estimated time $\tilde{t_{e_L}}$ to traverse the edge is also fuzzy (Fig. 5d), coming from the adaptive Mamdani FRBS based taxi time estimation (Section 5.1). This is added on to $\tilde{\tau}_{in}$ to produce the fuzzy exit time $\tilde{\tau}_{out}$ for the edge, illustrated in Fig. 5e. The exit time $\tilde{\tau}_{out}$ for an aircraft on an edge e_L, given a label L representing a route to the start of e_L, depends on three variables:

- $\tilde{a_L}$, the arrival time at the start of the edge
- $\tilde{a^j_{e_L}}$, the start of the time window j (marking when the edge becomes available)
- $\tilde{t_{e_L}}$, the time taken to taxi along the edge.

These operations are carried out on the full fuzzy membership functions, without defuzzificiation.

4.4.2. Weight for Fibonacci heap

QPPTW stores the labels representing the shortest paths to nodes found so far in a Fibonacci heap. Each iteration (Step 7), the minimal label is removed and an attempt is made to expand the route represented by it. The labels in the heap are sorted according to their weight, which is a scalar value. For QPPTW this is the total taxi time for the partial route represented by the label.

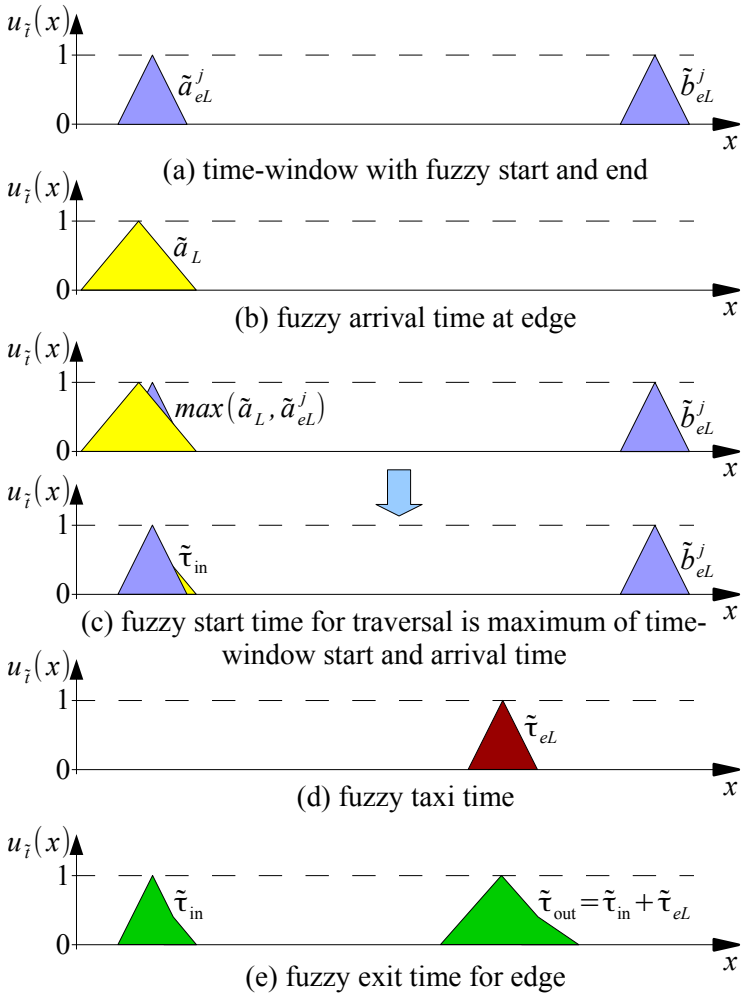


Figure 5: Adding fuzzy taxi times. No defuzzification takes place at this stage: the addition and maximum operations both return fuzzy values. Note that the resulting times in (e) are not triangular, due to the max() operation in computing the start time at (c).

The membership function for the total taxi time is not necessarily triangular, so we cannot use a simple metric like GMV for the weight. Instead, Yager's index at the present α-cut is used here for comparisons.

4.4.3. Ordering of time-windows

Once a label L is removed from the heap, QPPTW considers the set of time-windows F_{e_L} on each outgoing edge e_L of the vertex v_L represented by L. The loop in Steps 13 to 35 aims to find the times during which the edge is free to be traversed by the aircraft, and so determine the times at which the aircraft will reach the next vertex in the route. Earlier time-windows are considered first, with the aim of achieving shorter taxi times.

For this to work, F_{e_L} is sorted on the start times of each time-window. In Fuzzy-QPPTW, the membership function representing the possible start times of each time-window are defuzzified using the centre of gravity. F_{e_L} is then sorted in ascending order of these scalar values.

4.4.4. Comparing fuzzy times

Steps 15 and 18 check whether the current time-window contains the time of arrival at the start of the current edge. Two comparisons are made: less-than and greater-than, considering whether the time-window $F^j_{e_L}$ starts before the label L ends, and whether $F^j_{e_L}$ ends before L starts.

Recall that Fuzzy-QPPTW iterates over a set of α-cuts on the fuzzy membership functions representing taxi times. When comparing the fuzzy times, these are considered at the current α. Time period A is considered to have finished after B if the upper bound u^α of A is before the lower bound l^α of B, at the current α. So, the time-window $F^j_{e_L}$ starts before the label L ends if the following is false:

$$l^\alpha(a^j_{e_L}) > u^\alpha(b_L) \tag{12}$$

Likewise, the time-window $F^j_{e_L}$ ends before the label L starts if the following is false:

$$u^\alpha(b^j_{e_L}) > l^\alpha(a_L) \tag{13}$$

These comparisons replace those for crisp values at Steps 15 and 18 of the algorithm.

4.4.5. Comparing $\tilde{\tau}_{out}$ with $\tilde{b^j}$

In Step 23, the calculated fuzzy exit time for the aircraft's movement along the edge is compared to the end of the time-window. If the time-window finishes after the aircraft's movement, then we can move on to create a new label, representing an expansion of the route. If not, then the aircraft cannot complete its movement while the edge remains unreserved. This comparison is exactly the same as for the start times ((4) above), except that it is on the maximum values with membership at the current α-cut. This allows the algorithm to consider overlaps in the functions in iterations with lower α-cuts, where the probability of conflict is small.

4.4.6. Label dominance

Once it is determined that the aircraft can complete its movement along the edge within the present time-window, a new label is created to reflect the arrival time at the edge's end node. Steps 26 to 35 determine whether this new label dominates any existing labels for the node. A label is said to dominate another if it represents both an earlier possible arrival time at a node and a later possible departure time: more formally, L dominates $L\prime$ if and only if $a_L \leq a_{L\prime}$ and $b_L \geq b_{L\prime}$. In Fuzzy-QPPTW, the comparisons are made using the Yager's index on the membership functions for a and b on both labels, at the present α-cut.

4.4.7. Reconstructing the route

Once QPPTW's main loop reaches a label representing a route to the destination, the algorithm terminates. As the minimal label (that with the lowest weight: the shortest length of time) is always removed from the heap, this represents the quickest path. The route can be reconstructed by following each node's reference to its predecessor in turn. Timings for the route (edge entry and exit times) are determined by copying the earliest and latest arrival times at each node contained in each label. These are used to reallocate each edge's time-windows (Section 4.4.9). Note that this is an instance of the standard dynamic programming algorithm, being an extension of Dijkstra's shortest path algorithm (a classic dynamic programming exemplar) to handle times.

There is an added level of complexity here for Fuzzy-QPPTW, because the membership function $\tilde{\tau}_{out}$ representing an aircraft's possible edge exit times might overlap $\tilde{b}^j_{eL}$, the end of the time-window. This means that there is uncertainty about whether the aircraft will complete taxiing along the edge within the time-window. In this situation, the aircraft might need to wait for the next time-window. To accommodate this, when the route is reconstructed, the fuzzy times for each node are compared to the time-windows, and if necessary, a set of multiple exit times π_{out} for an edge are computed following Algorithm 3. This carries over to become a set of multiple entry times π_{in} for the following edge, ultimately leading to a set of multiple possible completion times for the route. Each time in this set is itself a membership function.

4.4.8. Choosing the route to allocate

As noted in Section 4.2, Fuzzy-QPPTW performs the above operations for multiple values of α. In this way, we obtain, for multiple values of α, a route and a fuzzy membership function representing the total taxi time to complete it. As per Section 4.4.7, there are usually multiple functions representing a spread of possible exit times. A sample of exit times generated using the procedure is shown in Fig. 6. Note in this figure that the highest values of α have the greatest uncertainty in the exit times. This might seem counter-intuitive: $\alpha = 1$ corresponds to using crisp taxi-times on the edges. As noted at the end of Section 4.2, the exit times are computed for each α using the full membership functions, not the α-cut. The point of this is that the alternative routes are being compared on their possible performance under full uncertainty. We would

Algorithm 3 Finding the set of membership functions representing possible completion (exit) times for the route. Note that $\tilde{\tau}, \tilde{a}, \tilde{b}$ are fuzzy sets, and e.g. $l(\tilde{\tau})$ and $u(\tilde{\tau})$ denotes the lower and upper bounds of $\tilde{\tau}$

1: $\pi_{in} = \emptyset$ ▷ For the first edge, initialise π_{in} to the route start time
2: $\pi_{in} \leftarrow \tau_{start}$ ▷ (after the first edge, π_{in} can contain multiple times)
3: **for all** edges e in route R **do**
4: $\pi_{out} = \emptyset$
5: **for all** $[\tilde{\tau}_{in}, \tilde{\tau}_{out}]$ in π_{in} **do**
6:
7: **for all** time-windows $F^j = [\tilde{a^j}, \tilde{b^j}]$ on e **do**
8: **if** $u(\tilde{b}) < l(\tilde{\tau}_{in})$ **then**
9: **next** F^j ▷ Time-window too early, skip (line 7)
10: **end if**
11: **if** $u(\tilde{b}) < l(\tilde{\tau}_{out})$ **then**
12: ▷ Exit & end times overlap
13: ▷ Cut off $\tilde{\tau}_{out}$ at mid-point of overlap:
14: $z = \frac{u(\tilde{b}) + l(\tilde{\tau}_{out})}{2}$
15: $\tilde{\tau\prime}_{out} = min(\tilde{\tau}_{out}, z)$
16:
17: ▷ Create new start time in next TW:
18: $\tilde{\tau\prime\prime}_{in} = \tilde{a^j}$
19: $\pi_{in} \leftarrow \tilde{\tau\prime\prime}_{in}$
20:
21: ▷ Calculate corresponding exit time:
22: $\tilde{\tau\prime\prime}_{out} = \tilde{\tau\prime\prime}_{in} + \tilde{t}_{e_L}$
23: $\pi_{out} \leftarrow \tilde{\tau\prime\prime}_{out}$
24:
25: ▷ Replace current times with new
26: ▷ and check against next TW:
27: $\tilde{\tau}_{in} = \tilde{\tau\prime\prime}_{in}$
28: $\tilde{\tau}_{out} = \tilde{\tau\prime\prime}_{out}$
29: **else**
30: ▷ Exit & end times do not overlap
31: $\pi_{out} \leftarrow \tilde{\tau}_{out}$
32: **end if**
33: **end for**
34: **end for**
35: $\pi_{in} = \pi_{out}$
36: **end for**
37: Return τ_{out}

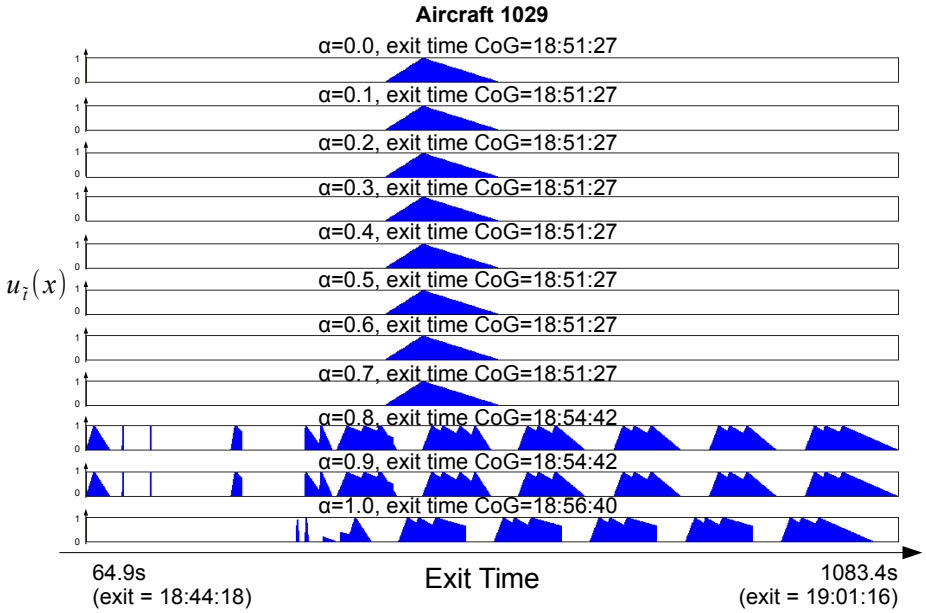


Figure 6: Exit times for routes generated by Fuzzy-QPPTW at various α-cuts. The lower and upper bounds in milliseconds are given at the top, with the centre of gravity for each set of exit times given as a time in hh:mm:ss format alongside the corresponding α. Here, the algorithm found the same route for α-cuts 0-0.7, and for 0.8-0.9, so there are three sets of possible exit times.

expect that routes computed assuming low levels of uncertainty would be more heavily impacted by the full uncertainty. This is consistent with much greater variation in the possible exit times being seen for routes calculated using high α values.

The centre of gravity (CoG) is computed over all possible exit times at each α for the route: the route with the lowest CoG being allocated to the aircraft. No assumption is made as to the likelihood of each time, so each is given an equal weight in the CoG calculation. This approach regresses to the crisp version of QPPTW with $\alpha = 0$, so routes generated by that method will still be considered. It is also worth noting that the runs at each α are independent, and can be conducted in parallel to reduce the overall run time.

4.4.9. Reallocating time-windows

Once a route has been allocated to an aircraft, the final step is to reallocate the time-windows on any edges that conflict with it. This means that routes allocated to subsequent aircraft can avoid earlier ones. This is achieved by either shortening or splitting time-windows to reflect the slots taken up by the aircraft's movement. If any of the resulting time-windows are too short to be useful (that is, they are shorter than the minimum taxi time for their edge), they are deleted. For Fuzzy-QPPTW, the same process is followed, deleting a time-window if its upper bound is less than the edge's minimum taxi time.

An additional cleaning-up step is performed as part of this to reduce memory requirements and to improve the speed of the algorithm. Given the time-splitting when reconstructing the route (noted above), a single route will often have multiple start times for each edge. As the route is built up, particularly with the presence of many other aircraft leading to many time-windows, later edges can have many start times, some being highly similar. For each edge, if a membership function for a start time was found to match the shape of another (with a tolerance of $\pm$1s), the two start times are merged, taking the minimum of their lower bounds, the maximum of their upper bounds, and the mean of their modal points.

4.4.10. Backwards routing

The original QPPTW algorithm [30] included a backwards variant for departing aircraft. The route was constructed working back from the runway at the allocated take-off time, so the computed pushback time was as late as possible while still meeting the target runway time. Any waiting could then be absorbed at the stand before the engines had started, reducing fuel burn and emissions. This cannot be easily adapted to handle uncertainty, because uncertainty grows as the route is constructed. Working backwards from a target exit time, the route is constructed having greatest source of uncertainty in timings at the start, with the greatest certainty at the end. Obviously, this is the opposite of reality.

Instead, we propose an iterative approach to determining the latest pushback time, following a bisection method, presented formally in Algorithm 4. This aims to find a *feasible* route R (where the exit time is before the aircraft's

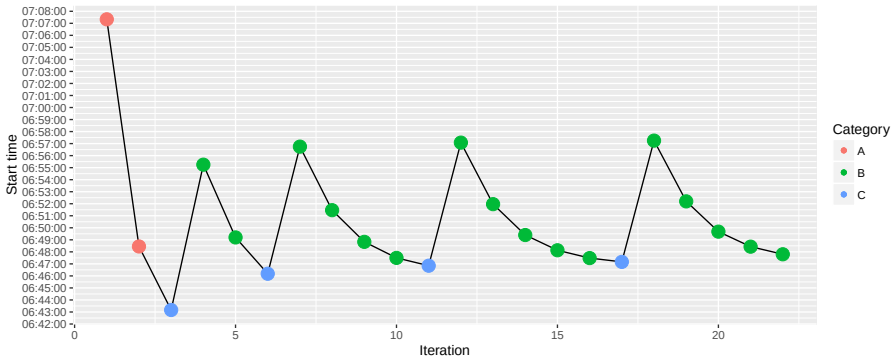


Figure 7: Start times for one aircraft, revised using the bisection approach. Times labelled A are later than the allocated runway time, and at this point no feasible route has been found, so the algorithm makes large jumps back. Times labelled C are earlier than the runway time, so when these are found, the algorithm tried making the start time later. Times labelled B are later than the allocated runway time, found after we have obtained a feasible route, so the algorithm makes small steps back, trying to find the "sweet spot" where the start time is as late as possible but the route is still feasible.

allocated runway time), which starts as late as possible. We assume a start time τ_{start} of the runway time τ_{rw}, minus the estimated taxi time for the shortest possible route in terms of distance (computed using the classic Dijkstra's algorithm). Fuzzy-QPPTW is run forwards, building up the route from the stand. The exit time τ_{out} allocated route will likely be fuzzy: here we consider only the modal value of the fuzzy set. In the event that no route was found at all, 10 minutes are subtracted from τ_{start}. If no feasible route has been found yet, and τ_{out} is later than τ_{rw}, then τ_{start} is made earlier by the difference $\tau_{rw} - \tau_{out}$. If a feasible route has been previously found, and τ_{out} is later than τ_{rw}, then we added too much time to τ_{start} previously, so τ_{start} is made a little earlier (by half as much as was added to it in the previous iteration). If τ_{out} is earlier than τ_{rw}, then τ_{start} is made later by half the difference $\tau_{out} - \tau_{rw}$. This repeats until $\tau_{out} = \tau_{rw}$ or a fixed number of iterations have passed (20 in our experiments), returning R_{best}, the latest-starting feasible route found. In our experiments, this number of iterations always found a feasible route. In heavy traffic situations the departing aircraft are just given an earlier pushback time in order to meet their target runway time. 20 iterations allows for pushback up to 200 minutes early, which in practice is far more than enough. Example start times for one aircraft as bisection proceeds are illustrated in Fig. 7.

5. Analysed Case: Manchester Airport

Manchester Airport is the third busiest airport in the UK, both in annual passengers (20.7m) and aircraft movements (159000), according the UK Civil

Algorithm 4 Algorithm to find the route R_{best} having the latest start time that still meets the runway time for departing aircraft a

```
1: R_best ← ∅
2: τ_inc = 0
3: Compute R_min over G using Dijkstra's algorithm
4: Estimate the time t_min to complete R_min
5: τ_0 ← τ_start ← (τ_rw − t_min)
6: repeat
7:     R ← FuzzyQPPTW(a)
8:     if R == ∅ then
9:         τ_start = τ_0 − 10min
10:    else
11:        if R_τout > τ_rw then                          ▷ late
12:            if R == ∅ then                             ▷ large jump earlier
13:                t_inc = τ_rw − R_τout
14:            else                                       ▷ small jump earlier
15:                t_inc = t_inc/2
16:            end if
17:            τ_start = τ_0 − t_inc
18:        else if R_τout < τ_rw then                     ▷ early, make later
19:            t_inc = t_inc − min(t_inc/2, R_τout − τ_rw)
20:            τ_start = τ_0 − t_inc
21:        else                                           ▷ perfectly on time
22:            R_best ← R
23:        end if
24:    end if
25: until (count > 20) or (R_τout == τ_rw)
26: return R_best
```

Aviation Authority[1]. From a Ground Movement perspective it has several interesting features. There are three terminals, two runways (05L/23R and 05R/23L) and 148 aircraft stands. 54 stands are 'shadowed' and cannot be used when larger aircraft are on adjacent stands. 57 stands are served by terminal piers, and 91 are remote and accessed by bus. Access to runway 05R/23L is achieved by crossing 05L/23R. Access to stands on the apron serving terminal 2 and part of terminal 1 can be limited to a single taxiway depending on stand usage because several stands block the alternative taxiway. Aircraft on longer stopovers are often towed to remote stands to free up stands at the terminals, placing further demand on the taxiways. A stylised diagram of the airport is given in Fig. 8.

[1]Taken from CAA data:
`http://www.caa.co.uk/docs/80/AviationTrends_Q3_2013.pdf` and
`http://www.caa.co.uk/docs/80/airport_data/201401/Table_01_Size_of_UK_Airports.pdf`

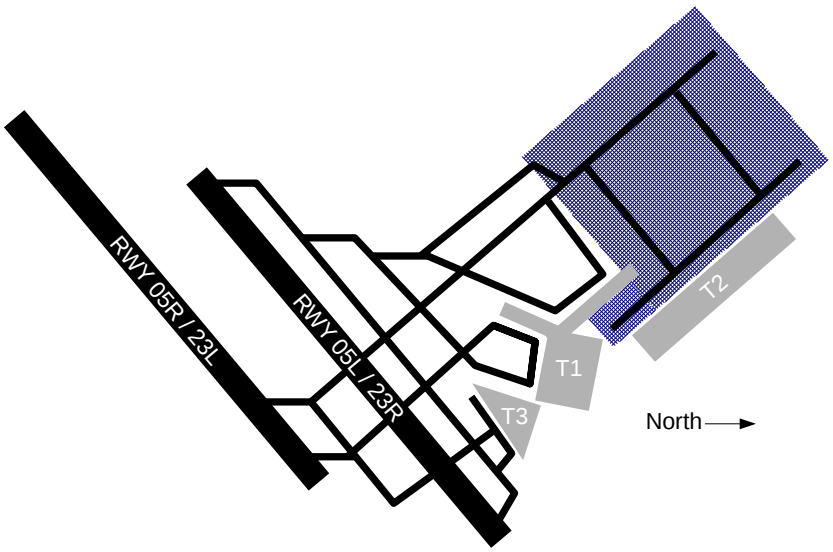


Figure 8: Stylised diagram of Manchester Airport showing the terminals (grey), runways (black) and major taxiways (black). The apron (blue hatched) can only be reached by two taxiways: this bottleneck restricts access to around half of the airport's stands.

Most days (except in unusual prevailing wind conditions), the airport switches between two operating modes. In single runway mode, runway 05L/23R is used for both arrivals and departures. In twin runway mode (daytime busy periods only), arrivals use runway 05L/23R and departures use 05R/23L.

The graph specifying the airport layout was generated using the GM Tools[2] [78]. A set of real aircraft movements for a day of operations was provided by Manchester Airport. We took these and generated additional sets of movement data for varying traffic levels by selecting aircraft at random and either duplicating or removing them. Duplicated flights used the same runway as the original, with the runway time having a two minute offset added in order not to take the runway at exactly the same time. In this way, sets having 0.8, 0.9, 1.1, 1.2 and 1.3 times the number of original aircraft movements were created, giving us scenarios for our experiment ranging from low-density traffic (80% of reality), through a real day of operations, to high-density traffic (130% of reality). Both layout and traffic data sets are available from the repository of Ground Movement benchmarks at ASAP Nottingham[3]. The number of aircraft movements in each density is shown in Table 1.

[2] `https://github.com/gm-tools/gm-tools/wiki`
[3] `http://www.asap.cs.nott.ac.uk/external/atr/benchmarks/index.shtml`

Table 1: Number of aircraft at each density

Density	Number of Aircraft Movements
0.8	516
0.9	575
1.0	640
1.1	701
1.2	790
1.3	851

5.1. Taxi time model with uncertainty

The flight movement data used to train the adaptive Mamdani FRBS (described in Section 4.1) represents real aircraft movements taken from freely-available data on the website FlightRadar24 (FR24), following the techniques described in [78] using the tools available at `https://github.com/gm-tools/gm-tools/wiki`. The same source has also been used to gather airborne flight tracks [79–82]. FR24 gathers automatic dependent surveillance-broadcast (ADS/B) messages that are transmitted by many aircraft and contain the latitude, longitude and altitude of the aircraft, typically every 5 to 10 seconds. The coordinates have a resolution of 10^{-4} degrees: approximately 10m at the latitude of Manchester Airport. While not all aircraft broadcast ADS/B data, and some of those that do show calibration errors or other corruptions that need to be cleaned before it is useable, enough flight movements are available so that a reliable taxi time estimation model can be derived.

For this work, all 1767 tracks for 5-12 November 2013 for aircraft with an altitude of zero within 5km of the airport's centre were collected (in the same period, public flight times on the web showed 3211 flights). These tracks were snapped to the actual taxiways at Manchester Airport by searching for all taxiways within 10m of each coordinate and deriving the most likely route taken. 1413 aircraft Ground Movements remained in the data after this processing. Each movement contained an actual taxi route taken, with the real time at the start and end, as well as at some intermediate points. The data was divided at random into 2/3 training and 1/3 test data sets. Following training of the FRBS, the predicted modal times estimated by the model were found to fit the real times in the validation data with $R^2 = 0.70$. For Ground Movement, accuracy of taxi times within 3 and 5 minutes are the most common measures used [49]. 84% of movements were accurate to within 3 mins and 95% were accurate to within 5 mins. A plot showing the time estimates for the validation data is given in Fig. 9. Further 10-fold cross validation of the FRBS proved the robustness of the model with $R^2 = 0.61$, 85% and 94% of accurate movements within 3 and 5 mins, respectively, comparable with previous studies [49, 50].

6. Ground Movement simulator

In order to measure the impact of uncertain taxi times on aircraft movements, a simulator was developed to reflect real airport operations. The rules

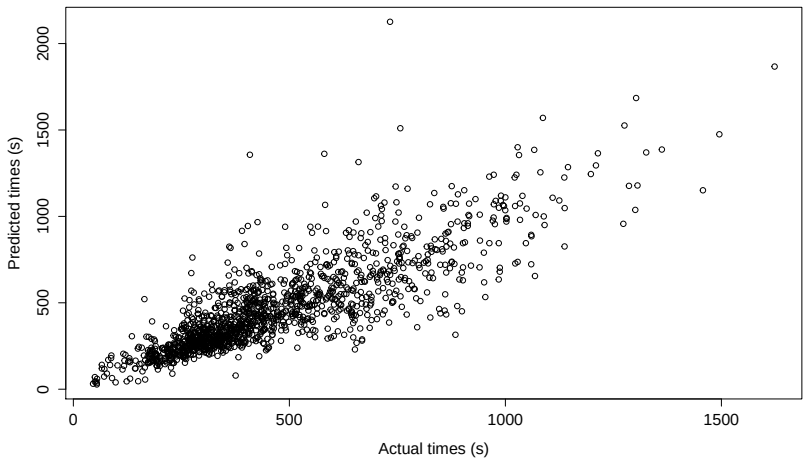


Figure 9: Model estimated taxi times vs real measured times for validation data.

incorporated in this were determined in consultation with our industrial partners at Manchester and Zurich Airports.

A pilot is guided by the air traffic control tower, with some discretion in stopping points. The typical minimum separation between aircraft in good conditions is 60m. To avoid collision, an aircraft may stop or reduce speed; for simplicity, the aircraft is assumed to stop at the last possible point. The duration of this stop contributes towards the aircraft's delay.

The simulator employs an object-oriented architecture, implemented in Java 7. The major data structures are as follows. Aircraft objects encapsulate data including runway time, allocated route, current position (i.e. how far it is along which edge in its allocated route). A JGraphX graph object represents the "current" layout of taxiways as edges and nodes. Each edge has an associated slot for an aircraft on it. A manager object keeps track of any aircraft currently on the runway, and blocks any edges that cross it. The simulation takes a discrete time-step approach, and follows the procedure outlined in Algorithm 5, which we will now explain. Prior to commencing, the routes for all aircraft are preprocessed using the taxi time estimation model to determine the average speed on each edge of the taxiway graph. Steps 5-18 represent the main loop of the simulator. Each iteration represents one increment in time (0.1s in our experiments). In Steps 6-16, all the aircraft are placed in new positions on the taxiway graph. Aircraft will be moved to new locations by taking their previous location and adding the distance travelled in 0.1s at the speed for this part of the route, calculated in Step 1. The new locations may be revised if moving to the new location would conflict with another already-placed aircraft.

To detect conflicts between aircraft, the procedure is similar to that in QPPTW. If an aircraft crosses or shares a path with another, it travels to

Algorithm 5 Procedure for the simulator working with the set of all routed aircraft A

```
1: Preprocess(A)
2: A_current ← ∅                                   ▷ Set of current aircraft
3: t = 0                                           ▷ Current timestamp
4: Add any a_i ∈ A with R_τstart = t to A_current
5: while t < τ_finish do
6:     for all A_current as a_i do
7:         removeFromGraph(a_i)
8:         p_i = calcNewPosition(a_i)
9:         if isCompleted(a_i) then
10:            Remove a_i from A_current
11:        end if
12:        if inConflict(a_i) then
13:            Update a_i position to latest clear point in route
14:        end if
15:        addToGraph(a_i, p)
16:    end for
17:    t = t + 0.1
18: end while
```

the last point before the conflicting segment of the taxiway graph. The first aircraft reaching the conflicting segment proceeds and the other aircraft must wait until the segment is clear. Aircraft are counted as occupying all edges crossing the runway for 2 minutes prior to runway time for arrivals and 1 minute after runway time for departures. This procedure is illustrated in Fig. 10.

On this basis, the simulator recreates the expected movements of the aircraft, detecting stops due to conflicts, and allowing measurements of the total taxi times and delays to be made. In order to measure the impact of uncertainty, we perturb the time estimates used to calculate taxiing speeds in the preprocessing stage. In our study, we are trying to measure the robustness of the allocated routes to uncertainty in the taxi times. Recall that the adaptive Mamdani FRBS produces fuzzy estimates of the taxi time to cover each edge. These approximate the uncertainty present in the underlying real-world data. We could just sample each function at random as if it were a probability distribution, similar to the method described by [83]. Here, for a triangular function (a, b, c), we have two equiprobable probability density functions, (a, b) and (b, c), assuming that the area under the lines representing the membership values in each is 1. Sampling these distributions for each edge would produce a range of possible times in the range (a, c), biased towards the modal time b. The problem with doing this is that, along the whole route, the fast and slow taxi times will cancel out. Instead, we have opted for the more realistic scenario that a given aircraft will be slower or faster than expected along the whole route. This represents the worst-case scenario in terms of uncertainty: in practice there will be some parts of the taxi movement that cancel out other slower or faster parts. The idea is

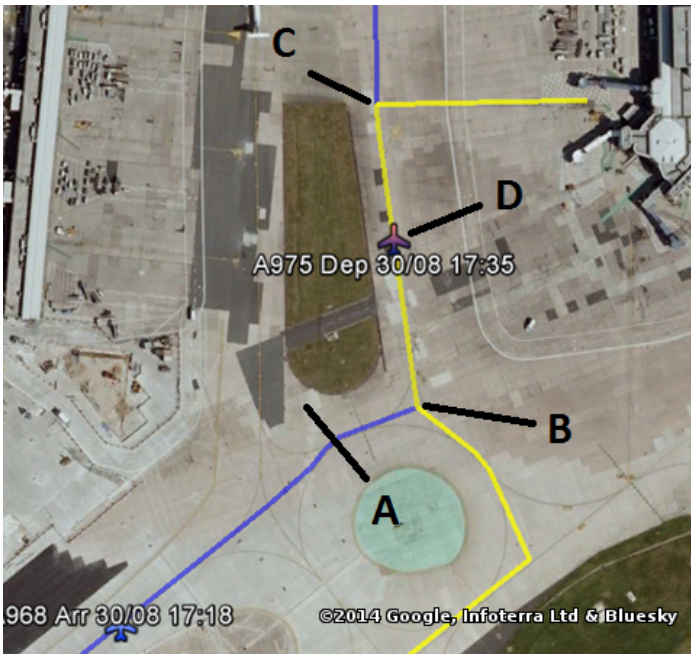
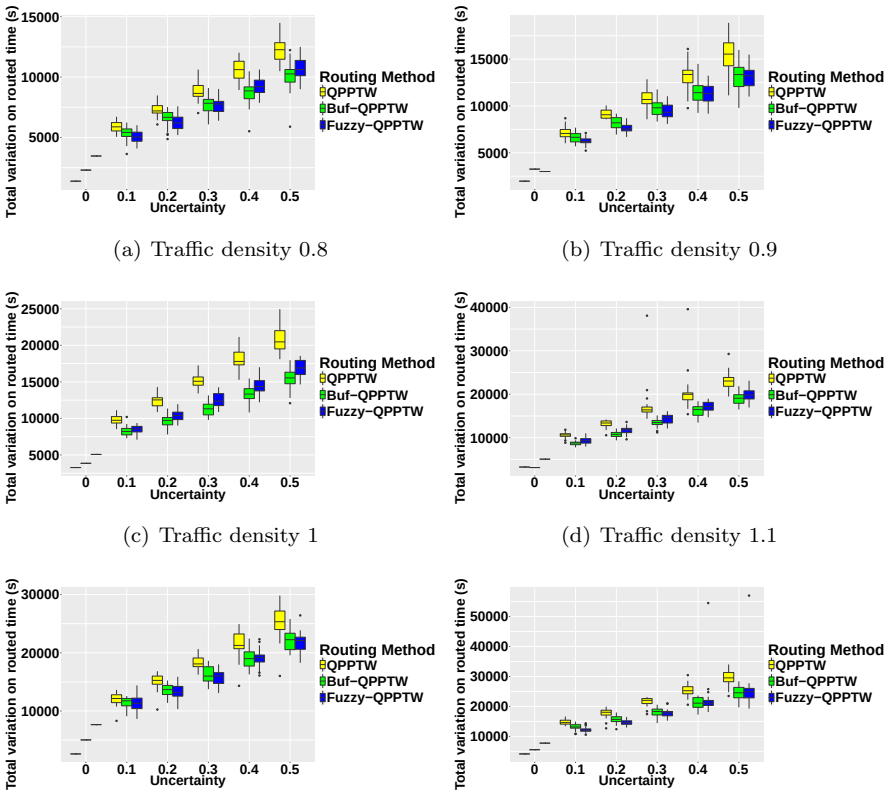


Figure 10: Handling of conflicts in the simulator. Aircraft 968 (arrival) and 975 (departure) have conflicting paths between B and C: 968 moving B→C and 975 C→B. At the current time step, 975 has reached D. Conflict is assessed in terms of the remaining path, so the conflict area is between B and D. A is the last node before 968 is within 60m of the conflict area: if 968 reaches A before 975 clears B, 968 stops. 968 only restarts once 975 has passed B.

that for a given aircraft a_i, a random number u_i is chosen in the range $[0, 1]$ inclusive. This controls from where – in the fuzzy set of times for each edge – the taxi time for the simulator is taken:

- If $u = 0.5$, modal time is used for each edge on the route.
- If $u < 0.5$, time for each edge is $a + 2u(b - a)$
- If $u > 0.5$, time for each edge is $b + 2u(c - b)$

u is allocated to each aircraft prior to preprocessing the speeds for the taxiway graph at the beginning of the simulator run. The precise u values are varied from one run to another by changing the random number generator seed, but in any one run, u will be unique for each aircraft. The amount of uncertainty present in the system can be controlled by limiting the range of values the u can take to $0.5 \pm \delta$. In the results given in the next section, we quantify uncertainty in terms of this δ. Zero uncertainty means that $u = 0.5$ for all aircraft (so only modal times are used). The maximum value is 0.5 uncertainty, which means that $u \in [0, 1]$, that is, 0.5 ± 0.5.



(a) Traffic density 0.8

(b) Traffic density 0.9

(c) Traffic density 1

(d) Traffic density 1.1

(e) Traffic density 1.2

(f) Traffic density 1.3

Figure 11: Total variation of simulated times on routed times, for each traffic density. Each group of three bars has the same uncertainty δ. Dijkstra omitted to reduce skewing of the scale: see Figure 12 for the results at traffic density 1 with Dijkstra included.

7. Experiments and Results

We applied Fuzzy-QPPTW to the set of benchmark scenarios for Manchester Airport described in Section 5. We considered four routing algorithms: (1) the original QPPTW; (2) Fuzzy-QPPTW; (3) the original QPPTW with fixed buffers, and (4) a simple heuristic approach using fixed shortest paths as a baseline. Approach (3), here referred to as "Buf-QPPTW", was suggested in [30]. A buffer is added to the estimated time on each edge, so that the edge is reserved for longer than needed to contain any delay. An obvious disadvantage is the extra parameter (buffer size) introduced by this approach, for which there is little guidance as to an appropriate value. We used buffers equivalent to a taxi speed of 3m/s lower than the model estimated speed, this figure being taken from [48]. Approach (4), here referred to as "Dijkstra", used a single fixed shortest path between each runway and stand pair. The shortest paths

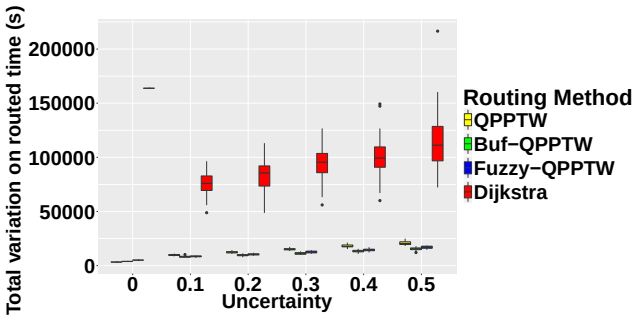


Figure 12: Total variation of simulated times on routed times, for traffic density 1.0, including results for Dijkstra. Each group of four bars has the same uncertainty δ.

were computed using Dijkstra's shortest path algorithm, with edge costs being the modal value from our fuzzy taxi time model for each edge.

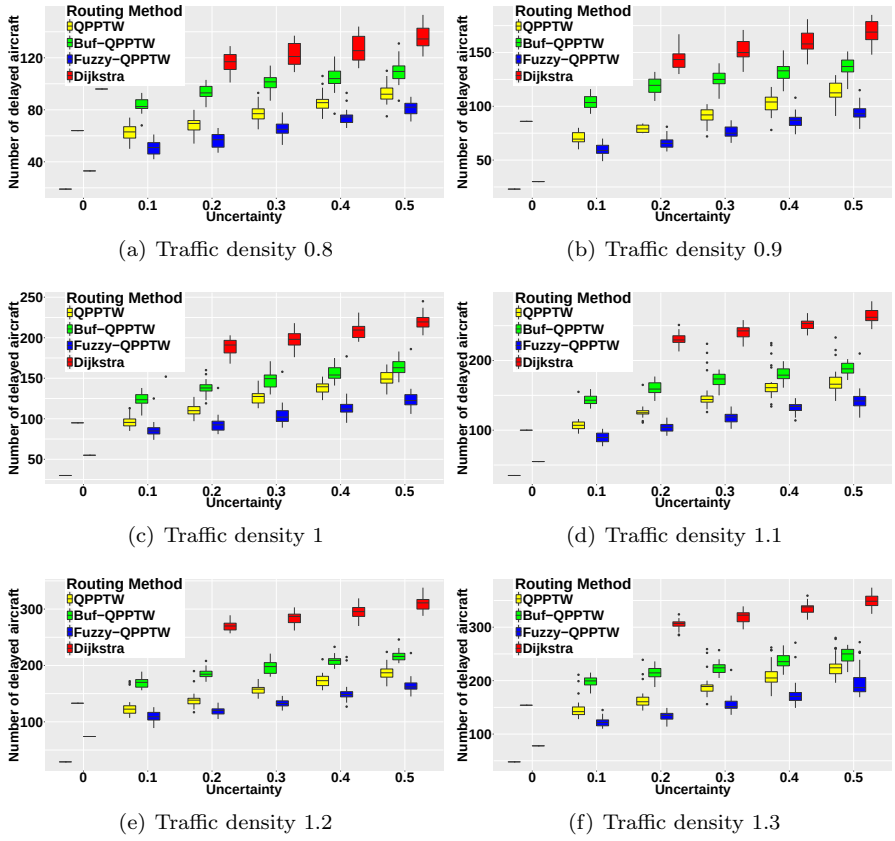
The four algorithms were each run to generate routes for all aircraft on each traffic density. These routes were then run through the simulator 30 times using different seeds for the random number generator (the same 30 seeds being used for each algorithm). Simulator runs were also repeated over six uncertainty levels δ: 0.0 to 0.5 in 0.1 increments.

During each simulator run, the simulated taxi times were recorded for each aircraft. Also recorded for each aircraft was the time that it would have taken to follow the route allocated by QPPTW (the *routed time*) and the time that it would have taken to complete the shortest route (allocated by Dijkstra's algorithm) without stopping (the *minimum time*).

The results of these runs are given in Figs. 11, 12 and 13. Figs. 11 and 12 show the total of simulated times minus routed times over all aircraft (their variation on the routed times). Fig. 13 shows the number of aircraft that were, according to their simulated times, delayed by more than 30s over their routed times. Both metrics are useful. The total variation on routed taxi times reflect the total amount of delay experienced by passengers, how efficiently the taxiways are being used, and are directly related to the fuel consumed for taxiing. In contrast, the total number of aircraft delayed by more than 30s reveals how many movements will be noticeably impacted by delays.

Figs. 11 and 13 show boxplots for the minimum, maximum and median figures over the 30 repeats of the simulation for each scenario and level of uncertainty. Yellow boxes are for QPPTW, green for Buf-QPPTW, blue for Fuzzy-QPPTW, and red for Dijkstra.

It is first worth noting that in 21 individual runs for Buf-QPPTW (spread over traffic densities of 1.0 and higher), and 5 runs for Dijkstra (over densities of 1.2 and higher), several aircraft were unable to complete their movements before the end of the simulator run. The simulator was allowed to run for 15 minutes after the expected completion time of the last movement, so this means that several movements were delayed by more than 15 minutes due to conflicts with



(a) Traffic density 0.8

(b) Traffic density 0.9

(c) Traffic density 1

(d) Traffic density 1.1

(e) Traffic density 1.2

(f) Traffic density 1.3

Figure 13: Number of aircraft delayed (per the simulated times) by more than 30s over the routed time as uncertainty increases, at each traffic density. Each group of four bars has the same uncertainty δ.

other aircraft. To avoid skewing the aggregated results, these were omitted from the figures. For QPPTW and Fuzzy-QPPTW, all aircraft were able to complete their movements in all runs.

Furthermore, the variations on routed times for Dijkstra were so much higher than the other algorithms (see Figure 12) that we have omitted them from Figure 11 to keep the scale spread out enough to distinguish between the other routing algorithms. Figure 12 only gives the results for traffic density 1.0, but the trend was the same for all densities in the experiment.

In Fig. 11, the total variations on the routed times are all positive, reflecting delays. Both Figs. 11 and 13 give a similar picture: as uncertainty in the taxi times increases, so do delays. Buf-QPPTW and Fuzzy-QPPTW show similar levels of total delay over the routed time (Fig. 11), both having 10-20% less total delay than QPPTW. For all levels of uncertainty, Fuzzy-QPPTW produces more robust routes than both QPPTW and Buf-QPPTW (Fig. 13). This means

Table 2: Total distance for routes allocated

Traffic Density	Distance for routes (m)				Fuzzy-QPPTW % increase over		
	QPPTW	Buf-QPPTW	Fuzzy-QPPTW	Dijkstra	QPPTW	Buf-QPPTW	Dijkstra
0.80	1 064 608	1 070 608	1 074 528	1 057 295	0.93	0.37	1.62
0.90	1 113 171	1 121 036	1 122 446	1 105 170	0.83	0.13	1.56
1.00	1 243 064	1 252 805	1 270 889	1 232 931	2.24	1.44	3.08
1.10	1 386 145	1 395 537	1 398 004	1 372 253	0.86	0.18	1.88
1.20	1 517 227	1 533 419	1 537 102	1 500 897	1.31	0.24	2.41
1.30	1 631 457	1 646 840	1 644 432	1 611 860	0.80	−0.15	2.02

that fewer aircraft are stopping due to other aircraft in their path. The same trend was also observed for the simulated times compared to the minimum times for each aircraft, so the improved robustness does not come at the cost of longer taxi durations.

For each traffic scenario, where there is no uncertainty, a couple of observations may be made. The boxplots show no variation (i.e., the median, minimum and maximum are equal) because without uncertainty, the simulation runs identically over the 30 repeats. The routes generated by QPPTW almost always show substantially less delay than those for Fuzzy-QPPTW. This is because QPPTW allocates the quickest routes assuming fixed taxi times, and without uncertainty added to the simulator, the routes taken by the aircraft will match these. Delays over the routed times in this situation are unavoidable, being caused simply by the high traffic levels. Buf-QPPTW and Dijkstra have the most delayed aircraft (Fig. 13) and substantially so for low traffic levels. For Buf-QPPTW, this is because the extra time reserved for movements is never used when there is zero uncertainty. For Dijkstra, this is because there is no planning for the movements of other aircraft. In both cases, this means that no aircraft follows the expected timings, leading to more aircraft conflicts than would otherwise be the case.

Fuzzy-QPPTW allocates routes with some expectation of uncertainty. In order to make these routes more robust, they are more conservative, and occasionally longer than those allocated by QPPTW and Buf-QPPTW. The total length of all routes allocated by each algorithm is shown in Table 2. The total for those allocated by Fuzzy-QPPTW is around 1-2% longer than for those allocated by QPPTW. They are much closer in length to the routes allocated by Buf-QPPTW, and are shorter for the highest traffic scenario. It is also interesting to note that the routes allocated by Dijkstra are only fractionally shorter than those allocated by QPPTW: only a small increase in length is needed to avoid many of the aircraft conflicts.

Overall, Fuzzy-QPPTW allocated routes that are more *achievable* in the presence of uncertainty. While the routes tend to be slightly longer, aircraft will be able to complete their allocated route without stopping or slowing as often to avoid other aircraft. The number of stops and associated acceleration

Table 3: Run time (seconds) for route allocation step

Density	Aircraft count	QPPTW	Buf-QPPTW	Fuzzy-QPPTW
0.8	476	3.10	3.75	737
0.9	518	3.36	4.04	635
1.0	578	3.49	4.90	744
1.1	636	4.12	5.70	1654
1.2	694	4.31	6.56	3440
1.3	752	4.88	7.60	1732

events together with the taxi time, which includes waiting, are significant factors contributing to poor fuel consumption [84]. Therefore, it can be expected that routes generated by Fuzzy-QPPTW will result in less fuel consumption by the taxiing aircraft, compared to QPPTW and Buf-QPPTW. This represents a trade-off, from having more predictable route timings (Fuzzy-QPPTW), shorter routes that likely involve more stopping (QPPTW), or somewhere between (Buf-QPPTW).

It is also worth noting that, to the best of our knowledge, there is no systematic way to determine the buffer size for Buf-QPPTW. As documented in Fig. 13b, Buf-QPPTW is sensitive to buffer setting. If not set properly, it will result in worse performance, comparable to QPPTW in this case. In contrast, Fuzzy-QPPTW gives a systematic way driven by historical aircraft movements and tailored for individual aircraft, airport and operating mode, as determined by explanatory variables in the adaptive Mamdani FRBS described in Section 4.1.

Finally, we consider the run-time for the approaches. Table 3 gives the mean run-time in seconds for each algorithm on each traffic density. These times were achieved on an Intel i7 3820 CPU @3.6GHz with 16GB RAM and no CPU-intensive background processes. Despite this, there is still substantial variability in the run-times. Fuzzy-QPPTW has considerable overhead compared to the other approaches: runs take 3-5s for QPPTW, 3-9s for Buf-QPPTW, and 600-3400s for Fuzzy-QPPTW. Profiling the Java Virtual Machine while the experiments are running suggests this is mostly due to garbage collection of the numerous objects associated with the time-windows as they are updated. However, these figures are for routing 400-800 aircraft: around 5s each. Although the approach is not intended for real-time use (the idea being to run ahead of time and generating routes that are robust to changes that occur in the interim), this is still within the 10s per aircraft that the ICAO requires for online routing and scheduling [85]. The times could also be improved by running the 11 route finding steps in parallel (they were run sequentially in our experiments, but are independent).

8. Conclusions

In this paper, an adaptive Mamdani fuzzy rule based system has been developed as the first attempt for accurate estimation of taxiing times and their associated uncertainty. Theoretically, if each rule only accounts for one taxiing scenario the resulting output membership function will have high certainty at the modal value (i.e. the estimated taxi time) and a support which gives good estimation of uncertainty. However, due to the high number of explanatory variables involved in this work, it is inevitable that several rules will be fired simultaneously. This situation will be intensified when there are no mechanisms in the optimisation procedure to mitigate such correlation between different rules, leading to a big support and low certainty for most of the taxiing scenarios (in the extreme case, this means uncertainty everywhere). Indeed, the quality of estimation of uncertainty is compromised in this situation in exchange for a better estimation of taxi times. Simply adding a constraint on the support may solve the problem at the expense of accuracy in taxi time estimation, which in turn will affect uncertainty estimation. Therefore, a multi-objective adaptive Mamdani FRBS, which can simultaneously optimise the structure of the rule base as well as the estimation accuracy [69], deserves more investigation in future studies.

The second major contribution of this paper is Fuzzy-QPPTW, an extension of the existing QPPTW algorithm, to handle the fuzzy taxi time estimates provided by the adaptive Mamdani FRBS. This was applied to meeting routing requests for real historical aircraft movements at Manchester Airport. Simulations tested the impact of increasing levels of uncertainty on the conflicts and delays experienced by aircraft. Fuzzy-QPPTW produced more conservative taxi routes than QPPTW or Buf-QPPTW, being 1-2% longer in distance on average. However, these routes were more robust, being less disrupted by uncertainty in the taxi times and reducing delays due to other aircraft by 10-20% for higher uncertainty levels. This leads to less stopping and starting of taxiing aircraft, reducing fuel consumption and making better use of the congested airport taxiways. Ultimately this represents a strategic decision on the preferred point in the trade-off between faster or more predictable routes. This trade-off will be highly dependent on airport layout and air traffic patterns, and further research is needed to consider this in more detail. The approach is dependent on the order in which aircraft are routed, particularly when integrating runway crossings where arrivals can block departures and vice versa. Further consideration of this issue forms part of the next stage of this research. Furthermore, it is conceivable that the approach as it stands will fail when departures are held for long enough that arrivals allocated to the same stand will be blocked. The only real solution to this is integration of routing with both gate/stand allocation [24, 25] and runway sequencing [26, 47] together, which represents a challenging direction for future work.

9. Data Access Statement

Final URLs to the data sets used in this paper will be provided here.

10. Acknowledgements

Work funded by UK EPSRC [grants EP/H004424/2, EP/J017515/1, EP/N029496/2, and EP/N029577/1]. Results obtained using the EPSRC funded ARCHIE-WeSt HPC [EPSRC grant EP/K000586/1]. We are also grateful to Manchester and Zurich Airports for valuable advice and data provision.

[1] EUROCONTROL, Challenges of growth 2013 - task 4: European air traffic in 2035 (2013).
URL http://www.eurocontrol.int/sites/default/files/article/content/documents/official-documents/reports/201306-challenges-of-growth-2013-task-4.pdf

[2] European Commission, Roadmap to a single european transport area: Towards a competitive and resource efficient transport system. white paper (2011).
URL http://ec.europa.eu/transport/themes/strategies/2011_white_paper_en.htm

[3] J. A. D. Atkin, Airport airside optimisation problems, in: A. Uyar, E. Ozcan, N. Urquhart (Eds.), Automated Scheduling & Planning, Vol. 505 of Stud. in Comp. Intell., Springer, 2013, pp. 1–37. doi:10.1007/978-3-642-39304-4_1.

[4] J. A. D. Atkin, E. K. Burke, S. Ravizza, The Airport Ground Movement Problem: Past & Current Research and Future Directions, in: Int. Conf. on Research in Air Transportation, 2010, pp. 131–138.

[5] J. A. Bennell, M. Mesgarpour, C. N. Potts, Airport runway scheduling, 4OR 9 (2) (2011) 115–138. doi:10.1007/s10288-011-0172-x.

[6] G. Slveling, J.-P. Clarke, Scheduling of airport runway operations using stochastic branch and bound methods, Transp. Res. Pt. C: Emerg. Tech. 45 (2014) 119 – 137, advances in Computing and Communications and their Impact on Transportation Science and Technologies. doi:https://doi.org/10.1016/j.trc.2014.02.021.

[7] U. Dorndorf, A. Drexl, Y. Nikulin, E. Pesch, Flight gate scheduling: State-of-the-art & recent developments, Omega 35 (3) (2007) 326–334.

[8] M. Dell'Orco, M. Marinelli, M. G. Altieri, Solving the gate assignment problem through the fuzzy bee colony optimization, Transp. Res. Pt. C: Emerg. Tech. 80 (2017) 424 – 438. doi:https://doi.org/10.1016/j.trc.2017.03.019.

[9] Honeywell, Electric green taxiing system (2013).
URL www.greentaxiing.com/resources/EGTS_Positioning_paper.pdf

[10] L. Hao, M. S. Ryerson, L. Kang, M. Hansen, Estimating fuel burn impacts of taxi-out delay with implications for gate-hold benefits, Transp. Res. Pt. C: Emerg. Tech. 80 (2017) 454 – 466. doi:https://doi.org/10.1016/j.trc.2016.05.015.

[11] Y. Jiang, Z. Liao, H. Zhang, A collaborative optimization model for ground taxi based on aircraft priority, Math Probl Eng 2013 (2013) 1–9. doi:10.1155/2013/854364.

[12] K. Yin, C. Tian, B. X. Wang, L. Quadrifoglio, Analysis of Taxiway Aircraft Traffic at George Bush Intercontinental Airport, Houston, Texas, Transp. Res. Record 2266 (1) (2012) 85–94.

[13] J. Koeners, R. Rademaker, Creating a simulation environment to analyze benefits of real-time taxi flow optimization using actual data, AIAA Modeling & Simulation Technologies Conf.doi:10.2514/6.2011-6372.

[14] D. M. Pfeil, H. Balakrishnan, Identification of robust terminal-area routes in convective weather, Transport Sci 46 (1) (2012) 56–73.

[15] H. Khadilkar, H. Balakrishnan, Network congestion control of airport surface operations, J Guid Control Dyn 37 (3) (2014) 933–940.

[16] S. Ravizza, J. Chen, J. A. D. Atkin, P. Stewart, E. K. Burke, Aircraft taxi time prediction: Comparisons & insights, Applied Soft Computing 14 (Part C) (2014) 397 – 406. doi:10.1016/j.asoc.2013.10.004.

[17] H. Lee, H. Balakrishnan, Fast-time simulations of Detroit Airport operations for evaluating performance in the presence of uncertainties, IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)doi:10.1109/dasc.2012.6382349.

[18] I. Simaiakis, H. Balakrishnan, A queuing model of the airport departure process, Transp. Science 50 (1) (2016) 94–109.

[19] T. V. Truong, The distribution function of airport taxi-out times & selected applications, J. Tran. Res. For. 50 (2) (2012) 33–44.

[20] T. K. Simić, O. Babić, Airport traffic complexity & environment efficiency metrics for evaluation of ATM measures, J. of Air Transport Management 42 (2015) 260 – 271. doi:http://dx.doi.org/10.1016/j.jairtraman.2014.11.008.

[21] J. Guépet, O. Briant, J. Gayon, R. Acuna-Agost, The aircraft ground routing problem: Analysis of industry punctuality indicators in a sustainable perspective, European Journal of Operational Research 248 (3) (2016) 827–839. doi:10.1016/j.ejor.2015.08.041.

[22] C. Evertse, H. Visser, Real-time airport surface movement planning: Minimizing aircraft emissions, Transportation Research Part C: Emerging Technologies 79 (2017) 224–241. doi:10.1016/j.trc.2017.03.018.

[23] T. Zhang, M. Ding, B. Wang, Q. Chen, Conflict-free time-based trajectory planning for aircraft taxi automation with refined taxiway modeling, Journal of Advanced Transportation 50 (3) (2015) 326–347. doi:10.1002/atr.1324.

[24] C. Yu, D. Zhang, H. H. Lau, A heuristic approach for solving an integrated gate reassignment and taxi scheduling problem, Journal of Air Transport Management 62 (2017) 189–196. doi:10.1016/j.jairtraman.2017.04.006.

[25] J. A. Behrends, J. M. Usher, Aircraft gate assignment: Using a deterministic approach for integrating freight movement and aircraft taxiing, Comput. & Indust. Eng. 102 (2016) 44–57. doi:10.1016/j.cie.2016.10.004.

[26] J. Guépet, O. Briant, J.-P. Gayon, R. Acuna-Agost, Integration of aircraft ground movements and runway operations, Transportation Research Part E: Logistics and Transportation Review 104 (2017) 131–149. doi:10.1016/j.tre.2017.05.002.

[27] R. Morris, C. S. Pasareanu, K. S. Luckow, W. Malik, H. Ma, T. S. Kumar, S. Koenig, Planning, scheduling and monitoring for airport surface operations, in: Proceedings of the 13th AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA, 2016, pp. 608–614.

[28] C. Stergianos, J. Atkin, P. Schittekat, T. E. Nordlander, C. Gerada, H. Morvan, The importance of considering pushback time and arrivals when routing departures on the ground at airports, in: Proceedings of ICAOR 2016, Vol. 8 of Lecture Notes in Management Science, Springer, Rotterdam, The Netherlands, 2016, pp. 41–46.

[29] L. Yang, S. Yin, K. Han, J. Haddad, M. Hu, Fundamental diagrams of airport surface traffic: Models and applications, Transportation Research Part B: Methodologicaldoi:10.1016/j.trb.2017.10.015.

[30] S. Ravizza, J. A. D. Atkin, E. K. Burke, A more realistic approach for airport ground movement optimisation with stand holding, Journal of Scheduling 17 (5) (2013) 507–520. doi:10.1007/s10951-013-0323-3.

[31] L. Yang, K. Li, Z. Gao, Train timetable problem on a single-line railway with fuzzy passenger demand, IEEE T. Fuzzy Syst. 17 (3) (2009) 617–629. doi:10.1109/tfuzz.2008.924198.

[32] S.-C. Huang, M.-K. Jiau, C.-H. Lin, Optimization of the carpool service problem via a fuzzy-controlled genetic algorithm, IEEE T. Fuzzy Syst. 23 (5) (2015) 1698–1712. doi:10.1109/tfuzz.2014.2374194.

[33] P. Fortemps, Jobshop scheduling with imprecise durations: a fuzzy approach, IEEE T. Fuzzy Syst. 5 (4) (1997) 557–569. doi:10.1109/91.649907.

[34] F.-T. Lin, Fuzzy job-shop scheduling based on ranking level (λ, 1) interval-valued fuzzy numbers, IEEE T. Fuzzy Syst. 10 (4) (2002) 510–522. doi:10.1109/tfuzz.2002.800659.

[35] S. Petrovic, X. Song, A new approach to two-machine flow shop problem with uncertain processing times, Optim Eng 7 (3) (2006) 329–342. doi:10.1007/s11081-006-9975-6.

[36] J.-B. Gotteland, N. Durand, Genetic algorithms applied to airport ground traffic optimization, in: Proc. IEEE CEC, Canberra, Australia, 2003, pp. 544–551. doi:10.1109/CEC.2003.1299623.

[37] J.-B. Gotteland, N. Durand, J.-M. Alliot, E. Page, Aircraft ground traffic optimization, in: Proc. Int'l Air Traf. Mgmnt. R&D Seminar, 2001.

[38] B. Pesic, N. Durand, J.-M. Alliot, Aircraft ground traffic optimisation using a genetic algorithm, in: Proc. GECCO, 2001.

[39] G. L. Clare, A. G. Richards, Optimization of taxiway routing & runway scheduling, IEEE T Intell Tran Syst 12 (4) (2011) 1000–1013. doi:10.1109/TITS.2011.2131650.

[40] M. Samà, A. D'Ariano, F. Corman, D. Pacciarelli, Coordination of scheduling decisions in the management of airport airspace and taxiway operations, Transportation Research Procedia 23 (2017) 246–262, papers Selected for the 22nd International Symposium on Transportation and Traffic Theory Chicago, Illinois, USA, 24-26 July, 2017. doi:10.1016/j.trpro.2017.05.015.

[41] J. Ma, D. Delahaye, M. Sbihi, M. Mongeau, Integrated Optimization of Terminal Manoeuvring Area and Airport , in: EUROCONTROL (Ed.), 6th SESAR Innovation Days (2016) . , Proceedings of the SESAR Innovation Days 2016, Delft, Netherlands, 2016, pp. ISSN 0770–1268.
URL https://hal-enac.archives-ouvertes.fr/hal-01404006

[42] J. Chen, M. Weiszer, P. Stewart, M. Shabani, Toward a more realistic, cost-effective, and greener ground movement through active routing–part i: Optimal speed profile generation, IEEE Trans. on Intel. Transp. Systems 17 (5) (2016) 1196–1209. doi:10.1109/TITS.2015.2477350.

[43] J. Chen, M. Weiszer, G. Locatelli, S. Ravizza, J. A. Atkin, P. Stewart, E. K. Burke, Toward a more realistic, cost-effective, and greener ground movement through active routing: A multiobjective shortest path approach, IEEE Trans. on Intell. Transp. Systems 17 (12) (2016) 3524–3540. doi:10.1109/TITS.2016.2587619.

[44] M. Weiszer, J. Chen, P. Stewart, A real-time active routing approach via a database for airport surface movement, Transp. Res. Pt. C: Emerg. Tech. 58 (2015) 127 – 145. doi:https://doi.org/10.1016/j.trc.2015.07.011.

[45] L. Adacher, M. Flamini, E. Romano, Airport ground movement problem: Minimization of delay and pollution emission, IEEE Transactions on Intelligent Transportation Systems PP (99) (2018) 1–10. doi:10.1109/TITS.2017.2788798.

[46] O. E. Guclu, C. Cetek, Analysis of aircraft ground traffic flow and gate utilisation using a hybrid dynamic gate and taxiway assignment algorithm, The Aeronautical Journal 121 (1240) (2017) 721745. doi:10.1017/aer.2017.20.

[47] U. Benlic, A. E. I. Brownlee, E. K. Burke, Heuristic search for the coupled runway sequencing & taxiway routing problem, Trans Res C: Emerg Tech 71 (2016) 333–355.

[48] C. Lesire, An iterative A* algorithm for planning of airport ground movements, in: Proc. Euro. Conf. on AI, IOS Press, Amsterdam, Netherlands, 2010, pp. 413–418.

[49] P. Balakrishna, R. Ganesan, L. Sherry, Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures, Transport Res. C: Emerg. Tech. 18 (6) (2010) 950–962. doi:10.1016/j.trc.2010.03.003.

[50] S. Ravizza, J. Atkin, M. Maathuis, E. Burke, A combined statistical approach & ground movement model for improving taxi time estimations at airports, J Oper Res Soc 64 (9) (2013) 1347–1360. doi:10.1057/jors.2012.123.

[51] J. Chen, S. Ravizza, J. A. D. Atkin, P. Stewart, On the utilisation of fuzzy rule-based systems for taxi time estimations at airports, in: Wkshp on Algor. Appr. for Transp. Model., Opt., & Syst., 2011, pp. 134–145. doi:10.4230/OASIcs.ATMOS.2011.134.

[52] O. Lordan, J. M. Sallan, M. Valenzuela-Arroyo, Forecasting of taxi times: The case of barcelona-el prat airport, Journal of Air Transport Management 56 (2016) 118 – 122, growing airline networks -Selected papers from the 18th ATRS World Conference, Bordeaux, France, 2014. doi:https://doi.org/10.1016/j.jairtraman.2016.04.015.
URL http://www.sciencedirect.com/science/article/pii/S0969699716301570

[53] H. Khadilkar, H. Balakrishnan, Integrated control of airport and terminal airspace operations, IEEE Transactions on Control Systems Technology 24 (1) (2016) 216–225. doi:10.1109/TCST.2015.2424922.

[54] J. Atkin, G. D. Maere, E. Burke, J. Greenwood, Addressing the pushback time allocation problem at heathrow airport, Transportation Science 47 (4) (2013) 584–602.

[55] D. Rappaport, P. Yu, K. Griffin, C. Daviau, Quantitative Analysis of Uncertainty in Airport Surface Operations, AIAA, 2009, pp. 1–16. doi:10.2514/6.2009-6987.

[56] Y. Liu, M. Hansen, G. Gupta, W. Malik, Y. Jung, Predictability impacts of airport surface automation, Transp. Res. Pt. C: Emerg. Tech. 44 (2014) 128 – 145. doi:https://doi.org/10.1016/j.trc.2014.03.010.

[57] M. Gendreau, G. Laporte, R. Séguin, Stochastic vehicle routing, Euro J Oper Res 88 (1) (1996) 3–12.

[58] U. Ritzinger, J. Puchinger, R. F. Hartl, A survey on dynamic & stochastic vehicle routing problems, Int J Prod Res 54 (1) (2016) 215–231.

[59] L. M. Hvattum, A. Løkketangen, G. Laporte, Solving a dynamic & stochastic vehicle routing problem with a sample scenario hedging heuristic, Transp Sci 40 (4) (2006) 421–438. doi:10.1287/trsc.1060.0166.

[60] N. Ando, E. Taniguchi, Travel time reliability in vehicle routing and scheduling with time windows, Netw Spat Econ 6 (3-4) (2006) 293–311. doi:10.1007/s11067-006-9285-8.

[61] S. Demeyer, P. Audenaert, M. Pickavet, P. Demeester, Dynamic & stochastic routing for multimodal transportation systems, IET Intel Transp Sys 8 (2) (2014) 112–123. doi:10.1049/iet-its.2012.0065.

[62] D. Taş, M. Gendreau, N. Dellaert, T. Van Woensel, A. De Kok, Vehicle routing with soft time windows & stochastic travel times: A column generation and branch-and-price solution approach, Euro J Oper Res 236 (3) (2014) 789–799.

[63] C. Verbeeck, P. Vansteenwegen, E.-H. Aghezzaf, Solving the stochastic time-dependent orienteering problem with time windows, Euro J Oper Res 255 (3) (2016) 699 – 718.

[64] E. Angelelli, C. Archetti, C. Filippi, M. Vindigni, The probabilistic orienteering problem, Comput Oper Res 81 (2017) 269 – 281. doi:http://dx.doi.org/10.1016/j.cor.2016.12.025.

[65] J. A. D. Atkin, E. K. Burke, J. S. Greenwood, TSAT allocation at London Heathrow: the relationship between slot compliance, throughput and equity, Public Transport 2 (3) (2010) 173–198. doi:10.1007/s12469-010-0029-2.

[66] B. Kosko, Fuzzy systems as universal approximators, IEEE T Computers 43 (11) (1994) 1329–1333.

[67] K. M. Passino, S. Yurkovich, M. Reinfrank, Fuzzy control, Vol. 20, Addison-wesley, Menlo Park, CA, 1998.

[68] J. Chen, M. Gallimore, C. Bingham, M. Mahfouf, Y. Zhang, et al., Intelligent data compression, diagnostics and prognostics using an evolutionary-based clustering algorithm for industrial machines, in: Fault detection: classification, techniques & role in industrial systems, NOVA Science Publisher, New York, USA, 2014.

[69] J. Chen, M. Mahfouf, Improving transparency in approximate fuzzy modeling using multi-objective immune-inspired optimisation, Int J Comput Intel Sys 5 (2) (2012) 322–342. doi:10.1080/18756891.2012.685311.

[70] H. Idris, J. Clarke, R. Bhuva, L. Kang, Queuing model for taxi-out time estimation, Air Traf Cont Quart 10 (1) (2002) 1–22.

[71] J.-S. Jang, C.-T. Sun, Neuro-fuzzy modeling and control, Proceedings of the IEEE 83 (3) (1995) 378–406.

[72] D. Dubois, H. T. Nguyen, H. Prade, Possibility theory, probability and fuzzy sets misunderstandings, bridges and gaps, in: Fundamentals of fuzzy sets, Springer, 2000, pp. 343–438.

[73] C. S. McCahon, E. S. Lee, Job sequencing with fuzzy processing times, Comput Math Appl 19 (7) (1990) 31–41. doi:10.1016/0898-1221(90)90191-l.

[74] S. M. Johnson, Optimal 2- & 3-stage production schedules with setup times included, Nav Res Log Quart 1 (1) (1954) 61–68. doi:10.1002/nav.3800010110.

[75] W. Pedrycz, F. Gomide, An Introduction to Fuzzy Sets, MIT Press, Cambridge, Mass., 1998.

[76] R. R. Yager, A procedure for ordering fuzzy subsets of the unit interval, Info Sci 24 (2) (1981) 143 – 161. doi:http://dx.doi.org/10.1016/0020-0255(81)90017-7.

[77] B. Stenzel, Online disjoint vehicle routing with application to AGV routing, Ph.D. thesis, TU Berline, Germany (2008).

[78] A. E. I. Brownlee, J. A. D. Atkin, J. A. W. Woodward, U. Benlic, E. K. Burke, Airport ground movement: Real world data sets & approaches to handling uncertainty, in: Proc. PATAT, York, UK, 2014.

[79] C. Petersen, M. Mühleisen, A. Timm-Giel, Evaluation of the aircraft distribution in satellite spotbeams, in: T. Bauschert (Ed.), Adv. in Comm. Networking, Vol. 8115 of LNCS, Springer, 2013, pp. 46–53. doi:10.1007/978-3-642-40552-5_5.

[80] R. Turner, S. Bottone, C. Stanek, Online variational approximations to non-exponential family change point models: With application to radar tracking, in: Proc. of NIPS 26, 2013, pp. 306–314.

[81] P. Ptak, J. Hartikka, M. Ritola, T. Kauranne, Long-distance multistatic aircraft tracking with VHF frequency doppler effect, IEEE T Aero Elec Sys 50 (3) (2014) 2242–2252. doi:10.1109/taes.2014.130246.

[82] A. J. Eele, J. M. Maciejowski, Sequential Monte Carlo Optimisation for Air Traffic Management, Tech. Rep. CUED/F-INFENG/TR.693, Cambridge University Engineering Department (2015).

[83] H. K. Baruah, Ch 6: Construction of normal fuzzy numbers using the mathematics of partial presence, Mathematics of Uncertainty Modeling in the Analysis of Engineering & Science Problems (2014) 109–126doi:10.4018/978-1-4666-4991-0.ch006.

[84] H. Khadilkar, H. Balakrishnan, Estimation of aircraft taxi fuel burn using flight data recorder archives, Transportation Res. D: Transport & Environment 17 (7) (2012) 532–537.

[85] ICAO, Advanced surface movement guidance & control systems (a-smgcs) manual, section 4.3.2 (2004).
URL http://www.icao.int/Meetings/anconf12/Document%20Archive/9830_cons_en[1].pdf

Appendix A. Notation and Process Overview

Table A.4: Summary of Notation

Symbol	Description
A	set of all aircraft to be routed
$G = (V, E)$	graph of taxiways, sets of vertices and edges
e	a single edge
τ	a specific point in time (timestamp)
t	a period of time
T_e	set of weights (taxi times) for e
t_{e_L}	a specific taxi time for e, given the previous label L and current aircraft
$conf(e)$	set of conflicting edges for e
$\mathcal{F}(e)$	set of time windows for e
a_i	aircraft and index
$I_L = [a_L, b_L]$	time period for label L
$L = (v_L, I_L, pred_L)$	label on vertex v for QPPTW, $pred_L$ being the previous label
$Q_i = (v_{start}, v_{end}, \tau_{rw})$	request to route aircraft a_i from $v_{start} \in V$ to $v_{end} \in V$, starting at allocated runway time τ_{rw} (arrivals) or ending at τ_{rw} (departures)
τ_{in} and τ_{out}	in and out times for an edge
$R_{\tau_{start}}, R_{\tau_{out}}$	start / exit time for allocated aircraft route R
$\tilde{t} = (l, b, u)$	a fuzzy value (time in this case), with lower bound l, modal b, upper bound u
$\mu_{\tilde{t}}(t)$	membership function for time t
$\tilde{t}^{\alpha}$	alpha cut of t
p_i	position of aircraft a_i on G
u_i	uncertainty applied to aircraft a_i
H_i^j	fuzzy set (a linguistic value) for the j-th explanatory variable of the i-th rule
c_i^j	centre of the Gaussian membership function for the j-th explanatory variable and i-th rule
σ_i^j	spread of the Gaussian membership function for the j-th explanatory variable and i-th rule
x_j	value of the j-th explanatory variable
y_i	output of the i-th rule
σ_i^y	spread of the bell-shaped function for the output of the i-th rule
$\mu_{H_i^j}(x_j)$	membership function associated with H_i^j
$\mu_{B_i}(y)$	output membership function for the i-th rule
Z_i	consequent of the i-th rule
n	number of explanatory variables
r	number of rules in adaptive Mamdani FRBS
δ	the uncertainty level applying to one run of the simulator

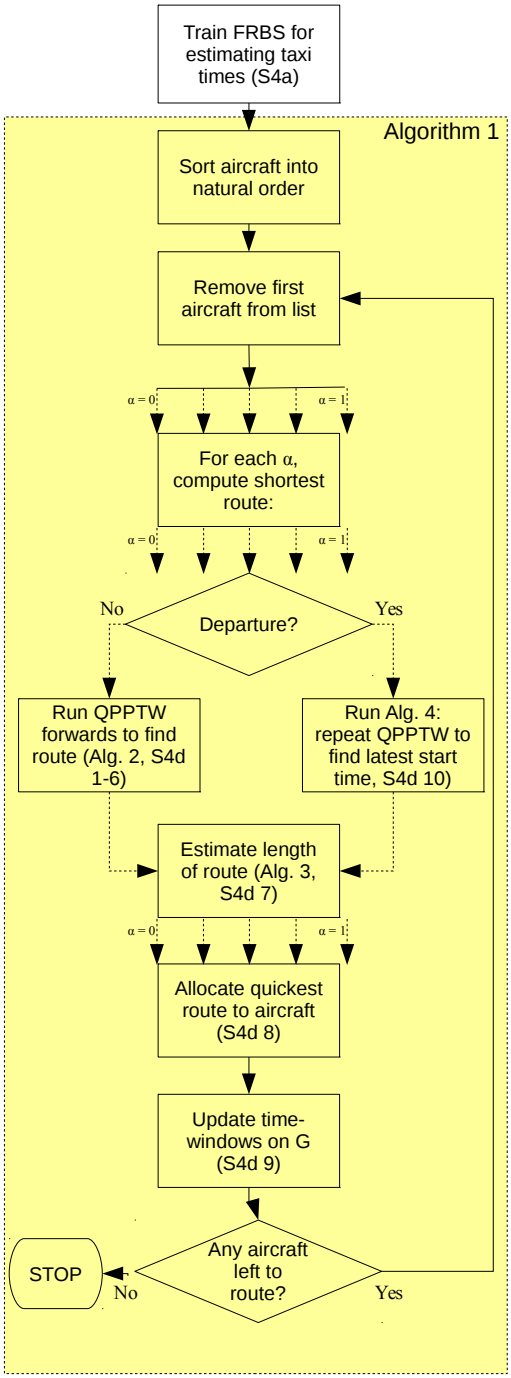


Figure A.14: Overview of the optimisation process, showing how the component algorithms fit together to allocate routes to a set of aircraft. Each stage identifies the algorithm number and section in the text giving more detail.