Heuristic search for the coupled runway sequencing and taxiway routing problem

Una Benlic [a,c], Alexander E.I. Brownlee [b], Edmund K. Burke [a]

[a] *School of Electronic Engineering and Computer Science, Queen Mary University of London, London, email: {u.benlic,e.burke}@qmul.ac.uk*

[b] *CHORDS Research Group, University of Stirling, Stirling FK9 4LA, email: sbr@cs.stir.ac.uk*

[c] *University of Electronic Science and Technology of China, North Jianshe Road, Sichuan 610054*

Abstract

This paper presents the first local search heuristic for the coupled runway sequencing (arrival & departure) and taxiway routing problems, based on the receding horizon (RH) scheme that takes into account the dynamic nature of the problem. As test case, we use Manchester Airport, the third busiest airport in the UK. From the ground movement perspective, the airport layout requires that departing aircraft taxi across the arrivals runway. This makes it impossible to separate arrival from departure sequencing in practice. Operationally, interactions between aircraft on the taxiways could prevent aircraft from taking off from, or landing on, runways during the slots assigned to them by an algorithm optimizing runway use alone. We thus consider the interactions between arrival and departure aircraft on the airport surface. Compared to sequentially optimized solutions, the results obtained with our approach indicate a significant decrease in the taxiway routing delay, with generally no loss in performance in terms of the sequencing delay for a regular day of operations. Another benefit of such a simultaneous optimization approach is the possibility of holding aircraft at the stands for longer, without the engines running. This significantly reduces the fuel burn, as well as bottlenecks and traffic congestion during peak hours that are often the cause of flight delays due to the limited amount of airport surface space available. Given that the maximum computing time per horizon is around 95 seconds, real-time operation might be practical with increased computing power.

Keywords: Runway arrival and departure sequencing, taxiway routing, ground movement, local search.

1 Introduction

Due to a huge increase in the air traffic over the past decade, and with further growth forecast (EUROCONTROL, 2013), air traffic congestion on the airport surface is a major constraint on efficient use of airport resources (runways, taxiways and gates/stands). Economically, congestion could cause airborne delays, whereas environmentally, it results in an increase in air pollutants and noise emissions. Even though the expansion of airport capacity and the increase of traffic controllers are the most obvious solutions, these are often not realistic due to cost and space limitations. A more practical solution is the use of highly innovative decision support systems for an effective management of existing resources. This has led research programs, such as the Single European Sky ATM Research (SESAR) project in Europe and NextGEN in the United States, to initiate the development of highly complex decision support systems based on sophisticated optimization methodologies.

The existing work in the field on transportation research has mainly focused on solving the key airport operations problems in isolation. These include gate allocation, runway sequencing, ground movement and baggage handling, with the relative importance of each varying by airport. Nevertheless, there are obvious benefits from considering different airport operations as a unit from both economical and environmental point of view (Atkin et al., 2010). As discussed in (Atkin et al., 2010), the ground movement problem is that of allocating efficient taxi routes to aircraft moving between the runways and stands. Ground movement forms the link between other airport operations problems, including arrival sequencing, departure sequencing and gate allocation. Indeed, an optimal departure sequence is of no use if aircraft cannot reach the runway at allocated take-off times. In real operations, aircraft typically leave the gates to meet on time their departure slot as soon as they are ready for pushback. Since the amount of available airport surface is limited, this results in bottlenecks and traffic congestion during peak hours causing flight delays. Furthermore, arrival aircraft can have a significant effect on ground movement planning, especially at airports where runway crossing is necessary for taxiing aircraft(Mirković et al., 2016). Therefore, prior knowledge of the landing times is required for realistic runway sequencing and ground movement optimization.

Some effort has recently been made on the design of approaches for tackling multiple problems simultaneously. For instance, the authors in (Atkin et al., 2007) present a decision-support system based on heuristic search for the departure runway scheduling at Heathrow. A tabu search algorithm is used for finding good take-off orders, which are then tested for feasibility given the holding-point restrictions. In (Roling and Visser, 2008), the authors propose a mixed-integer linear programming approach that aims to optimize a weighted combination of the total taxi time and the total holding time in such a way to deconflict the taxi plans. The work in (Montoya et al., 2011) presents a dynamic programming algorithm for op-

timal runway sequencing. In addition to miles-in-trail and wake vortex separation constraints, runway crossings are also taken into account. In (Clare and Richards, 2011), the authors describe an automated tool that incorporates departure runway scheduling with taxiway routing in continuous time at Heathrow, based on a receding horizon technique. The method adopts MILP optimization, while the proposed model imposes the runway separation requirements as a constraint to the taxiway routing process. The work in (Jung et al., 2011) describes an airport surface decision support tool that combines departure routing with departure sequencing, and tests the combined solutions using a detailed simulation. In (Atkin et al., 2013), the authors propose a two-stage approach that finds a take-off sequence in the first stage, and then uses this in the second stage to calculate push-back times such that an appropriate amount of the delay is absorbed at the stand, prior to starting the engines. The feasibility of the second stage is considered within the first stage. The work in (Weiszer et al., 2015) applies multi-objective optimisation to the integrated problems of departure sequencing (excluding arrivals), ground movement and airport bus scheduling, with results showing improved fuel and time efficiency over treating the problems in isolation. Other research considering combined airport operations problems can be found in (Deau et al., 2008; Lee and Balakrishnan, 2012; Neuman and Atkin, 2013; Nosedal et al., 2015; Weiszer et al., 2015).

In recent years, several models have been proposed for the integrated arrival sequencing, departure sequencing and runway routing problems. Among the first such works is a set partitioning model (Yu and Lau, 2014) that largely reduces the number of constraints and makes the problem more manageable. In the proposed model, each possible aircraft route is regarded as a decision variable, while the constraints enforce a minimum separation distance between aircraft at the taxiways and runways. The proposed method has been tested on a small taxiway layout of 36 nodes with one runway, and a single problem instance that includes 6 aircraft. The time required to reach optimality for the given instance is not reported. In (Bosson et al., 2015), the authors extended a previously developed mixed-integer-linear programming approach for arrival and departure sequencing to include taxiway operations. The approach is applied to a model of the Los Angeles International Airport, and a preliminary case study is conducted on a set of thirteen aircraft. This test case was solved to optimality in about 240 seconds. The work in (Bertsimas and Frankovich, 2015) presents an integer programming model that addresses simultaneously the optimisation of arrival sequencing, departure sequencing and surface routing in a tractable manner. The model is divided into two stages considering the problem complexity. The first stage focuses solely on the runway capacities (i.e., runway sequencing), while the second stage can be viewed as the routing phase which determines a routing of flights to achieve a runway processing schedule close to that obtained in the first stage. The approach results in a suboptimal solution, since runway sequencing and taxiway routing are performed in two separate stages. Along with the above mentioned literature, it is also worth mentioning the Spot and Run-

way Departure Advisor (SARDA) [1] , developed and studied by NASA. SARDA is an algorithm for effective management of departures at an airport, which plans gate departure times, spot crossing times, and runway sequences. So far, SARDA has been applied at three airports in USA, chosen for their diversity in geometric and operational characteristics.

While the works in (Bertsimas and Frankovich, 2015; Bosson et al., 2015; Yu and Lau, 2014) present mathematical models for exact solving of the combined problem, this paper presents probably the first heuristic, based on the Iterated Local Search (ILS) framework, for optimization of the coupled runway sequencing and taxiway routing problems in continuous time. Furthermore our approach incorporates the receding horizon (RH) framework (Bellingham et al., 2003; Hu and Chen, 2005; Hu and Paolo, 2008; Zhan et al., 2010) to take into account the dynamic nature of the problem. As test case, we use Manchester Airport, the third busiest airport in the UK. The airport has two runways, which are operated in segregated mode during busy periods. Manchester Airport has a runway crossing which makes it impossible to separate arrival from departure runway sequencing in practice. The proposed method considers arrival and departure sequencing as two separate yet interrelated processes, where the arrival sequence and the departure sequence are being optimized in a token-ring way. More precisely, the optimization of the arrival sequence is followed by the optimization of the departure sequence, always starting from the best solution found in the previous optimization phase. Each time an arrival or departure sequence S is selected from the neighborhood to replace the current sequence, the algorithm calls a dedicated iterative routing heuristic to determine the best routing solution R for S in terms of the total taxiing delay. In this context, the taxiing delay of an aircraft is defined as the delay of the given aircraft over the unimpeded taxi time for its allocated route. Finally, the acceptance of S is based on a combination of three criteria: (i) the total sequence delay of S; (ii) the number of infeasible sequencing slots in S in terms of the corresponding routing solution R; and (iii) the total taxiing delay of R. We use the term *infeasible sequencing slot* to denote those slots in S that result long taxiing delays in R. Criteria (ii) and (iii) are thus introduced for two reasons, the first being to ensure that departures are able to reach the runway for their allocated departure slot, while holding aircraft at stands for as long as possible. The second reason is to focus optimization on sequencing slots that result long taxiing delays in the corresponding routing solution. Coupling of the two problems is achieved by imposing a sequence feasibility constraint in terms of the ground movement. This helps attain a significant decrease in taxiing delay with a minimal sacrifice of the runway sequence.

For experimental evaluations, we compare our runway sequencing and routing results with sequentially optimized runway sequencing and routing solutions (when taxiway routing is not considered during runway sequencing, while a routing solution is optimized given a runway sequence). Compared to sequentially optimized

[1] http://www.aviationsystemsdivision.arc.nasa.gov/research/surface/sarda.shtml

solutions, the results obtained with our approach indicate that there is generally no loss in performance in terms of the runway sequencing delay for a regular day of operations, in spite of the sequence feasibility constraint imposed by ground movement. Since runway sequences and taxiway routes with the proposed method are performed at the stands, this facilitates the possibility of holding aircraft at the stands for longer, without the engines running, which significantly reduces the fuel burn.

The paper is organized as follows. Section 2 provides a thorough description and formulation of the problem. The proposed approach for the coupled runway sequencing and taxiway routing problems is given in Section 3. Section 4 shows experimental results and comparisons, followed by conclusions in Section 5.

2 Problem description and formulation

This work focuses on the coupled runway sequencing (including both arrivals and departures) and ground movement optimization in continuous time. Maximization of runway throughput and minimization of the total taxi times are considered as primary and secondary objectives respectively. Before describing the approach for simultaneous optimization of the considered problems, we briefly describe the layout of our test case, Manchester Airport, followed by a detailed description and formulation of the problems.

2.1 Manchester Airport

According to CAA statistics ((Civil Aviation Authority, 2013) and (Civil Aviation Authority, 2014)), Manchester is the third busiest airport in the UK in both annual passengers (20.7M) and aircraft movements (159 000). A stylized diagram of the airport is given in Figure 1. It has several interesting features from a ground movement perspective. The airport has three terminals, two runways (05L/23R and 05R/23L) and 148 aircraft stands. Of the stands, 54 are shadowed such that they cannot be used when larger aircraft are on the adjacent stands. 57 stands are served by terminal piers, and 91 are remote (accessed by bus transfer from the terminal). 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 is via two taxiways. Owing to the limited number of terminal stands, aircraft on longer stopovers are often towed to remote stands, placing further demand on the taxiways. In this work, we assume that a gate is available at any given time. Furthermore, we do not consider towing aircraft from one gate to another, as we do not take into account the gate allocation problem. However, this could be an interesting extension for a future work.

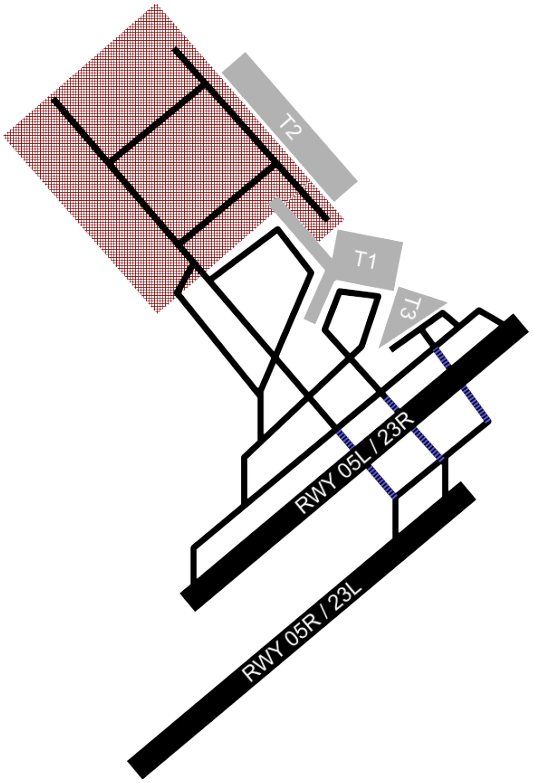


Fig. 1. Stylised diagram of Manchester Airport showing the terminals (grey), runways (black) and major taxiways (black). The apron (red hatched) can only be reached by two taxiways: this bottleneck restricts access to around half of the airport's stands, including those for T2, part of T1, and remote stands. The taxiways coloured with blue strips are crossings of runway 05R/23L: during busy periods in two-runway operating mode, all departures must make this runway crossing, timed to take place between arrivals.

The majority of the time (with the exception of unusual prevailing wind conditions), the airport switches between two operating modes over the course of a day. In mixed runway mode, the airport uses runway 05L/23R for both arrivals and departures. In segregated runway mode (busy periods in the daytime only), runway 05L/23R is used for arrivals and 05R/23L is used for departures. Given that busy periods are the most critical for airport management, this work considers the segregated runway mode.

2.2 Runway sequencing

Given a set of aircraft for landing / take-off, the aim of the runway sequencing problem (Dear, 1976; Psaraftis, 1978) is to increase the runways' throughput, while guaranteeing a minimal separation between aircraft to ensure safety and control congestion. Despite some similarities between arrival and departure sequencing (both can be modelled as machine scheduling problems), there are important differences between these processes in practice, lying in the details of the separation

rules, constraints, and objective functions. There exist different variations of the runway sequencing problem, depending on the number of runways considered, and depending on the mode in which the individual runways are managed (either segregated or mixed). As mentioned previously, we focus on two-runway operations in segregated mode. The runway sequencing constraints and objectives are detailed in the following sections.

2.2.1 Runway separations

The runway operations are constrained by three types of separations: (i) the wake-vortex separation; (ii) the aircraft speed separation; and (iii) the en-route separation.

The wake-vortex separation is to ensure that wake vortices left by the earlier aircraft i have dissipated before the landing / take-off of the following aircraft j. It is determined by the aircraft size and the relative positions of the two aircraft. This separation generally differs for arrivals and departures. Table 1 shows a standard for the wake-vortex separation in case of arrivals and departures. The separations adopted for arrivals are taken from (Bianco et al., 1997), while the departure wake-vortex separations are in accordance with the current regulations at the Manchester Airport [2] . It can be seen that the wake-vortex separation matrix in case of arrivals is asymmetric. Generally speaking, a smaller aircraft following a larger aircraft will require a longer separation time than the other way around. The asymmetric nature of the separation rules implies that shifting aircraft positions in a sequence may reduce delays and increase the runway throughput.

While an arrival sequence is solely constrained by the wake-vortex separation, a departure sequence is additionally constrained by the aircraft speed separation and the en-route separation. Indeed, departure flights have to follow a fixed departure route called the Standard Instrument Departure (SID) route. To avoid airspace congestion and to ensure a safe in-flight separation between departures, the departure frequency along each SID and group of SIDs has to be restricted. This is accomplished by maintaining a minimal separation between aircraft, based on their SID routes and speed categories. The en-route separation matrix for the four SID routes at Manchester Airport is provided in Table 2. As for the minimal speed separation, the gap is increased by one minute for each successive group when a faster aircraft follows a slower aircraft (e.g., 1 minute if Group 4 follows Group 3; 2 minutes if Group 3 follows Group 1, etc). When calculating the correct departure separation between two adjacent take-offs i and j, the highest of the three separation values is used.

At Manchester Airport, the minimal separation between departures complies with the triangle inequality. This implies that safe separation between take-offs is guar-

[2] Separation rules for departures at the Manchester Airport are available at http://ivao.co.uk/atc/egcc/out

Table 1
Minimal wake-vortex separation in seconds between two adjacent arrival aircraft i and j Bianco et al. (1997), and minimal departure wake-vortex separation between two adjacent departure aircraft i and j in accordance with the current regulations at the Manchester Airport.

Weight of the later arrival aircraft j					
		1	2	3	4
Weight of the earlier arrival aircraft i	1	96	200	181	228
	2	72	80	70	110
	3	72	100	70	130
	4	72	80	70	90
Weight of the later departure aircraft j					
		1	2	3	4
Weight of the earlier departure aircraft i	1	60	120	120	120
	2	120	60	120	120
	3	120	120	60	120
	4	120	120	120	60

Table 2
Minimal en-route separation matrices in accordance with the current regulations at the Manchester Airport (times in seconds)

SID of the later departure aircraft j					
		WEST	EAST	SAMBA	LISTO
SID of the earlier departure aircraft j	WEST	120	60	120	60
	EAST	60	120	60	60
	SAMBA	120	60	120	60
	LISTO	60	60	60	120

anteed by only taking into account separations between adjacent flights. However, this is not the case at all airports, e.g., at London Heathrow Airport (Atkin et al., 2007). Furthermore, the departure separation function is asymmetric so rearranging the flight orders in the departure sequence may reduce the total departure delay.

2.2.2 Landing/take-off time constraints

In most models for arrival scheduling (Beasley et al., 2000; Ernst et al., 1999), an aircraft is not allowed to land ahead of its *planned landing time PLT*. Aside from security risks, landing earlier than planned may require more fuel burn if the aircraft has to accelerate beyond its optimal cruise speed, thus introducing extra costs. The *actual landing time* $ALT(f_i)$ of each aircraft f_i in the arrival sequence is thus constrained with $ALT(f_i) \geq PLT(f_i)$, ensuring that a flight's earliest ALT is its PLT.

On the other hand, in departure scheduling, there is no equivalent benefit to delaying an aircraft beyond its earliest possible take-off time. Indeed, at busy times,

aircraft usually take off in the earliest slot that can accommodate them, since it is essential to maximize the runway throughput. Delayed departures can, however, be used to avoid en route congestion. To this end, a departure flight is generally assigned a 15 minute departure window, denoted by $[ETO(f_i), LTO(f_i)]$, during which the aircraft should take-off. More precisely, $PTOT(f_i) - 5 = ETO(f_i)$ and $PTOT(f_i) + 10 = LTO(f_i)$, where $PTOT(f_i)$ is the planned take-off time of f_i. This time window is imposed by a CTOT or Calculated Take-Off Time. The aircraft cannot take-off before its start (i.e. $ETO(f_i)$ is a hard constraint). In the case that an aircraft is unable to take-off within a given departure window, certain extensions are allowed but should be avoided whenever possible (i.e. $LTO(f_i)$ is a severely penalized soft constraint). Integration of runway sequencing and ground movement is crucial to ensure that an aircraft is only allocated a slot if it is likely to be able to complete taxiing in time to meet it.

2.3 Ground movement

Airport ground movement is a combined routing and scheduling problem (Atkin et al., 2010). Aircraft must be guided in a time-efficient manner along the taxiways, meeting assigned times at the runway and respecting safety constraints on the proximity of other aircraft. If the airport only has a few aircraft moving at once it would be possible to assign routes using a shortest path algorithm such as Dijkstra's algorithm or A*. Where an airport is more busy, the interactions between moving aircraft and changing obstacles such as runway crossings requires a more sophisticated approach. In this work, we seek to find a sequence for which all aircraft can be routed without conflicts in the shortest length of time.

The routing algorithm we adopt is a variant of the Quickest Path Problem with Time Windows (QPPTW) algorithm (Ravizza et al., 2013a). This resembles Dijkstra's shortest path algorithm, and routes the aircraft sequentially. For convenience, a summary of the algorithm is given in Appendix B.

We have made a number of small changes to the QPPTW algorithm presented in (Ravizza et al., 2013a). Firstly, we use an undirected graph, to reflect the operations at Manchester Airport. Secondly, an edge can have different weights depending on predecessor edges and aircraft type. Thirdly, runway crossings are included by explicitly reserving the appropriate edges during take-offs and landings. The reservations are 60 seconds, starting with the landing time $ALT(f_i)$ for arrivals, and ending with the take-off time $ATT(f_i)$ for departures. Finally, in (Ravizza et al., 2013a), it was shown that an overall reduction in taxi time could be obtained by using a simple swap-heuristic. If an aircraft is delayed over the shortest path possible, the delay-causing aircraft is found, and the two aircraft are allocated routes in reverse order. We have replaced this with a search heuristic, described in Section 3.4, that can re-order and re-route multiple aircraft, with the aim of resolving conflicts in-

volving more than two aircraft, which can easily occur in high-traffic conditions.

2.3.1 Feasibility of the sequencing solution in terms of routing

For a given aircraft f_i, the unimpeded taxi time $\mathscr{T}_{unimped_i}$ is the time for the route that would be allocated to the aircraft in the absence of any other aircraft.

The taxiing delay $\mathscr{D}_i$ for an aircraft f_i is the delay over the unimpeded taxi time for its allocated route, i.e. $\mathscr{T}_i - \mathscr{T}_{unimped_i}$, where $\mathscr{T}_i$ is the total taxi time of f_i. Longer taxiing delay implies greater environmental and economic costs due to more fuel burn. In case of departures, it may further cause a flight to miss its allocated take-off slot. Indeed, aircraft are not ready to push-back before boarding is complete and thus have a limited time to reach the runway before their allocated take-off slot.

We classify f_i as infeasible if:

- $\mathscr{D}_i > 5$ minutes for departures
- $\mathscr{D}_i > 10$ minutes for arrivals

A sequencing slot allocated to f_i is determined to be infeasible if the route R_i assigned to f_i is classed infeasible according to the rules above, or if no R_i can be found. This latter situation will occur if other traffic movements mean that no path is available starting / ending on the runway at the allocated time. We assign a higher feasibility threshold $\mathscr{D}$ for arrivals since it is more costly to delay aircraft landing (i.e., to have aircraft flying in circles while waiting for authorisation to land) than to tolerate longer taxiing time from runway to gate. Therefore, a higher feasibility threshold provides a higher tolerance for longer taxiing times and thus more routing flexibility.

The above defined constraint links the runway sequencing and routing problems by enabling the proposed approach to focus optimization on sequencing slots that result long taxi times in the corresponding routing solution. As it will be observed in Section 4, this strategy shows dramatic improvement of taxiway routing for minimal sacrifice of the runway sequencing time.

3 Integrating runway sequencing and ground movement problems

We present a Receding Horizon Control based Iterated Local Search (RHC-ILS) technique for tackling the runway sequencing problem (including both arrival and departure sequencing) and the ground movement problem in a dynamic Air Traffic Control (ATC) environment. ILS is a general stochastic approach that has shown to be effective on a wide range of NP-hard problems. Its basic idea is to iterate between intensification phase, to exploit in-depth the neighborhood of the current

solution, and diversification phase to explore new search space regions. Given the segregated two-runway mode, the proposed algorithm considers arrival and departure sequencing as two separate yet interrelated processes, where the arrival S_A and the departure S_D sequences are being optimized in a token-ring way. More precisely, the optimization of the arrival sequence is followed by the optimization of the departure sequence, always starting from the best solution found in the previous optimization phase, until a stopping condition is met. Let S_A^{best}, S_D^{best} and R^{best} be respectively the best found arrival sequence, the best found departure sequence and the routing solution for sequences S_A^{best} and S_D^{best}. Each time a new neighboring sequence S_A or S_D is obtained, the algorithm updates S_A^{best} or S_D^{best}, in case an improved solution is found in terms of the the following three criteria: (i) the total arrival and departure sequencing delay; (ii) number of feasible sequencing slots that can be met by the taxiing aircraft in R^{best} (see Section 2.3.1); and (iii) the total taxiing delay (see Section 2.3.1). To verify the latter two criteria, we apply a heuristic that iteratively performs unrouting and rerouting of a flight selection from the current sequences S_A and S_D, in search for an improved routing solution that minimizes the total taxiing delay.

Given the dynamic nature of the ATC setting, conventional local search approaches can hardly keep up with the need of real-time properties in practice. Predicted arrival and departure flows are subject to constant changes. For instance, some flights may be canceled or delayed due to technical problems or weather conditions, while some flights may ask for unanticipated emergency landing. Therefore, optimization of all the aircraft at once cannot necessarily ensure a valid solution. To cope with this limitation of a conventional local search technique, the proposed ILS strategy is based on the RHC framework that has been extensively used in the literature for the runway sequencing problems. The basic idea behind the RHC optimization strategy is to optimize the problem for the next N intervals in the near future, based on currently available information. Since not all flights are taken into account in the optimization process at a given time interval, RHC approach reduces the influence of inaccurate information and ensures that computational effort is not waisted on future flight plans which are likely to be revisited anyway.

The following sections provide a detailed description of the proposed approach for the integrated runway sequencing and ground movement problem. The corresponding pseudocodes are provided in Appendix A.

3.1 Receding horizon optimization scheme

The idea behind the Receding Horizon Control (RHC) technique is to approximate a single large planning problem as a sequence of smaller problems. More precisely, given the set of arrival and departure flights F sorted in an increasing order according to their PLTs/PTOTs (earliest flights first), at each planning instant

t_i, $i = 0, 1, 2, ..., N$ (also called horizon), the flights from F are grouped into one of the three categories: (i) inactive flights (aircraft that have already landed or taken off); (ii) active flights (flights that are expected to land or take off within the next p minutes, i.e., flights that are in the current horizon); (iii) forthcoming flights (flights that are not in the current horizon but are expected to land or take off in the future). In each horizon, the planning of both aircraft routing and runway sequencing is only performed for the set of active flights that will arrive or depart in the near future.

A simple diagram that shows the operation procedure is illustrated in Fig. 2. The general RHC framework of the proposed algorithm is presented in Alg. 1 (see Appendix A). Let S'_A and S'_D be the sequences of inactive arrival and departure flights respectively, and let S_A and S_D be the sequences of active arrivals and departures respectively. The algorithm repeats the following steps for a given number of horizons, until all aircraft are marked as inactive. At each horizon, S_A and S_D are first updated with a set of landing and take-off flights that enter the horizon. These newly considered flights are sequenced in a first-come first-served (FCFS) manner, and then routed from gate to runway or from runway to gate based on the calculated landing/take-off times. The resulting sequencing solution may be infeasible in terms of the corresponding routing solution, i.e., some departures may be unable to reach the runway on time for their allocated take-off slots, while some arrivals may take too long to taxi from runway to their gate. An iterated local search (ILS) procedure is then called to optimize S_A and S_D in a token-ring way, always starting from the best sequence found in the previous ILS run, the objective being to improve the solution in terms of the sequencing delay, taxiing delay and routing feasibility.

Finally, any aircraft, with a *feasible sequencing slot*, that has landed or taken off in the current horizon is recorded in S'_A or S'_D and removed from the sequence of active flights (see Section 2.3.1 for definition of infeasible sequencing slot). However, if a flight that is supposed to be marked as inactive at the end of t_i has an infeasible sequencing slot, it is delayed by at least δ minutes beyond its predicted landing or take-off time (δ is a fixed parameter) followed by further local optimization. The value for δ is determined so as to achieve a good trade-off between take-off/landing delay and the computing time. Indeed, the computing time requirements per horizon greatly vary depending on the amount of traffic at a given horizon, and hence depend on the difficulty of finding a feasible routing solution (with reduced taxiing times) for the runway sequences at hand. Intuitively, as take-offs/landings are postponed for longer periods (with the increase of δ), the chances of achieving feasibility in the following horizon increase.

In our implementation of the proposed approach, for the sake of simplicity of the evaluation procedure, the number of active flights in the current horizon (i.e., the horizon size) is a fixed parameter W. After the optimization of each planning horizon, a flight with the earliest estimated landing or take-off time from

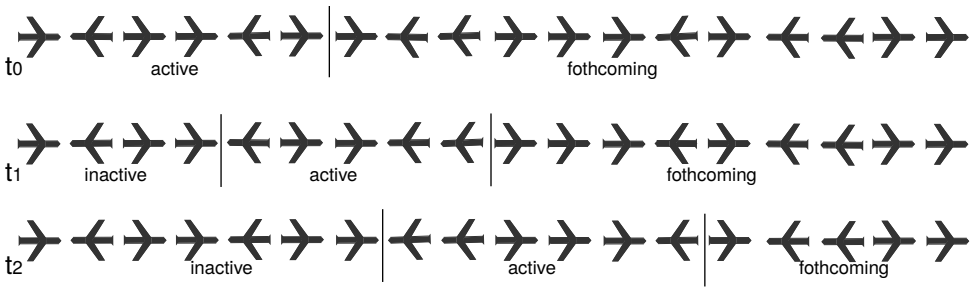


Fig. 2. Flight entering and leaving planning horizon

$S_A \cup S_D$ is marked as inactive in case its sequencing slot is feasible (ties are broken at random). If $|S_A \cup S_D| = W - 1$, a flight with an earliest PLT or PTOT from $F \setminus \{S_A \cup S_D \cup S'_A \cup S'_D\}$ is marked as active in the following horizon (ties are broken arbitrarily). We use two stopping criteria for the token-ring local optimization (*while* loop at line 16 of Alg. 1 in Appendix A): (i) the maximal number of iterations *max_iter*, and (ii) the maximal run-time *max_time*. Indeed, the computing effort of the proposed optimization procedure per iteration highly depends on the feasibility of the current arrival and departure sequences in terms of the corresponding routing solution.

The main algorithmic components are detailed in the following sections.

3.2 Optimization objective and evaluation functions of the integrated runway sequencing and ground movement problems

The objectives of the arrival and departure sequencing can take various forms and will depend upon the airport. In the case of arrival sequencing, the most popular metric adopted in the literature is the minimization of the total delay of aircraft, computed as

$$C_A(S_A) = \sum_{j=1}^{n} ALT(f_j) - PLT(f_j), \tag{1}$$

where ALT is the actual landing time of an aircraft. Considering the runway separation constraint $sep(f_{i-1}, f_i)$ between two consecutive arrivals and the earliest landing time constraint (see Sections 2.2.1 and 2.2.2), ALT is computed as

$$ALT(f_i) = \begin{Bmatrix} PLT(f_i), & \text{if } i = 1 \\ max\{PLT(f_i), ALT(f_{i-1}) + sep(f_{i-i}, f_i)\}, & \text{if } i > 1 \end{Bmatrix}, \tag{2}$$

where f_{i-1} precedes f_i in the arrival sequence. The objective function from Eq. 1 is also used as the evaluation function.

For departure sequencing, the objective is to minimize the total difference between the actual take-off time ATT and the planned take-off time PTOT. However, as it is not always possible for an aircraft to meet its departure window at congested times, the evaluation function presented in Eq. 3 penalizes each miss of a departure slot with a high penalty P_h, while aircraft whose ATT is planned γ minutes before the end of their departure window are penalized with a low penalty P_l ($\gamma = 3$ in our experiments).

$$C_D(S_D) = \sum_{j=1}^{m} \begin{Bmatrix} P_h(ATT(f_j) - PTOT(f_j)), & \text{if } (ATT(f_j) - PTOT(f_j)) > 10 \\ P_l(ATT(f_j) - PTOT(f_j)), & \text{else if } (10-\gamma) < (ATT(f_j) - PTOT(f_j)) \leq 10 \\ ATT(f_j) - PTOT(f_j), & \text{otherwise} \end{Bmatrix}. \tag{3}$$

Considering the runway separation constraint between departures as well as the earliest take-off time constraint, ATT is determined as

$$ATT(f_i) = \begin{Bmatrix} ETO(f_i), & \text{if } i = 1 \\ max\{ETO(f_i), ATT(f_{i-1}) + sep(f_{i-1}, f_i)\}, & \text{if } i > 1 \end{Bmatrix} \tag{4}$$

Let $F_A(t)$ be the set of active arrivals and let $F_D(t)$ be the set of active departures at the current planning instant t. The objective of the integrated runway sequencing and ground movement problem is to find an arrival sequence S_A of $F_A(t)$ and a departure sequence S_D of $F_D(t)$ that minimizes:

$$C(S_A, S_D) = C_A(S_A) + C_D(S_D), \tag{5}$$

while ensuring that the number of infeasible sequencing slots and the total taxiing delay for the corresponding routing solution R, is kept to a minimum.

Given the computational complexity of the IRH algorithm and the QPPTW routing procedure detailed in the following sections, the evaluation of each neighboring flight sequence (see Section 3.3) is solely based on the evaluation function $eval_1(S)$ that corresponds to $C_A(S)$ when the arrival sequence $S = S_A$ is being optimized and to $C_D(S)$ otherwise:

$$eval_1(S) = \begin{Bmatrix} C_A(S), & \text{if } S \text{ is an arrival sequence} \\ C_D(S), & \text{otherwise} \end{Bmatrix}. \tag{6}$$

As explained in the following section, we use two additional evaluation functions $eval_2(R)$ and $eval_3(R)$ to decide whether to update the best found sequencing and routing solutions, where R is the routing solution given S.

The second evaluation function $eval_2(R)$ determines the number of aircraft with an infeasible sequencing slot, while assigning higher penalty to those flights that were

delayed by at least δ minutes in the previous horizons because of infeasibility (i.e., long taxi times):

$$eval_2(R) = \sum_{f \in S_A \cup S_D} \left\{ \begin{array}{cc} (d(f)+1)^2, & \text{if } f \text{ has an infeasible sequencing slot} \\ 0, & \text{otherwise} \end{array} \right\}, \quad (7)$$

where $d(f)$ is the number of times the flight f was delayed by at least δ minutes until the current horizon. The function assigns a quadratic cost in terms of the number of times a flight has been delayed, thus favoring slot feasibility of arrivals/departures that have been deemed infeasible in the previous horizons.

Given the current routing solution R, the third evaluation function $eval_3(R)$ returns the total taxiing delay of all flights in $S_A \cup S_D$:

$$eval_3(R) = \sum_{f_i \in S_A \cup S_D} \mathscr{D}_i, \quad (8)$$

where $\mathscr{D}_i$ is the taxiing delay of f_i (see Section 2.3.1).

3.3 Neighborhood for runway sequencing, its exploitation and exploration

Given a sequence S of active arrival or departure flights, the ILS heuristic is based on the insert move $m(f', f'')$ which consists of removing an aircraft f' from its current position in S and inserting it after another flight f'' in S. This gives a total of $n-1$ possible positions (i.e., solutions) where n is the sequence length. An example of the insert move is provided in Fig. 3.

The framework of the proposed ILS procedure is presented in Alg. 2 (See Appendix A). The algorithm iterates between a simple best-improvement local search to exploit the above defined neighborhood, and a tabu-based perturbation to direct the search away from local optima. Each iteration of the best-improvement local search phase consists of selecting and performing the best insert move that maximizes the runway throughput, i.e., minimizes the evaluation function $eval_1(S)$ defined in Eq. 6 (see Section 3.2). This intensification phase ends as soon as a local optimum is attained.

The diversification phase is a standard tabu search procedure (Glover, 1989, 1990) where each iteration consists of performing the best insert move, under constraint that the given move is not prohibited by the tabu list. Move prohibition is determined in the following way. Let f be the flight prior to flight f' in the current sequence S. If f' is resequenced after another flight f'', the reverse move $m(f, f')$ is

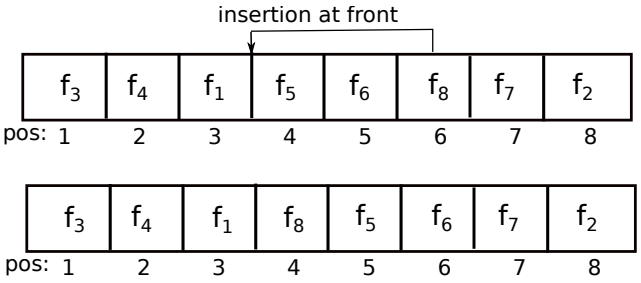


Fig. 3. Insertion of flight f_8 after f_1.

prohibited for at least $tabu(f, f')$ iterations. The value of $tabu(f, f')$ is determined with the following relation:

$$tabu(f, f') = iter + \sigma, \tag{9}$$

where σ is the tabu tenure parameter, and $iter$ is the iteration number when flight f' was resequenced after a different flight. The tabu status of a move is neglected only if the move leads to a new solution better than the best solution found so far in terms of the evaluation function $eval_1(S)$.

Note that the best-improvement local search of the given ILS algorithm is equivalent to the tabu-based perturbation mechanism when move restrictions are lifted. Therefore, both phases call the *TransformSequence* procedure, given in Alg. 3 (see Appendix A), for move selection and application. The procedure updates the corresponding best found sequence S_{best} and the routing solution R_{best} if (i) the transformed sequence S' is at least as good as S_{best} in terms of the total sequencing delay (see evaluation function $eval_1$ in Eq. 6, Section 3.2); and (ii) the resulting routing solution R' is at least as good as R_{best} in terms of infeasible sequencing slots (see evaluation function $eval_2$ in Eq. 7, Section 3.2). We further use a third criterion, which is the minimization of total taxiing delay (see evaluation function $eval_3$ in Eq. 8, Section 3.2), in case that $eval_1(S') = eval_1(S_{best})$ and $eval_2(R') = eval_2(R_{best})$. The latter two conditions are verified by means of the *IRH* procedure described in Section 3.4.

3.4 Iterative routing heuristic

Let S' be an active arrival (departure) sequence obtained after performing an insert move to sequence S in the *TransformSequence* procedure, and let R_{best} be the routing solution for the best found arrival and departure sequences. The purpose of the proposed iterative routing heuristic (IRH) is to determine the corresponding routing solution R' for S' that increases the number of feasible sequencing slots (minimizes $eval_2(R')$), while reducing the total taxiing delay (minimizing $eval_3(R')$). The quality of R', both in terms of $eval_2(R')$ and $eval_3(R')$, depends upon the routing order

of aircraft in R'. Therefore, the basic idea behind IRH is to iteratively improve the quality of R' by unrouting a subset UR of *active* aircraft from R' and routing them again in a partially randomized order. The routing procedure is an adaptation of the QPPTW algorithm (Ravizza et al., 2013a) detailed in Appendix B.2.

The IRH procedure is shown in Alg. 4 (see Appendix A). IRH does not determine R' for S' from scratch. In fact, the initial solution for R' constitutes a modified solution R_{best} obtained by unrouting and rerouting (in no specific order) those flights from R_{best} whose actual landing ALT (take-off ATT) times differ in S' and in the corresponding best found sequence S_{best}.

Let R_{tmp} be the temporary (working) routing solution initialized to R', and let IS be the set of active aircraft with infeasible sequencing slots given R_{tmp}, each iteration of IRH performs the following steps. First, IRH randomly selects an aircraft r from IS. It then determines a set UR of active aircraft for rerouting in R_{tmp}, consisting of those aircraft whose actual (landing or take-off) time (dented as AT) is γ minutes away from $AT(r)$. In the following step, IRH establishes the order of rerouting in R_{tmp} the flights from UR. The procedure gives routing priority to those flights with infeasible sequencing slots, i.e., to aircraft f whose taxiing time from the gate to runway (or from the runway to gate) exceeds a maximum time limit. Among this selection of active aircraft with infeasible sequencing slots, it further prioritizes routing arrivals before departures, since an aircraft that is landing has the right of way over an airplane taking off. The rest of aircraft in UR are assigned a higher and random rerouting order. Finally, IRH updates the best found routing solution R' with R_{tmp} if R_{tmp} constitutes an improvement over R' in terms of taxiing delay minimization (evaluation function $eval_3$, see Eq. 8), and resets R_{tmp} to R' if otherwise. The IRH procedure stops if the sequencing slots of all the active aircraft are deemed to be feasible, or if the number of IRH iterations without improvement to R' reaches a fixed threshold ϕ.

4 Experimental results

This section provides two experimental comparisons to evaluate the effectivness and robustness of the proposed algorithm for the combined runway sequencing and ground movement problems. The purpose of the first comparison is to evaluate the performance of the proposed RHC-ILS procedure for runway sequencing (see Section 3.3), with respect to the recent state-of-art algorithms for runway sequencing in segregated mode on a single runway airport. For this comparison, we only consider the arrival sequencing & scheduling problem as in the reference works (Hu and Chen, 2005; Hu and Paolo, 2008; Zhan et al., 2010). The aim of the second comparison is to evaluate and analyze the quality of runway sequencing and ground movement solutions obtained with our approach (denoted as RS-GM-sim), with respect to sequentially optimized flight sequencing and ground movement solutions

Table 3
Setting of parameters. *rnd* is a function that returns a random value from a given range.

Param.	Description	RS-GM-sim	RS-GM-seq	AS
W	Number of aircraft in the current horizon	40	40	20
P_h	High penalty coeff. for missing a departure slot	20	20	–
P_l	Low penalty coeff. for missing a departure slot	5	5	–
max_iter	Max. number of token-ring optimization iter. per horizon	5	20	–
max_time	Max. time limit for token-ring optimization per horizon	–	–	–
max_ILS_iter	Max. number of ILS iterations	4	5	800
σ	Tabu tenure for tabu-based pertubation	rnd(4, 14)	rnd(4, 14)	rnd(4, 14)
δ	Delay coefficient	5 min	–	–
ϕ	Max. number of consecutive IRH iter. without improvement	20	30	–
γ	Unrouting coefficient of IRH	10 min	10 min	–
α	Taxiing delay feasibility threshold for arrivals	10 min	10 min	–
β	Taxiing delay feasibility threshold for departures	5 min	5 min	–

obtained with a slight modification of our approach (denoted as RS-GM-seq).

4.1 Experimental protocol

Our code is programmed in Java (version 1.8). Experiments were run on a Dual Intel Xeon X5650 with 2.66GHz and 8GB RAM. The setting of parameters used in our experiments is given in Table 3. Column 'RS-GM-sim' shows the parameter settings of the proposed approach detailed in Section 3. Column 'RS-GM-seq' indicates the parameter settings of a slight variation of our algorithm which sequentially optimizes runway sequencing and ground movement problems, i.e., only the sequencing delay is considered during the optimization of the sequencing solutions. More precisely, at each planning horizon, the execution of IRH to obtain a routing solution starts after the optimization of the runway sequences with the proposed ILS procedure, and no action is taken in case of infeasible sequencing slots. Finally, column 'AS' provides the setting of parameters of our RHC-ILS approach, applied solely to the arrival sequencing and scheduling problem.

4.2 Comparison of procedures for arrival sequencing and scheduling

To evaluate the performance of our method detailed in Section 3, we first provide computational results of its runway sequence optimization procedure RHC-ILS. Since the proposed approach is designed for segregated sequencing airport operating mode, we provide comparisons with the following heuristics that are considered to be among the most effective for this variant of runway sequencing:

- RHC-GA (Hu and Chen, 2005): A genetic algorithm (GA), within the receding horizon control (RHC) framework, for solving the dynamic arrival sequencing;
- RHC-BRGA (Hu and Paolo, 2008): A genetic algorithm for arrival sequencing, based on a binary representation of arriving queues, combined with RHC;

Table 4
Comparison between RHC-ILS, RHC-GA Hu and Chen (2005), RHC-ACS Zhan et al. (2010) and the first-come first-served method on an instance with 30 aircraft taken from Hu and Chen (2005)

Data			FCFS				RHC-GA				RHC-ACS				RHC-ILS			
S_A	Cat.	PLT	S_A	Cat.	ALT	Delay	S_A	Cat.	ALT	Delay	S_A	Cat.	ALT	Delay	S_A	Cat.	ALT	Delay
1	1	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
2	1	79	2	1	96	17	2	1	96	17	2	1	96	17	2	1	96	17
3	1	144	3	1	192	48	3	1	192	48	3	1	192	48	3	1	192	48
4	2	204	4	2	392	188	5	1	288	24	5	1	288	24	5	1	288	24
5	1	264	5	1	464	200	6	1	384	64	6	1	384	64	6	1	384	64
6	1	320	6	1	560	240	4	2	584	380	4	2	584	380	4	2	584	380
7	2	528	7	2	760	232	7	2	664	136	7	2	664	136	7	2	664	136
8	1	635	8	1	832	197	9	2	744	14	9	2	744	14	9	2	744	14
9	2	730	9	2	1032	302	10	2	824	58	10	2	824	58	10	2	824	58
10	2	766	10	2	1112	346	8	1	896	261	8	1	896	261	8	1	896	261
11	1	790	11	1	1184	394	11	1	992	202	11	1	992	202	11	1	992	202
12	1	920	12	1	1280	360	12	1	1088	168	12	1	1088	168	12	1	1088	168
13	3	1046	13	3	1461	415	15	2	1288	152	13	3	1269	223	13	3	1269	223
14	4	1106	14	4	1591	485	16	2	1368	202	17	2	1369	136	16	2	1369	203
15	2	1136	15	2	1671	535	17	2	1448	215	15	2	1449	313	15	2	1449	313
16	2	1166	16	2	1751	585	13	3	1518	472	16	2	1529	363	17	2	1529	296
17	2	1233	17	2	1831	598	14	4	1648	542	14	4	1639	533	14	4	1639	533
18	1	1642	18	1	1903	261	18	1	1720	78	18	1	1711	69	18	1	1711	69
19	1	1715	19	1	1999	284	19	1	1816	101	19	1	1807	92	19	1	1807	92
20	3	1770	20	3	2180	410	20	3	1997	227	20	3	1988	218	20	3	1988	218
21	1	2074	21	1	2252	178	21	1	2074	0	21	1	2074	0	21	1	2074	0
22	1	2168	22	1	2348	180	22	1	2170	2	22	1	2170	2	22	1	2170	2
23	4	2259	23	4	2576	317	23	4	2398	139	23	4	2398	139	23	4	2398	139
24	2	2427	24	2	2656	229	24	2	2478	51	24	2	2478	51	24	2	2478	51
25	1	2481	25	1	2728	247	25	1	2550	69	25	1	2550	69	25	1	2550	69
26	2	2679	26	2	2928	249	26	2	2750	71	26	2	2750	71	26	2	2750	71
27	3	2883	27	3	2998	115	27	3	2883	0	27	3	2883	0	27	3	2883	0
28	2	2982	28	2	3098	116	28	2	2983	1	28	2	2983	1	28	2	2983	1
29	1	3046	29	1	3170	124	29	1	3055	9	29	1	3055	9	29	1	3055	9
30	1	3091	30	1	3266	175	30	1	3151	60	30	1	3151	60	30	1	3151	60
Tot.						8027				3763				3721				3721

Table 5
Comparison between our RHC-ILS sequencing procedure and RHC-ACS Zhan et al. (2010) on the instance with 30 aircraft taken from Hu and Chen (2005)

Algorithm	Worst	Best	Mean	Std. Dev	(Mean-Best)/Best	CPU Time (ms)	Best Ratio
RHC-ILS	**3721**	**3721**	**3721**	**0.0%**	**0.00**	**24.02**	**100%**
RHC-ACS	4075	**3721**	3730.3	53.6	0.25%	222.18	97%

- RHC-ACS (Zhan et al., 2010): An ant colony system (ACS) algorithm for arrival sequencing based on RHC.

We further compare our results with those obtained with the simple first-come first-served (FCFS) method. It is perhaps worth mentioning a recent dynamic programming algorithm (Maere and Atkin, 2015) for optimal runway sequencing at an extremely low computational cost. Since this algorithm was only considered for single runway operations in mixed mode, it is not used in the comparison.

For this comparison, we use two test cases with 30 and 20 aircraft taken from (Hu and Chen, 2005) and (Hu and Paolo, 2008) respectively. Tables 4 and 6 present

Table 6
Comparison between RHC-ILS, BRGA Hu and Paolo (2008), RHC-ACS Zhan et al. (2010) and the first-come first-served method on an instance with 20 aircraft taken from Hu and Paolo (2008)

Data			FCFS				BRGA				RHC-ACS				RHC-ILS			
S_A	Cat.	PLT	S_A	Cat.	ALT	Delay	S_A	Cat.	ALT	Delay	S_A	Cat.	ALT	Delay	S_A	Cat.	ALT	Delay
1	1	1935	9	4	35	0	9	4	35	0	9	4	35	0	9	4	35	0
2	3	400	5	3	142	0	5	3	142	0	5	3	142	0	5	3	142	0
3	4	879	10	1	307	0	10	1	307	0	10	1	307	0	10	1	307	0
4	1	328	4	1	403	75	4	1	403	75	4	1	403	75	4	1	403	75
5	3	142	12	2	603	241	19	1	499	5	19	1	499	5	19	1	499	5
6	2	1980	2	3	673	273	17	1	595	30	17	1	595	30	17	1	595	30
7	2	915	19	1	745	251	12	2	795	433	18	3	776	111	2	3	776	376
8	2	1814	17	1	841	276	18	3	865	200	2	3	846	446	18	3	846	181
9	4	35	18	3	1022	357	2	3	935	535	7	2	946	31	12	2	946	584
10	1	307	3	4	1152	273	7	2	1035	120	12	2	1026	664	7	2	1026	111
11	3	1414	7	2	1232	317	3	4	1145	266	3	4	1136	257	15	4	1136	183
12	2	362	15	4	1342	389	15	4	1235	282	15	4	1226	273	3	4	1226	347
13	4	1279	14	1	1414	434	13	4	1325	46	13	4	1316	37	13	4	1316	37
14	1	980	13	4	1642	363	20	2	1408	0	20	2	1408	0	20	2	1408	0
15	4	953	20	2	1722	314	11	3	1478	64	11	3	1478	64	11	3	1478	64
16	3	1726	11	3	1792	378	14	1	1550	570	14	1	1550	570	14	1	1550	570
17	1	565	16	3	1862	136	16	3	1731	5	16	3	1731	5	16	3	1731	5
18	3	665	8	2	1962	148	8	2	1831	17	8	2	1831	17	8	2	1831	17
19	1	494	1	1	2034	99	6	2	1980	0	6	2	1980	0	6	2	1980	0
20	2	1408	6	2	2234	254	1	1	2052	117	1	1	2052	117	1	1	2052	117
Tot.						4578				2765				2702				2702

Table 7
Comparison between our RHC-ILS sequencing procedure and RHC-ACS Zhan et al. (2010) on the instance with 20 aircraft taken from Hu and Paolo (2008)

Algorithm	Worst	Best	Mean	Std. Dev	(Mean-Best)/Best	CPU Time (ms)	Best Ratio
RHC-ILS	**3165**	**2702**	2850.16	215.98	5.4%	182.5	**68%**
RHC-ACS	3266	**2702**	**2823.38**	**160.55**	4.49%	**160.62**	45%

the arrival sequences obtained after one run of the reference algorithms. The PLTs and weight categories are taken from the corresponding papers, while the last row in each table gives the total arrival delay. We observe that the simple FCFS yields poor performance in both cases. Our RHC-ILS ensures arrival sequences of the same quality as RHC-ACS for the two instances, while slightly outperforming RHC-GA and RHC-BRGA. More detailed comparisons between RHC-ILS and RHC-ACS are thus provided in Tables 5 and 7 for the two test cases. The reported results are based on 100 independent runs. The figures for RHC-ACS are taken from (Zhan et al., 2010). For the first test case, we observe that RHC-ILS is able to find the best known solution in every trial (i.e., Best Ratio = 100%) with very short computing time (CPU time = 24.02ms on average), while RHC-ACS finds the best known solution in 97% of the cases with an average computing time of 222.18ms. On the other hand, the results in Table 7 show that RHC-ACS is somewhat better than RHC-ILS on the second instance. Indeed, the average percentage deviation from the best-known solution is 4.49% (vs 5.4% for RHC-ILS), while the average computing times are comparable.

In all, the results show that our RHC-ILS is highly competitive with the current state-of-art sequencing algorithms, both in terms of solution quality and computing time.

4.3 Comparison between simultaneous and sequential runway sequencing & ground movement optimization

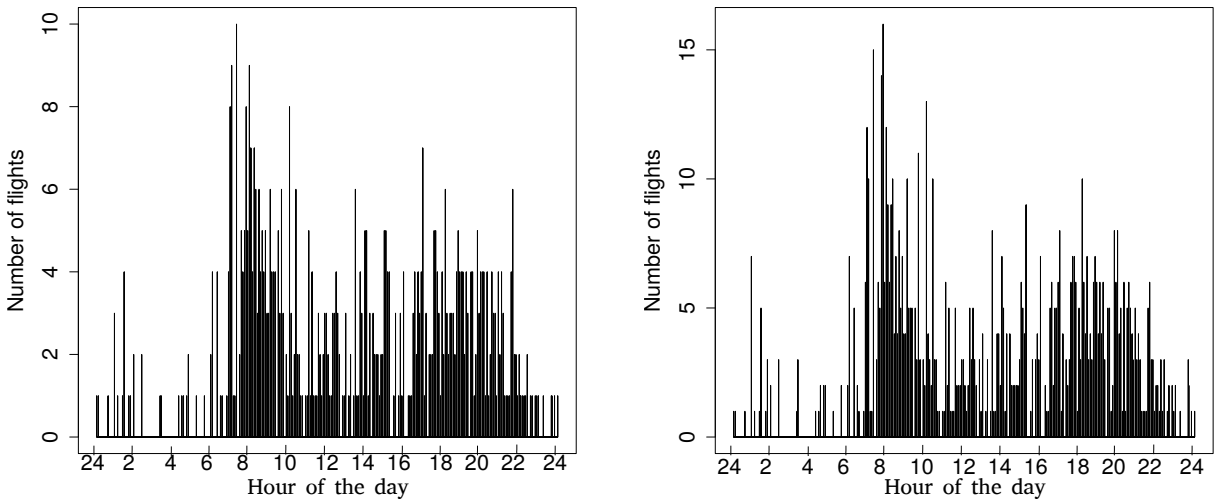
This section evaluates the benefit of coupled optimization of runway sequencing & ground movement with our RS-GM-sim approach with respect to RS-GM-seq, using the Manchester Airport as a test case. The results are based on 30 executions of both approaches, using the parameter settings given in Table 3. The benchmark set is detailed in the next section, followed by analyses of the comparative results.

4.3.1 Benchmark set

We perform experiments and comparisons on a set of 36 instances, based on data covering six days of operations (29 August to 3 September 2011) at Manchester Airport. Six instances *DayX*_1.0 ($1 \leq X \leq 6$) represent original data provided by the airport, while the rest were generated by increasing or decreasing the number of flights to a multiple of the original. For example, instances *DayX*_0.8 have 20% of the flights removed from the original data, while instances *DayX*_1.4 have 40% extra flights added. To increase/decrease the instance size, fights were chosen at random and either deleted, or copied by duplicating the runways used and flight times. Duplicated flight times have a two minute offset added in order not to take the runway at exactly the same time. Fig. 4 shows the number of flights per minute for instances Day3_1.0 and Day3_1.4. We observe from these two plots that the maximum number of flights occurring at the same minute is 10 and 16 for Day3_1.0 and Day3_1.4 respectively. The weight classes, speed classes and SIDs for all instances are generated at random. The instances are freely available under DOI `10.5281/zenodo.21027`.

The taxiway layout is taken from OpenStreetMap [3] . OSM data has some imperfections, but is surprisingly accurate and can easily be edited to resolve any detected issues. Since OSM did not have stand coordinate data of Manchester Airport, this was added using coordinates taken from NATS AIS [4] . The OSM data was processed into a usable format using the TaxiGen tool [5] (Brownlee et al., 2014). More recently, stand coordinate data has been added to OSM, allowing the redistributable layouts [6] to be made available.

[3] OpenStreetMap is available at: `www.openstreetmap.org`
[4] NATS AIS: `http://www.nats-uk.ead-it.com`
[5] TaxiGen tool is available at `https://github.com/gm-tools/gm-tools`
[6] `http://www.asap.cs.nott.ac.uk/external/atr/benchmarks/index.shtml\`



(a) Day3_1.0 (b) Day3_1.4

Fig. 4. Distribution of flights per minute

4.3.2 Test results

Table 8 shows the runway sequencing and routing results for RS-GM-seq and RS-GM-sim in terms of solution feasibility, sequencing delay and taxiing delay. We can make the following observations. In case of arrival delays, column 'Arr. delay' indicates that the average delay per flight is practically equal for both RS-GM-sim and RS-GM-seq sequences, except for Day1_1.2 where this delay is around 50 seconds longer for RS-GM-sim. Even though the optimal solutions for the coupled problems are not known given the computational hardness, we may assume, based on the comparison performed in the previous section, that the runway sequencing solutions obtained with RS-GM-seq are of high quality.

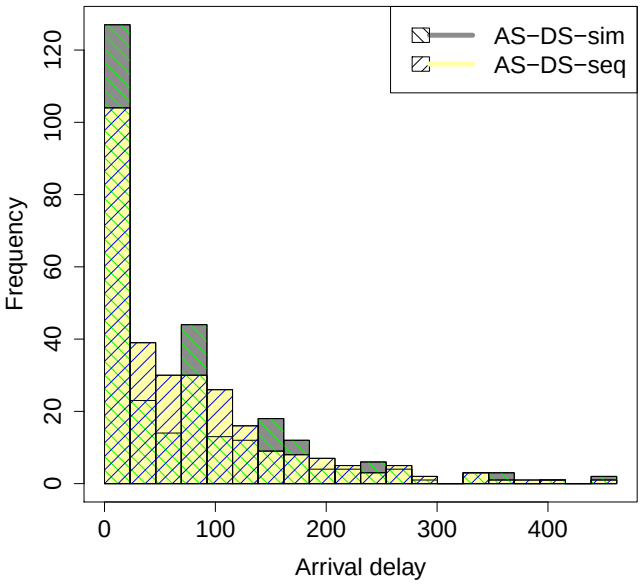
As for the departure delays, in cases of increased traffic (especially at peak hours, i.e., instances DayX_1.1 to DayX_1.4), the average delay per aircraft for an RS-GM-sim sequence may be from 0.5 minutes up to 4 minutes longer than for an RS-GM-seq sequence. In few cases (with a 20%-40% traffic increase), we further observe a significant standard deviation of delays for RS-GM-sim departure sequences with up to 225 seconds. Such results are as expected given the increased traffic at taxiways and the runway crossing. However, during regular days (instances DayX_1.0), or days with reduced traffic (instances DayX_0.8 and DayX_0.9), the overhead departure delay for an RS-GM-sim sequence is generally negligible.

To further support these observations, we provide overlapping histograms in Fig. 5 to compare average arrival delays (left subfigures) and departure delays (right

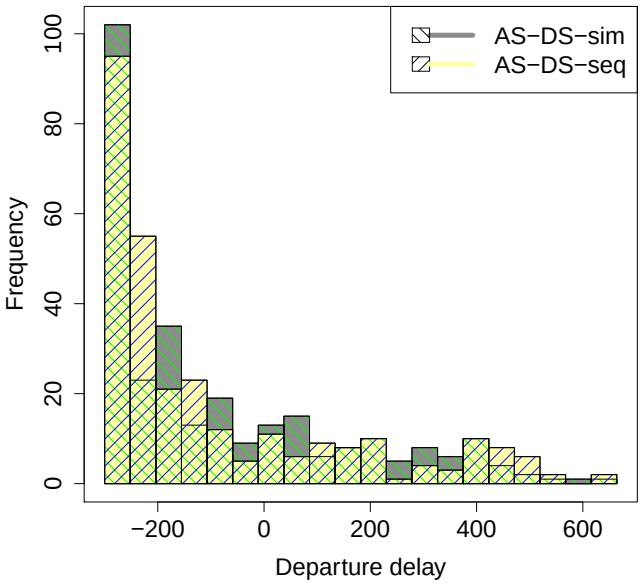
`#groundmovementDataSetsMAN`

Table 8: **Comparison of sequencing delays for solutions obtained with RS-GM-sim and RS-GM-seq. Columns 'Arr. delay' and 'Dep. delay' denote respectively the average arrival and departure sequencing delays per flight. Columns '#Infeas. arr.' and '#Infeas. dep.' indicate the average number of infeasible arrival and departure slots respectively, while column 'Taxi delay' provides the average taxiing delay per aircraft. Standard deviation values are given in parentheses. The delays are expressed in seconds.**

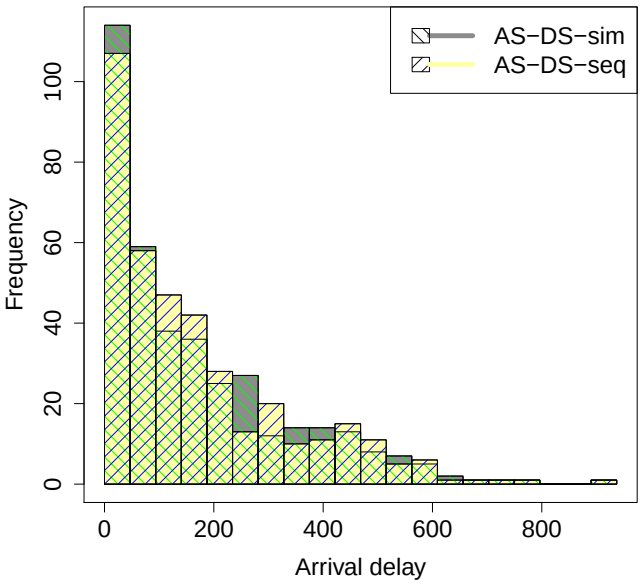
	RS-GM-seq					RS-GM-sim				
Instance	Arr. delay	Dep. delay	#Infeas. arr.	#Infeas. dep.	Taxi delay	Arr. delay	Dep. delay	#Infeas. arr.	#Infeas. dep.	Taxi delay
Day1_0.8	18.85(0.00)	-102.59(2.07)	0.20(0.65)	10.60(2.72)	77.80(12.37)	18.85(0.00)	-102.41(1.90)	0.00(0.00)	0.00(0.00)	13.82(0.70)
Day1_0.9	20.74(0.00)	-85.35(2.21)	0.17(0.45)	12.00(3.50)	83.84(11.52)	20.74(0.00)	-84.06(1.85)	0.00(0.00)	0.00(0.00)	23.28(1.39)
Day1_1.0	24.90(0.00)	-72.45(1.56)	0.03(0.18)	22.03(4.19)	86.93(7.59)	24.90(0.00)	-72.13(1.21)	0.00(0.00)	0.00(0.00)	19.29(1.35)
Day1_1.1	35.36(0.00)	-51.84(0.19)	4.30(6.36)	39.10(4.71)	130.09(34.65)	35.36(0.00)	-51.28(1.73)	0.00(0.00)	0.00(0.00)	30.54(2.05)
Day1_1.2	49.15(0.00)	-39.56(1.90)	13.73(20.14)	61.63(7.70)	249.77(108.26)	97.64(5.19)	203.97(128.41)	0.00(0.00)	0.00(0.00)	47.25(3.37)
Day1_1.3	63.49(0.00)	7.55(4.61)	10.43(10.51)	78.47(6.20)	239.11(48.53)	63.49(0.00)	17.21(16.38)	0.00(0.00)	0.00(0.00)	46.76(2.86)
Day2_0.8	26.00(0.00)	-78.66(0.71)	1.40(3.45)	14.67(3.45)	101.11(20.17)	26.00(0.00)	-75.91(8.44)	0.00(0.00)	0.00(0.00)	18.45(1.57)
Day2_0.9	36.60(0.00)	-50.80(1.60)	5.50(8.67)	25.60(3.67)	171.00(50.02)	36.60(0.00)	-54.23(15.07)	0.00(0.00)	0.00(0.00)	25.09(2.62)
Day2_1.0	51.21(0.00)	-28.57(2.48)	17.37(7.88)	31.40(6.74)	197.49(39.27)	51.21(0.00)	3.35(12.49)	0.00(0.00)	0.00(0.00)	31.80(2.79)
Day2_1.1	78.83(0.00)	-3.08(3.26)	22.03(9.36)	43.70(4.57)	205.60(53.94)	78.84(0.02)	28.17(30.93)	0.00(0.00)	0.00(0.00)	33.98(2.16)
Day2_1.2	96.27(0.00)	74.77(6.63)	39.40(13.04)	76.00(6.99)	444.40(99.30)	97.49(3.67)	130.03(104.32)	0.00(0.00)	0.00(0.00)	46.08(2.78)
Day3_0.8	24.19(0.00)	-103.27(1.48)	0.37(1.45)	9.83(2.08)	74.94(6.45)	24.19(0.00)	-103.71(1.46)	0.00(0.00)	0.00(0.00)	20.12(1.13)
Day3_0.9	31.77(0.00)	-74.20(0.85)	3.77(5.67)	28.20(3.71)	150.12(71.49)	31.77(0.00)	-74.66(1.90)	0.00(0.00)	0.00(0.00)	25.71(1.55)
Day3_1.0	35.49(0.00)	-47.66(1.97)	12.13(11.98)	45.17(6.20)	228.91(72.68)	35.49(0.00)	-44.28(8.83)	0.00(0.00)	0.00(0.00)	35.93(2.28)
Day3_1.1	46.81(0.00)	-26.11(5.60)	29.97(15.53)	61.20(6.65)	342.41(94.75)	46.20(0.20)	-0.43(17.57)	0.00(0.00)	0.00(0.00)	44.72(2.79)
Day3_1.2	70.49(0.00)	24.13(3.29)	26.73(21.10)	87.80(8.54)	400.11(132.46)	70.73(1.06)	99.18(91.68)	0.00(0.00)	0.00(0.00)	48.35(3.60)
Day3_1.3	83.09(0.00)	80.21(11.74)	45.97(22.85)	97.43(9.47)	472.96(128.77)	82.89(0.43)	94.43(93.88)	0.00(0.00)	0.00(0.00)	51.08(4.30)
Day3_1.4	100.52(0.00)	171.08(6.71)	154.33(53.63)	139.90(10.40)	1235.52(345.19)	113.49(23.01)	297.76(225.06)	0.00(0.00)	0.00(0.00)	51.91(3.83)
Day4_0.8	24.14(0.00)	-99.05(0.37)	0.23(0.50)	13.90(2.75)	81.23(5.31)	24.14(0.00)	-98.97(0.36)	0.00(0.00)	0.00(0.00)	19.05(1.41)
Day4_0.9	26.11(0.00)	-71.00(0.11)	0.23(0.50)	18.80(3.83)	99.74(12.39)	26.11(0.00)	-74.23(5.89)	0.00(0.00)	0.00(0.00)	22.68(1.71)
Day4_1.0	32.44(0.00)	-55.46(0.34)	3.83(8.92)	29.93(3.93)	142.17(33.64)	32.44(0.00)	-55.91(1.69)	0.00(0.00)	0.00(0.00)	30.32(1.58)
Day4_1.1	42.75(0.00)	-54.92(3.61)	7.17(9.49)	49.47(5.88)	201.87(47.32)	42.75(0.00)	-40.89(10.43)	0.00(0.00)	0.00(0.00)	36.24(2.19)
Day4_1.2	56.57(0.00)	-4.68(3.11)	58.13(37.24)	69.60(8.87)	430.10(164.40)	57.13(0.11)	-18.91(55.53)	0.00(0.00)	0.00(0.00)	44.35(3.27)
Day4_1.3	67.89(0.00)	7.46(2.11)	130.93(111.93)	108.83(13.44)	832.15(515.71)	67.40(0.50)	33.50(72.52)	0.00(0.00)	0.00(0.00)	45.27(2.63)
Day5_0.8	24.14(0.00)	-102.96(0.57)	1.30(3.53)	20.30(3.48)	102.66(20.52)	24.14(0.00)	-103.04(1.51)	0.00(0.00)	0.00(0.00)	20.56(2.17)
Day5_0.9	25.40(0.00)	-81.50(2.69)	3.83(6.87)	28.47(3.47)	125.29(33.09)	25.40(0.00)	-82.31(4.56)	0.00(0.00)	0.00(0.00)	28.15(1.52)
Day5_1.0	30.89(0.00)	-53.54(3.62)	3.03(3.87)	37.23(3.79)	150.64(41.82)	30.89(0.00)	-52.02(4.94)	0.00(0.00)	0.00(0.00)	34.39(2.05)
Day5_1.1	43.88(0.00)	-50.51(5.16)	17.30(12.56)	60.30(5.44)	232.73(52.23)	43.96(0.23)	-16.93(35.56)	0.00(0.00)	0.00(0.00)	41.19(2.33)
Day5_1.2	54.29(0.00)	3.54(5.25)	367.15(243.53)	40.97(38.89)	87.90(11.06)	53.86(0.32)	35.27(46.32)	0.00(0.00)	0.00(0.00)	46.34(3.18)
Day6_0.8	20.69(0.00)	-110.17(0.21)	0.10(0.40)	10.27(1.98)	72.82(7.61)	20.69(0.00)	-109.83(0.20)	0.00(0.00)	0.00(0.00)	14.27(0.85)
Day6_0.9	25.10(0.00)	-92.47(1.50)	1.00(4.31)	16.77(2.39)	95.65(23.08)	25.10(0.00)	-90.77(0.53)	0.00(0.00)	0.00(0.00)	14.44(1.80)
Day6_1.0	30.12(0.00)	-85.91(0.22)	0.20(0.65)	19.47(3.44)	97.59(7.87)	30.12(0.00)	-85.61(0.71)	0.00(0.00)	0.00(0.00)	24.16(1.89)
Day6_1.1	41.53(0.00)	-76.18(1.21)	4.53(6.33)	38.17(4.20)	148.03(18.92)	41.53(0.00)	-79.72(8.21)	0.00(0.00)	0.00(0.00)	32.11(2.41)
Day6_1.2	48.50(0.00)	-48.88(0.32)	3.10(4.59)	50.27(6.58)	179.04(31.53)	48.50(0.00)	-47.56(9.56)	0.00(0.00)	0.00(0.00)	34.28(1.95)
Day6_1.3	59.55(0.00)	-47.06(0.96)	35.57(36.34)	81.70(8.08)	441.98(212.72)	58.95(0.24)	-17.35(30.71)	0.00(0.00)	0.00(0.00)	40.82(3.03)
Day6_1.4	82.35(0.00)	-21.92(1.30)	102.70(58.87)	95.57(8.94)	780.70(427.59)	82.93(0.67)	34.48(36.65)	0.00(0.00)	0.00(0.00)	117.40(36.60)



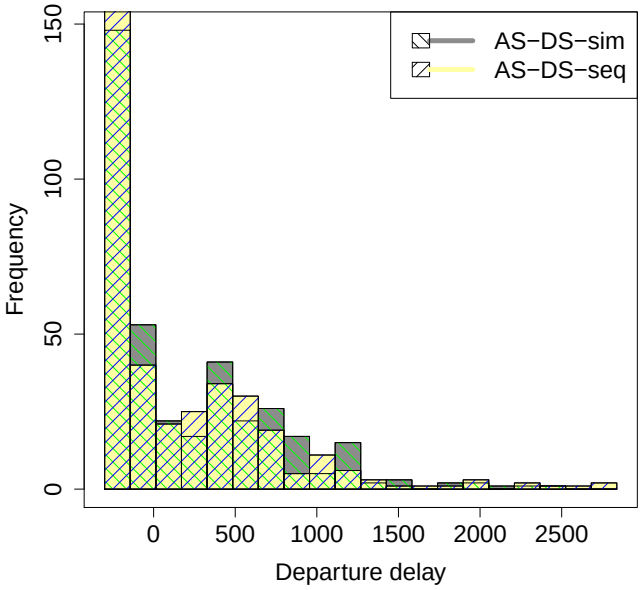
(a) Day3_1.0 - Arrival delays



(b) Day3_1.0 - Departure delays



(c) Day3_1.3 - Arrival delays



(d) Day3_1.3 - Departure delays

Fig. 5. Histograms comparing average arrival and departure sequencing delays per aircraft for solutions obtained with RS-GM-sim and RS-GM-seq. Delays are expressed in seconds.

subfigures) for RS-GM-sim and RS-GM-seq sequences, on instances Day3_1.0 and Day3_1.3. We also show the average number of missed CTOTs for RS-GM-sim and RS-GM-seq departures in Table 9. The plots confirm that the difference in the arrival delays, as well as the difference in the departure delays in case of Day3_1.0

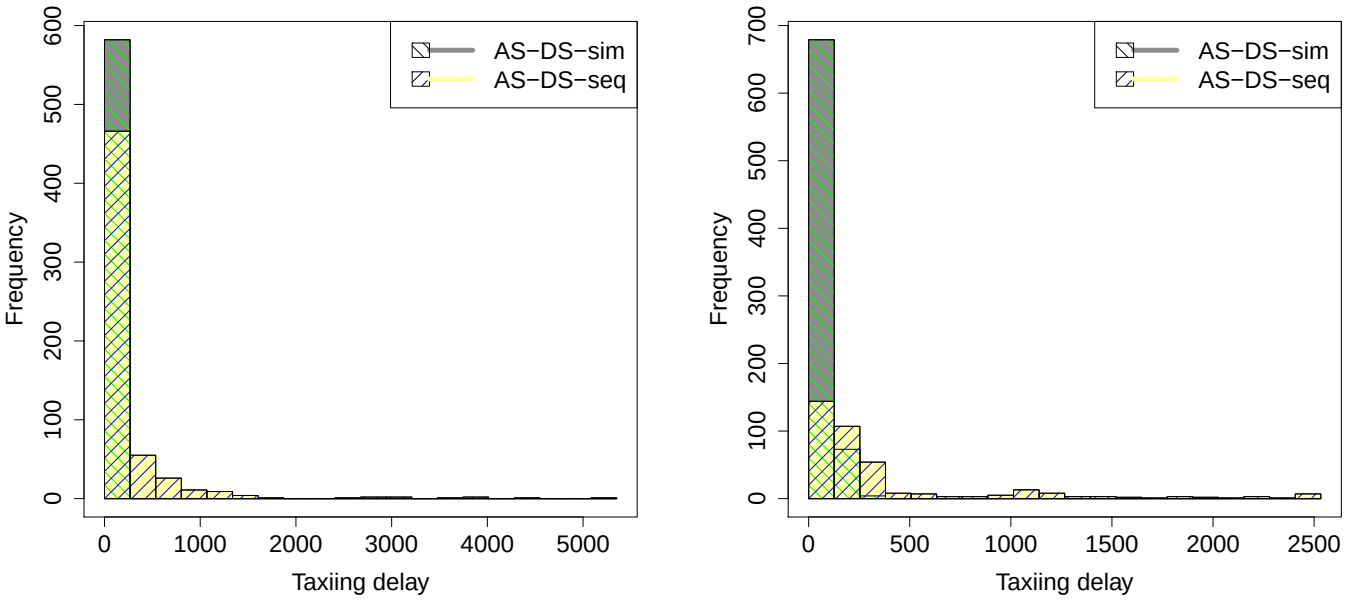
Table 9
Average number of missed CTOTs for RS-RM-sim and RS-GM-seq departure sequences. Column '#Depart.' indicates the total number of departures.

Instance	#Depart.	RS-GM-sim	RS-GM-seq	Instance	#Depart.	RS-RM-sim	RS-GM-seq
Day1_0.8	236	0.0	0.0	Day4_0.8	240	0.0	0.0
Day1_0.9	265	0.0	0.0	Day4_0.9	270	0.0	0.0
Day1_1.0	295	2.0	2.0	Day4_1.0	300	0.0	0.0
Day1_1.1	324	6.2	6.1	Day4_1.1	330	46.5	3.1
Day1_1.2	354	28.1	7.3	Day4_1.2	360	14.3	21.2
Day1_1.3	383	40.3	30.2	Day4_1.3	389	33.1	26.2
Day2_0.8	240	0.0	0.0	Day5_0.8	258	0.0	0.0
Day2_0.9	270	3.0	2.9	Day5_0.9	290	0.0	0.0
Day2_1.0	300	39.1	11.6	Day5_1.0	323	6.1	4.0
Day2_1.1	330	47.5	19.1	Day5_1.1	355	28.2	4.1
Day2_1.2	360	97.9	67.8	Day5_1.2	387	46.5	22.3
Day3_0.8	232	0.0	0.0	Day6_0.8	216	0.0	0.0
Day3_0.9	261	0.0	0.0	Day6_0.9	243	0.0	0.0
Day3_1.0	291	1.9	1.9	Day6_1.0	270	0.0	0.0
Day3_1.1	320	18.0	3.0	Day6_1.1	297	0.0	0.0
Day3_1.2	349	78.2	27.8	Day6_1.2	324	2.0	2.0
Day3_1.3	378	80.5	62.0	Day6_1.3	351	14.1	4.4
Day3_1.4	407	165.1	104.6	Day6_1.4	378	24.1	6.5

is negligible. For Day3_1.0 (a regular operations day), all departures are able to meet their allocated CTOT (with the exception of 1-2 flights). As the airport traffic increases, the number of departures that are unable to take-off at their allocated time windows increases, both in case of RS-GM-sim and RS-GM-seq, while this number is significantly higher for sequences obtained with RS-GM-sim.

As expected, the runway sequencing solutions obtained with RS-GM-seq include a significant number of arrivals and departures with infeasible sequencing slots (see columns '#Infeas. arr.' and '#Infeas. dep.'), especially in case of increased traffic. On the other hand, the proposed algorithm always manages to find a feasible arrival/departure slot for all aircraft. The increase of runway sequence infeasibility for RS-GM-seq is directly linked to the average increase in taxiing delay (see column 'Taxi delay'), stemming from a higher degree of traffic congestion at taxiways and runway crossing. Overlapping histograms showing taxiing delays for RS-GM-seq and RS-GM-sim on Day3_1.0 and Day3_1.1 are given in Fig. 6. For RS-GM-sim, the taxiing delay never exceeds 5 minutes and is under a minute in most cases. On the other hand, the taxiing delay could go up to 25 minutes for RS-GM-seq on instance Day3_1.0, and further increases for Day3_1.1. The shorter delays with RS-GM-sim are partly because a departure that cannot meet its CTOT window is delayed while still on gate/stand.

As for the computing time requirements per horizon, it greatly varies depending on the amount of traffic at a given horizon, and hence on the difficulty of finding a feasible routing solution for the runway sequences at hand. Plots in Fig. 7 compare average computing times per horizon for RS-GM-sim and RS-GM-seq on instances Day3_1.0, Day3_1.1, Day3_1.2 and Day3_1.3. As expected, we observe that the computing time per horizon for RS-GM-seq is generally considerably less than in case of RS-GM-sim, due to only one call of the iterated routing heuristic



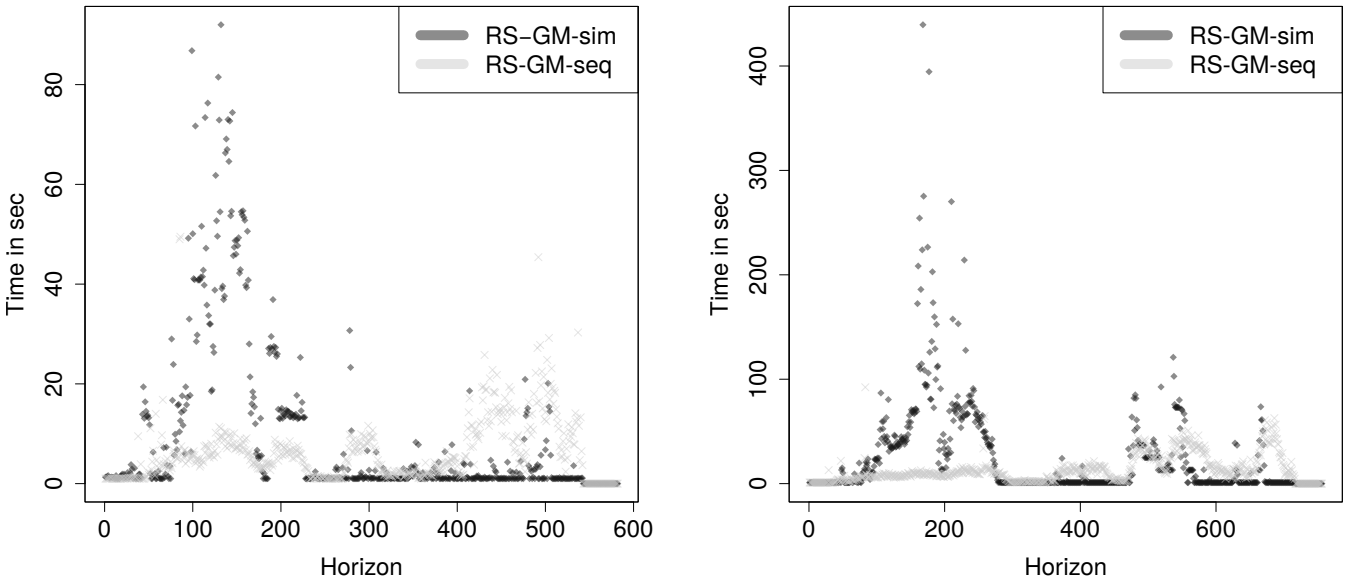
(a) Day3_1.0 - Taxiing delays

(b) Day3_1.1 - Taxiing delays

Fig. 6. Histograms comparing average taxiing delays per aircraft for solutions obtained with RS-GM-sim and RS-GM-seq. Delays are expressed in seconds.

(at the end of the runway sequence optimization phase). The computing time per horizon for RS-GM-seq on these four instances varies from less than a second up to 374.5 seconds, and from less than a second up to 439.5 seconds for RS-GM-sim. In case of a regular day of operations at the Manchester airport (considering instances Day1_1.0 to Day6_1.0), the maximum computing time per horizon for RS-GM-sim is 97 seconds. For an increase of 10% in the amount of traffic (instances Day1_1.1 to Day6_1.1), the maximum computing time per horizon with RS-GM-sim increases to 290 seconds. Because the replanning occurs every 40 seconds (Clare and Richards, 2011), this is not fast enough for real-time operation. However, the difference in the case of regular operation days at Manchester Airport is small enough to suggest that, with increased computing power, real-time operation might be practical.

To conclude, for regular days of operations at the Manchester Airport, the feasibility constraint, imposed on runway sequencing by taxiway routing, does not have a significant impact on arrival and departure delays. In all the cases, the RS-GM-sim sequences are feasible in terms of the corresponding routing solution, while the number of departures that are unable to meet their allocated CTOT window is kept to a minimum. Since the computing time per replanning horizon varies from less than a second up to 97 seconds for a regular day of operations, the proposed method could perhaps be considered for practical use on a high speed processor.



(a) Day3_1.0

(b) Day3_1.3

Fig. 7. A comparison on average computing times per horizon for solutions obtained with RS-GM-sim and RS-GM-seq. Times are given in seconds.

5 Conclusion

As discussed in previous research, the ground movement problem forms the link between arrival and departure sequencing processes, especially when runway crossing is necessary for taxiing aircraft. Furthermore, an optimal departure sequence is of no use if aircraft cannot reach the runway at allocated take-off times. In this paper, we have presented a novel heuristic for simultaneous optimization of the runway sequencing and the ground movement problems, which takes into account the interactions between arrival and departure aircraft on the airport surface. The minimization of the sequencing delay and the minimization of the taxiing delay are considered as the primary and secondary objectives respectively. By the end of execution, the algorithm ensures a feasible runway sequencing solution in terms of taxiway routing, i.e., all sequencing slots can be met by the taxiing aircraft. To improve computational scalability, the approach is based on the Receding Horizon framework which considers only a "window" of aircraft from the entire problem, with aircraft entering and leaving the problem at each horizon.

To evaluate the potential of the proposed approach, we use the Manchester Airport as a test case. The airport has two runways, operated in segregated mode during busy periods, and a runway crossing. We compare our results with sequentially optimized runway sequencing and routing solutions, on a set of 36 instances with varying degrees of traffic density. Compared to sequentially optimized solutions,

the results obtained with our approach indicate a significant decrease in the taxiway routing delay, with generally no loss in performance in terms of the sequencing delay for a regular day of operations. The approach could perhaps be considered for use in practice on a high speed computer, since the maximum computing time per horizon is around 95 seconds. Another benefit of a simultaneous optimization approach is the possibility of holding aircraft at the stands for longer, without the engines running. This significantly reduces the fuel burn, as well as bottlenecks and traffic congestion during peak hours that are often the cause of flight delays due of limited amount of airport surface space available.

Acknowledgment

This work was funded by the UK EPSRC, under project *SANDPIT: Integrating and Automating Airport Operations* [grant number EP/H004424/2]. Results were obtained using the EPSRC funded ARCHIE-WeSt High Performance Computer (www.archie-west.ac.uk). [EPSRC grant no. EP/K000586/1]. We are also grateful to Manchester Airport for the provision of data on which the benchmark problems were based, and to the anonymous referees for valuable suggestions and comments.

Data access statement

The benchmark data sets generated as part of this work are available under `http://dx.doi.org/10.5281/zenodo.21027`. The taxiway layout and aircraft movement data arising from FR24 cannot be shared due to licensing restrictions.

A Algorithm pseudo-code

Detailed pseudo-code of the proposed algorithms including: the receding horizon scheme based on iterated local search optimization (RHC-ILS), and the proposed ILS and IRH algorithms.

Algorithm 1 RHC-ILS

Require: F: set of arrivals and departures ordered according to PLTs/PTOTs;

Ensure: S'_A and S'_D: arrival and departure sequences;
R: routing solution for sequences S'_A and S'_D.

$S_A \leftarrow S_D \leftarrow \emptyset$ /*Arrival and departure sequences of active flights*/
$R \leftarrow \emptyset$ /*Routing solution*/
$S'_A \leftarrow S'_D \leftarrow \{\}$ /*Sequences of inactive arrival and departure aircraft*/

/*At each planning horizon*/
for all $t_i, i = 1, \ldots, N$ **do**
/*Get the set A of active arrivals and departures from S_A and S_D at horizon t_i*/
  $A \leftarrow getActiveAircraft(F \setminus (S_A \cup S_D), t_i)$
  for all $a \in A$ **do**
/*Add a at the end of the arrival or the departure sequence*/
    if $(type(a) = arrival)$ **then**
      $S_A \leftarrow S_A \cup \{a\}$
    else
      $S_D \leftarrow S_D \cup \{a\}$
    end if
  end for
  $R \leftarrow Route(A, R)$ /*Route the set of active aircraft at the current horizon*/
/*Perform token-ring local optimization of the arrival and the departure sequences*/
  while (stopping condition not met) **do**
    $S_A \leftarrow ILS(S_A, R)$
    $S_D \leftarrow ILS(S_D, R)$
  end while
/*Get the set I of inactive arrivals and departures from S_A and S_D at end of t_i*/
  $I \leftarrow getInactiveAircraft(S_A \cup S_D, t_i)$
/*For all $i \in I$, add i at the end of the sequence of inactive arrivals/departures if the sequencing slot of i is feasible. Otherwise, delay the landing/departure of i by δ min.*/
  for all $i \in I$ **do**
    if ($isSequencingSlotFeasible(i)$=true) and $(type(i) = arrival)$ **then**
      $S'_A \leftarrow S'_A \cup \{i\}$
      $S_A \leftarrow S_A \setminus \{i\}$
    else if ($isSequencingSlotFeasible(i)$=true) and $(type(i) = departure)$ **then**
      $S'_D \leftarrow S'_D \cup \{i\}$
      $S_D \leftarrow S_D \setminus \{i\}$
    else if ($isSequencingSlotFeasible(i)$=false) and $(type(i) = arrival)$ **then**
      $PLT(i) \leftarrow PLT(i) + \delta$;
    else
      $PTOT(i) \leftarrow PTOT(i) + \delta$;
    end if
  end for
end for
/*At this point $|S'_A \cup S'_D| = |F|$*/

Algorithm 2 $ILS(S, R_{best})$

Require: S: initial arrival or departure sequence;
$maxiter$ (parameter): number of iterations of the sequence optimization procedure;
r_1 and r_2 (parameters): Range for the number of tabu-based moves.
Ensure: S_{best}: Optimized arrival or departure sequence.
$S_{best} \leftarrow S$
/*Apply descent-based local search*/
for i:=1 to max_ILS_iter **do**
  while (Local optimum not reached) **do**
    $S \leftarrow TransformSequence(S, S_{best}, R_{best}, true)$
  end while
  /*Apply tabu-based perturbation*/
  $n \leftarrow RandomNumberInRange(r_1, r_2)$
  for j:=1 to n **do**
    $S \leftarrow TransformSequence(S, S_{best}, R_{best}, false)$
  end for
end for
return S_{best}

Algorithm 3 TransformSequence(S, S_{best}, R_{best}, $prohib\_lifted$)

Require: S: initial arrival or departure sequence;
S_{best}: best found arrival or departure sequence;
$prohib\_lifted$: boolean variable indicating whether move prohibitions are lifted;
R_{best}: routing solution for the best found arrival S_A^{best} and departure S_D^{best} sequence

Ensure: S': neighboring (transformed) solution of S;
S_{best}: Updated best arrival or departure sequence.

$m \leftarrow null$ /*Selected move to be applied to S*/
$e \leftarrow \infty$ /*Evaluation function value for solution $S \bigoplus m$*/

/*Determine move m to be applied to S*/
for all flight pairs (f', f'') in S, $f' \neq f''$ visited in random order **do**
  if $((e > eval_1(S \bigoplus (f', f'')))$ **and** $(prohib\_lifted = true$ or $is\_tabu(f', f'') = false))$ **then**
  /*Record best move and corresponding evaluation function value*/
    $e \leftarrow eval_1(S \bigoplus (f', f''))$
    $m \leftarrow (f', f'')$
  end if
end for
$S' \leftarrow S \bigoplus m$ /*Apply selected move to S to obtain a transformed sequence S'*/
$tabu(reverse(m)) \leftarrow iter + \gamma$ /*Prohibit reverse move for γ iterations*/

/*Update best found sequence in case of improvement*/
if $((eval_1(S') \leq eval_1(S_{best})$ and $eval_2(R_{best}) \neq 0)$ or $((eval_1(S') < eval_1(S_{best}))$ **then**
  $R' \leftarrow IRH(S', S_{best})$ /*See Alg. 4*/
  if $(eval_2(R') < eval_2(R_{best}))$ **or**
    $((eval_2(R') = eval_2(R_{best})$ and $(eval_1(S') < eval_1(S_{best}))$ **or**
    $(eval_2(R') = eval_2(R_{best})$ and $eval_3(R') < eval_3(R_{best}))$ **then**
      $S_{best} \leftarrow S'$
      $R_{best} \leftarrow R'$
  end if
end if
$iter \leftarrow iter + 1$
return S'

Algorithm 4 $IRH(S', S_{best})$

Require: S': transformed (neighboring) arrival or departure sequence;
S_{best}: best found arrival or departure sequence;
R_{best} (global variable): routing solution for the best found arr. and depart. seq.;
ϕ (parameter): maximal number of consecutive iterations without improvement;
γ (parameter): minimal time difference (in seconds).
Ensure: R': routing solution for the current sequencing solution S';

/*Get initial set UR of flights for unrouting and routing; AT is the actual landing or take-off time (depending on the sequence type) */
$UR \leftarrow \{S'(i) | AT(S'(i)) \neq AT(S^{best}(i))\}$
$R' \leftarrow Unroute(UR, R_{best})$
$R' \leftarrow Route(UR, R')$
$IS \leftarrow FlightsWithInfeasibleSequencingSlots(R')$ /* IS is a subset of active aircraft at the current planning instant*/
$e \leftarrow eval_3(R')$ /*Total taxi delay corresponding to the best found routing solution R' for S'*/
$R_{tmp} \leftarrow R'$
$impr \leftarrow 0$ /*Initialize the number of consecutive iterations without feasibility improvement */

while ($eval_2(R') \neq 0$ and $impr < \phi$) **do**
  $r \leftarrow SelectFlightAtRandom(IS)$
  $UR \leftarrow \{f | abs(AT(r) - AT(f)) < \gamma\}$
  $UR \leftarrow Order(UR)$
  $R_{tmp} \leftarrow Unroute(UR, R_{tmp})$
  $R_{tmp} \leftarrow Route(UR, R_{tmp})$
  $IS \leftarrow FlightsWithInfeasibleSequencingSlots(R_{tmp})$
  if ($eval_3(R_{tmp}) < e$) **then**
    $impr \leftarrow 0$
    $R' \leftarrow R_{tmp}$
    $e \leftarrow eval_3(R_{tmp})$
  else
    $impr \leftarrow impr + 1$
    $R_{tmp} \leftarrow R'$
  end if
end while
return R'

Fig. B.1. Part of the undirected graph representing Manchester Airport

B Ground movement algorithm

B.1 Representation and definitions

The routing algorithm we adopt is a variant of the Quickest Path Problem with Time Windows (QPPTW) algorithm (Ravizza et al., 2013a). This resembles Dijkstra's shortest path algorithm, and routes the aircraft sequentially. Prior to describing the algorithm, some definitions are necessary.

The airport layout is represented as an undirected graph $G = (V, E)$ (Figure B.1). Edges E represent taxiways and the vertices V represent stands, junctions and intermediate points. Each edge $e \in E$ has a set of weights W_e: these are the times to traverse the edge depending on previous edge in the route, the airport operating mode and the aircraft type (i.e., arrival or departure). Each e may only contain one aircraft at any one time, and aircraft must maintain a separation of 60 meters at all times.

To ensure conflict-free routing, every e has a set of time windows $\mathscr{F}(e)$, representing the times that the edge can be used as part of a new route. The $\mathscr{F}(e)$ exclude times when e or an edge which conflicts with e are used by previously routed aircraft. After a route is allocated for each aircraft, the $\mathscr{F}(e)$ are updated, ensuring that routes allocated later avoid it. In our implementation, a history of the $\mathscr{F}(e)$ for all e is retained to allow time windows to be reinstated if aircraft routes are removed for re-routing.

B.2 The QPPTW algorithm

We now summarise the QPPTW algorithm for convenience: further details can be found in (Ravizza et al., 2013a). Runway times are assumed to be fixed. The algorithm constructs the route for all aircraft working out from the vertex representing entry to, or exit from, the runway. Given a taxi request $T_i = (q_i, p_i, time_i)$ for aircraft f_i, QPPTW finds the conflict-free route R for f_i from vertex q_i to p_i over G, with the minimal taxi time $\mathscr{T}_i$, that respects the time-windows in E. QPPTW iteratively applies labels to each vertex v_L, specifying the earliest time that the aircraft could reach v_L. These are stored in a Fibonacci heap and updated as the algorithm explores the G and the set of time-windows on each edge. Each iteration, the label L representing the shortest-time path from the runway is removed from the heap, and the labels on vertices neighbouring v_L are updated. During this process, the time-windows on each outgoing edge from v_L are checked, so that new labels are only created if there is a suitable time-period during which the aircraft can transit along the edge. In working out from the runway to construct the route, departures are routed backwards. This means that any additional wait time is absorbed at the start of the aircraft movement, thus allowing the engines to start as late as possible, reducing fuel consumption and emissions.

It has been shown (Stenzel, 2008) that the variants of this algorithm will solve the problem in polynomial time in the number of time-windows: $O(|\mathscr{F}|^3 \log |\mathscr{F}|)$.

B.3 Taxi time estimation

QPPTW depends on having accurate estimates of the time w_e that aircraft take to traverse edges. This is influenced by many factors (Ravizza et al., 2013b; Idris et al., 2002; Rappaport et al., 2009; Balakrishna et al., 2010).

Following the comparisons of models in (Stefan et al., 2014), as with (Ravizza et al., 2013a), we use the Mamdani fuzzy rule-based system of (Chen et al., 2011), with factors identified in (Ravizza et al., 2013b). A taxi time model is constructed using historical aircraft movements, considering the factors of: airport operating mode; whether an aircraft is departing or arriving; total distance covered; total turning angle; whether a push-back manoeuvre was performed; and number of other moving aircraft of various types. To predict times for QPPTW, factors related to other moving aircraft are zeroed, allowing the model to estimate unimpeded taxi-times.

The flight movement data used to train the model represents real aircraft movements taken from freely-available data on the website FlightRadar24 (FR24), using the tools available at `https://github.com/gm-tools/gm-tools/wiki` and described in (Brownlee et al., 2014). ADS/B from FR24 data was also used for

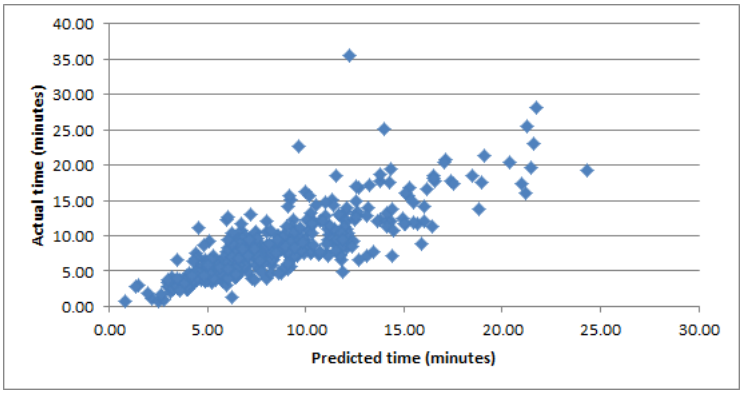


Fig. B.2. Plot of taxi time estimates for validation data.

gathering airborne flight tracks (Petersen et al., 2013; Turner et al., 2013). The co-ordinate points have a resolution of 10^{-4} degrees, approximately 10m at the latitude of Manchester, and are timestamped in intervals of 5-10s.

For this work, all available tracks during 5-12 November 2013 for aircraft with an altitude of zero within 5km of the airport's center were collected, comprising 1767 flights (over the same period, according to the public flight times on the web, there were 3211 flights). 1413 aircraft ground movements remained after processing, showing taxi routes with timings along them. This data was divided at random into training and test sets of 971 and 442 aircraft movements. After training, the model was found to fit the validation data with $R^2 = 0.68$, with 81% of movements accurate to within 3 minutes and 94% accurate to within 5 minutes. A plot showing the time estimates for the validation data is given in Figure B.2.

References

J. Atkin, E. Burke, J. Greenwood, and D. Reeson. Hybrid metaheuristics to aid runway scheduling at london heathrow airport. *Transportation Science*, 41(1): 90–106, 2007.

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

J. A. D. Atkin, E. K. Burke, and S. Ravizza. The Airport Ground Movement Problem: Past and Current Research and Future Directions. In *4th International Conference on Research in Air Transportation*, pages 131–138, 2010.

P. Balakrishna, R. Ganesan, and L. Sherry. Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures. *Transportation Research Part C: Emerging Technologies*, 18(6):950–962, Dec 2010.

J. Beasley, M. Krishnamoorthy, Y. Sharaiha, and D. Abramson. Scheduling aircraft landings - the static case. *Transportation Science*, 34(2):180–197, 2000.

J. Bellingham, Y. Kuwata, and J. How. Stable receding horizon trajectory control for complex environments. In *AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAA 2003-5635*, 2003.

D. Bertsimas and M. Frankovich. Unified optimization of traffic flows through airports. *Transportation Science, Published Online in Articles in Advance*, 2015.

L. Bianco, P. Dell'Olmo, and S. Giordani. Scheduling models and algorithms for tma traffic management. In L. Bianco, P. DellâĂŹOlmo, and A. Odoni, editors, *Modelling and Simulation in Air Traffic Management*, Transportation Analysis, pages 139–167. Springer Berlin Heidelberg, 1997.

C. Bosson, M. Xue, and S. Zelinski. Optimizing integrated arrival, departure and surface operations under uncertainty. In *10th USA/Europe ATM R&D Seminar (ATM2015)*, Lisbon, Portugal, 2015.

A. Brownlee, J. Atkin, J. Woodward, U. Benlic, and E. Burke. Airport ground movement: Real world data sets and approaches to handling uncertainty. In *Proc. of the Practice and Theory of Automated Timetabling*, York, UK, 2014.

J. Chen, S. Ravizza, J. Atkin, and P. Stewart. On the utilisation of fuzzy rule-based systems for taxi time estimations at airports. In *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, pages 134–145. 2011.

Civil Aviation Authority. Aviation trends - quarter 3 2013, 2013. URL `http://www.caa.co.uk/docs/80/AviationTrends_Q3_2013.pdf`. Retrieved 5/2/2015.

Civil Aviation Authority. Size of reporting airports february 2013 - january 2014, 2014. URL `http://www.caa.co.uk/docs/80/airport_data/201401/Table_01_Size_of_UK_Airports.pdf`. Retrieved 5/2/2015.

G. Clare and A. Richards. Optimization of taxiway routing and runway scheduling. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1000–1013, Dec 2011.

R. Dear. *The Dynamic Scheduling of Aircraft in the Near Terminal Area*. PhD thesis, Flight Transportation Laboratory, Cambridge, 1976.

R. Deau, J. Gotteland, and N. Durand. Runways sequences and ground traffic optimisation. In *International Conference on Research in Air Transportation, ICRAT 2008*, Jun 2008.

A. Ernst, M. Krishnamoorthy, and R. Storer. Heuristic and exact algorithms for scheduling aircraft landings. *Networks*, 34(3):229–241, 1999.

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`. Retrieved 21/7/2015.

F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3):190–260, 1989.

F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

X. Hu and W. Chen. Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. *Engineering Applications of Artificial Intelligence*, 18(5):633 – 642, 2005.

X. Hu and E. D. Paolo. Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling. *Intelligent Transportation Systems, IEEE Transactions on*, 9(2):301–310, June 2008.

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

Y. Jung, T. Hoang, J. Montoya, G. Gupta, W. Malik, L. Tobias, and H. Wang. Performance evaluation of a surface traffic management tool for dallas/fort worth international airport, 2011. Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM2011).

H. Lee and H. Balakrishnan. A comparison of two optimization approaches for airport taxiway and runway scheduling. In *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st*, pages 1–28, Oct 2012.

G. D. Maere and J. Atkin. Pruning rules for optimal runway sequencing with airline preferences. In *7th International Conference on Applied Operational Research*, pages 76–82, 2015.

B. Mirković, V. Tošić, P. Kanzler, and M. Höhenberger. Airport apron roundabout - operational concept and capacity evaluation. *Transportation Research Part C: Emerging Technologies*, pages –, 2016.

J. Montoya, Z. Wood, and S. Rathinam. Runway scheduling using generalized dynamic programming. In *AIAA Guidance, Navigation, and Control Conference*, 2011.

U. Neuman and J. Atkin. Gate assignment considering ground movement. volume 8197 of *Lecture Notes in Computer Science*, pages 184–198. 2013.

J. Nosedal, M. Piera, A. Solis, and C. Ferrer. An optimization model to fit airspace demand considering a spatio-temporal analysis of airspace capacity. *Transportation Research Part C: Emerging Technologies*, 61:11 – 28, 2015.

C. Petersen, M. Mühleisen, and A. Timm-Giel. Evaluation of the aircraft distribution in satellite spotbeams. In T. Bauschert, editor, *Advances in Communication Networking*, volume 8115 of *Lecture Notes in Computer Science*, pages 46–53.

Springer Berlin Heidelberg, 2013.

H. Psaraftis. *A Dynamic Programming approach to the Aircraft Sequencing problem*. PhD thesis, Flight Transportation Laboratory, Cambridge, 1978.

D. Rappaport, P. Yu, K. Griffin, and C. Daviau. *Quantitative Analysis of Uncertainty in Airport Surface Operations*, pages 1–16. American Institute of Aeronautics and Astronautics, Sep 2009.

S. Ravizza, J. Atkin, and E. Burke. A more realistic approach for airport ground movement optimisation with stand holding. *Journal of Scheduling*, 17(5):507–520, Oct 2013a.

S. Ravizza, J. Atkin, M. Maathuis, and E. Burke. A combined statistical approach and ground movement model for improving taxi time estimations at airports. *J. Oper. Res. Soc.*, 64(9):1347–1360, 2013b.

P. Roling and H. Visser. Optimal airport surface traffic planning using mixed-integer linear programming. *International Journal of Aerospace Engineering*, 2008(1):1:1–1:11, Jan. 2008.

R. Stefan, J. Chen, J. Atkin, P. Stewart, and E. Burke. Aircraft taxi time prediction: Comparisons and insights. *Applied Soft Computing*, 14(Part C):397 – 406, 2014.

B. Stenzel. *Online Disjoint Vehicle Routing with Application to AGV Routing*. PhD thesis, TU Berline, Germany, 2008.

R. Turner, S. Bottone, and C. Stanek. Online variational approximations to non-exponential family change point models: With application to radar tracking. In *Proceedings of NIPS 26*, pages 306–314. The Neural Information Processing Systems Foundation, December 2013.

M. Weiszer, J. Chen, and G. Locatelli. An integrated optimisation approach to airport ground operations to foster sustainability in the aviation sector. *Applied Energy*, 157:567–582, Nov 2015.

C. Yu and H. Lau. Integrated optimization of airport taxiway and runway scheduling. *Jounal of Automation and Control Engineering*, 2(4):338–342, Dec 2014.

Z. Zhan, J. Zhang, Y. Li, O. Liu, S. Kwok, W. Ip, and O. Kaynak. An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):399–412, 2010.