Annals of Operations Research manuscript No.
(will be inserted by the editor)

Lessons from Building an Automated Pre-Departure Sequencer for Airports

Daniel Karapetyan · Jason A.D. Atkin · Andrew J. Parkes · Juan Castro-Gutierrez

the date of receipt and acceptance should be inserted later

Abstract Commercial airports are under increasing pressure to comply with the Eurocontrol Collaborative Decision Making (CDM) initiative, to ensure that information is passed between stakeholders, integrate automated decision support or make predictions. These systems can also aid effective operations beyond the airport by communicating scheduling decisions to other relevant parties, such as Eurocontrol, for passing on to downstream airports and enabling overall airspace improvements.

One of the major CDM components is aimed at producing the target take-off times and target startup-approval times, i.e. scheduling when the aircraft should push back from the gates and start their engines and when they will take off. For medium-sized airports, a common choice for this is a "Pre-Departure Sequencer" (PDS). In this paper, we describe the design and requirements challenges which arose during our development of a PDS system for medium sized international airports. Firstly, the scheduling problem is highly dynamic and event driven. Secondly, it is important to end-users that the system be predictable and, as far as possible, transparent in its operation, with decisions that can be explained. Thirdly, users can override decisions, and this information has to be taken into account. Finally, it is important that the system is as fair as possible for all users

This work was supported in part by an EPSRC 'Research Development Fund' and also by EPSRC grant EP/F033613/1

D. Karapetyan
School of Computer Science, University of Nottingham
E-mail: daniel.karapetyan@gmail.com

J.A.D. Atkin
School of Computer Science, University of Nottingham
E-mail: jaa@cs.nott.ac.uk

A.J. Parkes (✉)
School of Computer Science, University of Nottingham
E-mail: ajp@cs.nott.ac.uk

J. Castro-Gutierrez
School of Computer Science, University of Nottingham
E-mail: jpcastrog@gmail.com

of the airport, and the interpretation of this is considered here. Together, these factors have influenced the design of the PDS system which has been built to work within an existing large system which is being used at many airports.

Keywords Automated decision support · Scheduling · Aviation · Airport ground operations · Modelling user preferences · Collaborative decision making

1 Introduction

Ongoing improvements to the European-wide airspace require all parties to share information with each other and to utilise the information which they are provided in order to make better decisions. To aid this, organisations such as Eurocontrol[1], who are responsible for the management of the airspace over Europe, have introduced and promoted "Collaborative Decision Making" (CDM) systems (for example, their CDM web site[2] provides many resources). One important element of CDM, from the point of view of airports, is that they provide to Eurocontrol predicted take-off times (TTOTs, "Target Take-Off Times") and pushback times (TSATs, "Target Start-up Approval Times", the times at which aircraft push back from their stands and start their engines).

This paper primarily considers the process of TSAT generation at a medium sized international airport. Ultra Electronics Airport Systems[3] has systems in many such airports worldwide. This paper describes a system which can collaborate with the existing Ultra systems in order to generate TSATs for airports. A real-world operational research problem is considered and the real-world elements had a crucial effect upon the decision making process and the system design. The airport environment, existing system, communication interfaces, user interaction and commands, and the preferences of the airlines and other airport users all had important effects upon the design of this system and are discussed in this paper, along with the algorithms which were developed to solve the problem. The real-world complexities that this introduces are important problems to consider and have influenced the design of the system. On the other hand, the situation has also ensured that the TSAT generator is self-contained and that the decision making logic can be described in a stand-alone manner.

In particular, the need for ease of explanation, with the consequent ability to build trust by the users, led us to implement an algorithm that was (iterated) constructive/rule-based. The system is built out of 'elementary components', meaning that the individual operations are themselves in a language that would make sense to the end users. For example, using priority-based allocations, rather than having recourse to more advanced (and harder to explain) optimisation methods such as maximum weight matching or stochastic search. Much of the previous research into airport operations has involved the application of optimisation algorithms where the decision making process is based upon an objective function and the reasons for individual decision elements may not be obvious, since an automated search often works in a different manner to human decision making. For

[1] Eurocontrol web site: `https://www.eurocontrol.int/`

[2] Eurocontrol CDM web site: `https://www.eurocontrol.int/services/acdm`

[3] Ultra Electronics Airport Systems web site: `http://www.ultra-as.com/`

example, runway sequencing research (which is perhaps the closest common academic problem to that which is considered here) usually uses an objective function which optimises for makespan (Newell, 1979; Bolander, 2000), total delay (Bianco et al, 1999) or deviation from target runway times (Beasley et al, 2000; Ernst et al, 1999). Where the re-sequencing is limited to try to enforce fairness, it is usually for reasons of lowering the cost assessed by the objective function, rather than considering explicit movement, and involved either applying maximum position shifts (Dear and Sherif, 1989; Psaraftis, 1990; Balakrishnan and Chandran, 2010), adding a penalty factor into the objective function to penalise positional shifts (Atkin et al, 2007; Beasley et al, 2004), using a non-linear factor for delay (Beasley et al, 2001), or a combination of two or more of these (Atkin et al, 2013). A particularly good example of this is the consideration of the dynamic sequencing problem in Beasley et al (2004), where the deviations from a previous solution were explicitly penalised, and the difficulties of obtaining an objective function which will match the decision making of the human controllers at Heathrow was observed in Atkin et al (2010a). A review of airport runway scheduling can be found in Bennell et al (2011), showing the extent of the academic research into the problem at that time, and an earlier broader review can be found in Wu and Caves (2002). The use of an optimisation function rather than swapping rules is also not uncommon in other problems as well, as exemplified by the objective function for gate allocation research (see Dorndorf et al (2007)) and ground movement research (see Atkin et al (2010b)).

In contrast, in this research it was found that the justification for decision making was key for ensuring its acceptance and has had a significant influence upon the design of the system, since justifications of each change rather than the overall result were needed in order to persuade the stakeholders at the airport to accept the decisions. Of course, the elementary nature of the algorithm components does not imply that the final result is elementary; and it actually took a number of iterations and careful design to arrive at a final system which was acceptable to users from the point of view of both efficacy and explainability.

The system must also react to user input, a situation which is too often ignored within academic models (for example, none of the academic papers cited above include user input in the decision making) and decisions had to be made about how to handle any conflicts between decisions which are made by user and automated decisions about changes based upon a constantly evolving situation at the airport. Indeed, it is quite possible for a situational change to be handled by the decision making algorithms before the user really becomes aware of it, in which case it would be inappropriate to apply a constraint which was based upon out of date information. For this reason, although the focus of the paper is upon the TSAT generation algorithm (the decision making element of the system), some other elements of the overall system are also briefly summarised, since these form the environment within which the algorithm has to execute and had important effects upon the design of the algorithm. We have endeavoured to keep such explanations to a minimum and to only discuss those elements which are key for understanding the system.

These real-world elements are considered in the following sections: Section 2 explains the problem which is being solved and discusses the relevant previous research. Section 3 considers the overall system architecture and the components which are relevant for the TSAT generation module, which is the real focus of this

paper. Section 4 formalises the TSAT generation problem, discusses the input and output data for the TSAT generator, and presents the TSAT generation algorithm and the options which are possible for its customisation. Section 5 discusses potential future enhancements of the system. Finally Section 6 summarises the key lessons from this research, which resulted in the development of a live decision support system for medium sized international airports.

2 Problem Description

Each departing flight in a commercial airport typically follows a sequence of steps:

1. The aircraft crew and other airport services report the time at which the flight will be ready to depart, called the TOBT (Target Off-Block Time).
2. The airport controllers schedule the 'off-block' / 'push back' times based upon these TOBTs and other information. The procedure is usually for a tug to push the aircraft back from the stand, since most of the aircraft are not capable of moving in the reverse direction without risking damage to the stands.
3. The aircraft is pushed back and its engines are turned on.
4. The aircraft taxies towards the end of the runway where it may join a queue of departing aircraft.
5. The aircraft taxies onto the runway, lines up and takes off.

TSATs (Target Start-up Approval Times) are generated at the airport, to specify the time at which an aircraft should commence its pushback (step 2 above) and start its engines. TSAT allocation is an important process, as highlighted by a Eurocontrol brochure about "Airport CDM" (A-CDM) (EUROCONTROL and ACIEurope, 2010), which states that some key aims are:

> *"Information sharing is the first and most essential element of A-CDM as it creates the foundation by creating a common situational awareness. In addition, it potentially brings predictability and resource efficiency benefits. ... With the pre-departure sequencing function the target start-up approval time (TSAT) can be calculated, providing an off-block sequence."*

In common with many systems, the system described in this paper achieves TSAT allocation by building a predicted take-off sequence even before aircraft leave the stands (a pre-departure sequence), then allocating TSAT times to aircraft which will allow them to achieve these take-off times.

Although the TSAT allocation process is usually common across airports, the details can vary greatly depending upon the constraints at each airport. These constraints will depend upon the demand (how many flights want to take off from the airport at any time), capacity (i.e. how many flights can it reasonably accommodate at that time), layout (e.g. complex and/or crossing taxiway structures can introduce complexity into the ground movement operations, which may need to be modelled) and operating mode (e.g. which runways are used for arrivals, departures or both).

There are a number of objectives, such as to ensure fairness (equity) across all users (airlines or individual aircraft), to ensure that delays are low and to ensure that sufficient overall capacity is maintained. Delay can be expensive for airlines, who often work on very low profit margins, so equity is often really important,

since earlier take-offs can lead to a competitive advantage for airlines. Interestingly, however, the constraints upon the system will also affect the importance of the objectives. For example, at an airport such as Heathrow, the demand can often exceed capacity, so avoiding wasted capacity can often be a primary objective, even at the cost of some equity loss (Atkin et al, 2008, 2010a, 2013). In such a circumstance, any loss in capacity can lead to cumulative delays for all parties, thus parties will accept some inequity if it can be shown that it will lower their overall delays (and hence costs). On the other hand, if an airport is not running so close to capacity, then it may be more important to ensure fairness, making sure that the required demand is met, rather than to maximise theoretical capacity. In other words, the equity cost for packing departures in as quickly as possible, may not be worthwhile if there is then a gap due to lack of any aircraft waiting.

An important consideration when predicting take-off sequences for most European airports is to comply with any take-off timeslots which are allocated by Eurocontrol. These are allocated to manage bottlenecks in the airspace and these *regulated* flights are issued a "Calculated Take-Off Time" (CTOT), which defines a fifteen minute time window (from CTOT $-$ 5 to CTOT $+$ 10) for the take-off time from the runway. A CTOT window is a hard constraint; if the flight is not ready to depart before the end of the CTOT window, a new CTOT usually has to be requested, which may cause additional delays and associated costs. In other words, any plan which would violate a CTOT is strongly undesirable for the airport.

2.1 Sequence vs Slot-based approaches

The operational mode, in terms of what types of aircraft (landings or take-offs) runways are used for at the time, can affect the importance of the various objectives. When aircraft take-off or land, separations have to be maintained between adjacent runway usages, to ensure that any wake vortices behind the aircraft do not affect the following aircraft, and to allow aircraft the time to fully, safely depart the runway. These 'wake vortex separations' depend upon the aircraft weight categories, and mean that controllers will often group aircraft together by weight category. When runways are used in mixed mode (for both arrivals and departures), it is usually better to alternate take-offs and landings (Newell, 1979), and the individual aircraft which are used, or even their weight classes, may not matter in most cases. In this case it may be sufficient just to ensure that there is a reasonably sized pool of aircraft at the runway from which the controller can choose, rather than planning individual aircraft specifically. In this case, it is often sufficient to provide generic slots for aircraft, and to be relatively unconcerned about which particular aircraft is allocated to each slot.

When a runway is being used for only arrivals or only departures, it is often more important to consider at least the types of the aircraft at the runway, since the separations are sequence-dependent, and changing the sequence can often reduce the total separations needed and hence improve the throughput (Newell, 1979; Beasley et al, 2000; Bolander, 2000; Atkin, 2008). In addition, when aircraft are taking off into busy airspace, downstream constraints may be more likely to affect the separations, and thus it may be important to consider the departure routes as well as the aircraft sizes, and potentially also the aircraft speed groups (Atkin et al, 2008). For example, the London airspace which Heathrow feeds into

can apply significant separation requirements upon aircraft depending upon their departure route, speed groups and weight classes of aircraft. With so many different combinations, exact sequencing of aircraft may be sensible. Such sequencing decisions are far from easy, however, in a constrained system, and even the arrival sequencing problem alone can be complex to solve (Beasley et al, 2000, 2001), with some departure problems being significantly more complex (Atkin et al, 2008) and not amenable to the simplification methods which are used for arrivals, such as applying maximum position shifts or grouping by weight class (Dear and Sherif, 1989; Psaraftis, 1990; Trivizas, 1998; H Balakrishnan, 2010). Conversely, when the airspace is less busy, it may be sufficient to only consider the weight classes of aircraft, and/or the options for controllers and pressure upon the system may be sufficient to allow controllers to make the sequencing decisions in their heads, in which case a generic take-off slot based system may again be sufficient.

One of the problems which has to be faced when targeting specific sequences is that elements such as startup times and taxi times are hardly ever precisely predictable. Significant 'slack' may need to be added into timing to ensure that aircraft arrive when expected and/or to ensure that there is an appropriate selection of aircraft available waiting at the runway that the runway controller can re-sequence in order to achieve an acceptable take-off sequence, even if one or more of the intended aircraft do not arrive at the runway on time. However, if too many aircraft are released, the queues can build up at the runway and can actually make it harder to achieve a desirable take-off sequence, since it may be impossible for the correct aircraft to get past others which are waiting (Atkin et al, 2007). It is therefore of use to reduce this idling with the engines running whenever this is possible. In addition, jet engines can consume 5–7% of the normal fuel burn even on idle, which is a significant cost for airlines as well as causing a significant emission of pollutants into the atmosphere. When it is important to target specific take-off sequences, delays for individual aircraft may be much more significant, and more slack may need to be included into taxi time predictions.

The system described in this paper uses a slot-based system. Since the aim is to ensure that a reasonable number of aircraft are waiting at the runway rather than to generate an exact take-off sequence which aims to minimise separations, a slot-based system is useful as each slot can theoretically be used by any aircraft (although some aircraft could use more than one adjacent slot if they have characteristics which will unusually reduce the throughput of the runway). This targets our system to medium-sized airports feeding into reasonably busy airspace; in contrast, larger/busier airports would benefit from using the sequence-based approach.

2.2 Pre-Departure Sequencers (PDSs) and Departure Managers (DMANs)

A Departure Manager (DMAN) attempts to find take-off sequences and times, which should (or at least could) be adopted. These systems aim to at least emulate a skilled controller, simplifying the decision making problem for the runway controller who has to sequence the aircraft. A DMAN is most useful when the sequencing problem is very complex and not feasible for controllers to be continually solving in their heads. At the most-constrained airports, a DMAN is appropriate

as it is important to ensure that runway capacity is not wasted, but the throughput may be highly dependent upon the exact take-off sequence.

A Pre-Departure Sequencer (PDS) on the other hand will attempt to produce a take-off sequence while aircraft are still at the stands but then apply no commitment to airlines or controllers to actually adhere to the sequence. Although a PDS produces take-off time predictions which are good enough to be of use for improving the airspace, and for giving visibility of delays to Eurocontrol (and hence to downstream airports), it does not constrain the controller to use the exact take-off sequence. These take-off times can still be used to produce TSATs, allocating appropriate stand holds (delays at the stands/gates before starting the engines) to aircraft. In summary, the main difference between a PDS system and a DMAN is the degree of commitment of the airport towards actually achieving the runway sequence plan that the system derives.

DMANs are popular in larger airports (even as early as 2004, a SESAR report (SESAR Joint Undertaking, 2004) listed a number of such DMANs which were in regular use: "in Zürich (DARTS), in Paris-CDG (MAESTRO), Munich (SEPL) and Frankfurt (Sequence Planner)". At least some of the DMANs can provide PDS facilities, such as DARTS delair air traffic systems (2013) and SEPL München (2006). However, at many airports, particularly when the throughput is less dependent upon the individual aircraft which are present at the runway, waiting for take-off, and there is significantly more freedom for controllers to tactically re-sequence aircraft without losing runway throughput, a DMAN may be unnecessary and a PDS may suffice. Many such PDS systems (even when they are provided by a DMAN), including the one presented in this paper, are 'slot-based' (e.g. DARTS (delair air traffic systems, 2013)), meaning that flights are allocated to time-slots of a fixed duration.

Commercial airports are under increasing pressure to ensure that they have a PDS or DMAN system deployed at the airport, to provide the take-off time and TSAT predictions. This paper considers a PDS for TSAT allocation at international airports, which are less constrained than airports such as Heathrow, so that the exact take-off sequence has less effect upon throughput, and which are consequently more concerned about both fairness and transparency of prioritisation rules.

2.3 Requirements

Given the previous explanations, the requirements to the PDS system can be summarised as follows:

- Produce a predicted take-off time for each aircraft, for use by Eurocontrol and for passing on to other parties (e.g. airports) to aid overall system optimisation.
- Ensure that all take-off times meet the allocated CTOT time-slots for regulated flights.
- Ensure that the sequence responds appropriately to changing circumstances.
- Ensure that the prioritisation (slot-allocation) is both transparent (using known rules) and justifiable to all parties.
- Ensure that the system is easily adjustable to the needs and practices of a particular airport and maintainable.

– Ensure reliability of the system in circumstances such as unstable connection with the other components, power cuts off, etc.

3 The Components of the System

In order to understand the full problem which has to be solved, it is important to understand the data which is available, the user commands which have to be supported (and which can override system decision and so will impose constraints), and the environment within which the algorithm has to sit. These real world issues can impose important real world constraints upon any practical system which is to run in airports, most of which will have existing data sharing systems, as a part of the increasing automation in the Airport CDM initiative, even where they do not yet have automated decision support or decision making systems.

The full PDS system can be considered to be a number of discrete but interconnecting modules. An outline of the PDS architecture is shown in Figure 1. All of the decision making elements are in the TSAT Generator, which is described in Section 4 and is the focus of this paper. The other two important modules are the Interface Module, which controls the database, and the Graphical User Interface (GUI). Each of the modules in the system can be restarted at any time, can be located on different machines at different locations, and is relatively independent of the others. This introduces some timing issues which have to be handled, such as the need to deal with events asynchronously, which can imply some additional constraints upon the problem, in that messages or requests can be received which may not be entirely relevant by the time that they are received. In addition, neither the TSAT Generator nor the GUI have direct access to the database but communicate

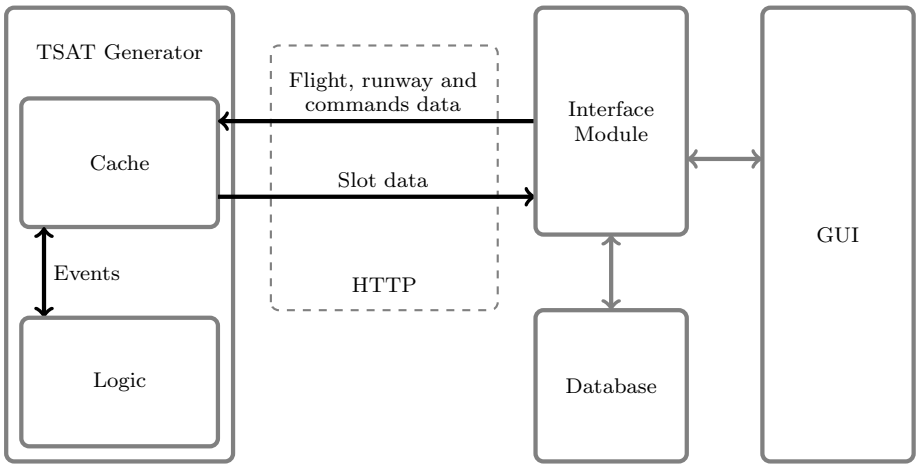

Fig. 1: PDS architecture. The TSAT Generator module communicates with the rest of the system through the Interface Module. The communications are implemented as one-way synchronisation of the flight, runway and commands data from the Interface Module to the TSAT Generator, and of the slot data from the TSAT Generator to the Interface Module.

with the Interface Module by means of an HTTP-based protocol. This aids with ensuring system extendibility and flexibility as changes in the database structure affect only the Interface Module. Using HTTP makes the interface with the TSAT generator easy to maintain and extend, as well as simpler to test.

The Interface Module is not the focus of this paper, despite being an important element of the system, so we only summarise its behaviour from the point of view of TSAT generation. The Interface Module has much of the general control logic for the system, and is responsible for marshalling the information and change requests from the other modules, including those from the other airport systems beyond the TSAT generator. The database actually contains a huge amount of data about the airport, much of which is not relevant for TSAT generation but is vital for the operation of the other systems at the airport. As the situation at the airport changes, the Interface Module is responsible for passing on the information about the new current state to the TSAT Generator module. This means that the TSAT generator will be receiving a number of asynchronous updates and requests, indirectly from many different sources, via a single interface.

The GUI module is responsible for displaying to the user the current state of the airport. The focus of this paper is also not upon the user interface, the design of which would deserve a paper on its own. In summary, the GUI module allows the user to see the current state of the system and to issue commands. These user commands will be converted to constraints upon the TSAT generator, such as to force an aircraft into a specific take-off slot or to lock it into the slot it is currently in. The algorithms then have to respect these constraints. From a practical implementation point of view, the asynchronous nature of the operations, whereby the situation may change since the user looked at it, but before they issue the command, means that the GUI has to pass some information about what it thought the status was at the time the command was issued (e.g. what slot was an aircraft moved from as well as being moved to) and the TSAT generator has to validate that the state was correct before creating the constraint, or reject the command and report that the situation has changed.

4 The TSAT Generator

The TSAT Generator is the key component of the PDS responsible for producing the pre-departure sequences. It handles the following data items:

1. *Flight data*: this contains all of the information about flights including their statuses, time predictions, etc. Flight data comes from the other airport systems and is, therefore, owned by the Interface Module.
2. *Runway data*: this includes the information on logical runways such as runway capacities at different time periods and temporary closure informations. Similarly to the flight data, the runway data is owned by the Interface Module, but will usually change less often.
3. *Slot data*: this defines the TSAT values and associated information as proposed by the TSAT Generator. Hence, the slot data is owned by the TSAT Generator.
4. *User commands data*: this is the list of manual user edits of the pre-departure sequence that require processing by the TSAT Generator. This data is created by the GUI and owned by the Interface Module.

Each data item is owned by a specific module. Modules which do not own the data item have read-only access to it; they have no ability to modify, or delete data of that type. Any editing or deletion of the data has to be performed by the owning module. In general the owning module will also create the data, except for the case of the User Commands data. The owner module has the responsibility for ensuring consistency in the data which it manages, resolving any inconsistencies between the requests which are made and the current state of the data. As an example of the use of this system, if a flight was deleted, the Interface Module is in charge of removing the corresponding record from the flights data and notifying the other modules of the event. However, the Interface Module cannot remove the corresponding slot data as the slot data is owned by the TSAT Generator. Instead, once the TSAT Generator receives the notification of the flight deletion, it removes the corresponding slot data and sends the appropriate update to the Interface Module.

The TSAT Generator can be considered as event based since it updates the slot data (the TSAT values) only when some flight or runway data changes, or when a specific time has expired (which could be considered to be a timer expiry event, even if not implemented that way). Its behaviour depends upon the type of the event and the associated data which accompanies the event notification. While the time-based events can be implemented inside the TSAT Generator, to respond to the flight and runway data changes it needs to be notified of those changes by the Interface Module.

4.1 Data Communication Algorithms

Since the volume of the data items described above can be significant, only the changes themselves are normally communicated. The TSAT Generator needs to have a local copy of the flight and runway data as it cannot request the entire data every time something changes. It will, however, request the entire flight data at intervals (such as every few minutes). This ensures that the system is periodically given a consistent view of data, regardless of the ordering of any change events, and also simplifies matters such as deciding when an aircraft should leave the system.

Conceptually, the communication for each data flow is designed as a one-way synchronisation mechanism. When the TSAT Generator starts, it requests the entire content of the synchronised data item and then subscribes itself to receive any changes in that data item. Internally it maintains two copies of the data: a cache which is stored in data structures compatible with the protocol, and an internal copy which is stored in data structures not directly linked to the protocol. Every time an update is received, the TSAT Generator applies that update to the cache and then runs synchronisation between the cache and the internal storage to identify and handle changes. By comparing these two versions of the data, it detects changes and raises corresponding events internally, regardless of whether the change was identified due to a full data refresh or due to the notification of only that event. When the TSAT Generator starts, it requests the latest version of the slot data (i.e. data item owned by the TSAT Generator). Every time the TSAT Generator changes the pre-departure sequence, it sends the updates to the Interface Module. In addition, at intervals, it re-requests the entire slot data and

replaces the local cache with the downloaded data. If the downloaded version of the data is different to the internal sequence, that will automatically trigger a new message to the Interface module with the update. This approach separates the protocol implementation from the TSAT Generator logic, which in turn reduces the cost of maintenance, and allows the TSAT Generator to store custom data even if the protocol does not support that. It also simplifies the communications protocol as any HTTP failures (which could happen occasionally) can be ignored due to the recovery procedure.

4.2 The TSAT Generation Problem

The TSAT Generation problem can be defined as follows. We are given the set of flights F and the set of current allocations A. For each flight $f \in F$ we are given its ID $n(f)$, TOBT, $t_{\mathrm{TOBT}}(f)$, CTOT $t_{\mathrm{CTOT}}(f)$ and expected taxi-out time (EXOT, $t_{\mathrm{EXOT}}(f)$, the time which the system should expect the aircraft to take from commencing pushback from the stand to arriving at the runway queue, which may include slack to allow for some delay). $t_{\mathrm{CTOT}}(f)$ can have the 'none' value which means that flight f is non-regulated. We also maintain a moves counter $m(f)$ for each flight f, which will be used to suppress changes for flights which otherwise receive too many changes of allocation.

For each existing allocation $a \in A$ we are given the current take-off slot start time $t_{\mathrm{TTOT}}(a)$, the TSAT value $t_{\mathrm{TSAT}}(a)$ and the flight information, $f(a) \in F$. The TSAT and TTOT values are always linked by $t_{\mathrm{TTOT}} = t_{\mathrm{TSAT}} + t_{\mathrm{EXOT}}$; the algorithm sets the TSAT and TTOT together, linking them in this fashion.

Finally, we are given the set of runway records R with prescribed start time $t_{\mathrm{start}}(r)$ and capacity $c(r)$ for each runway record $r \in R$.[4] Without loss of generality, we assume that the runway records $r \in R$ are ordered by the start time $t_{\mathrm{start}}(r)$. Then a runway record $r \in R$ is *active* from $t_{\mathrm{start}}(r)$ to $t_{\mathrm{end}}(r) = t_{\mathrm{start}}(r')$, where $r' \in R$ is the runway record in R immediately following r. If r is the last runway record, it is active forever, i.e. $t_{\mathrm{end}}(r) = \infty$. During the activity period of a runway record r, the capacity of the runway (the number of slots per hour) is $c(r)$. During the runway closures, the runway capacity is assumed to be $c(r) = 0$.

For convenience, let $a(f)$ be the allocation of the flight f:

$$a(f) = \begin{cases} a & \text{if } \exists a \in A \text{ such that } f(a) = f, \\ \text{'none'} & \text{otherwise.} \end{cases}$$

4.3 Re-Sequencing Algorithm

Whenever any change in the flight or runway data happens, or a timer expires, the re-sequencing algorithm is triggered. The re-sequencing algorithm is responsible for keeping the pre-departure sequence up-to-date. Effectively, the re-sequencing algorithm 'fixes' the pre-departure sequence to reflect the data changes. This approach reduces the number of flight moves and improves the system's predictability.

[4] For simplicity we omit some details here such as the existence of two logical runways corresponding to two directions of a physical runway since these have little effect on the main algorithm.

The pseudo-code of the re-sequencing procedure is presented in Algorithm 1. The input of the algorithm is the current flights, runway and slot data (F, R, A') as well as a snapshot F' of the flight data obtained during the last run of the algorithm. The algorithm starts by converting the runway data R into runway time slots S (line 1). Then, by comparing F to F' and matching F, A' and S, the algorithm identifies which flights need to be removed (line 4), added (line 5), or reallocated (usually due to some exogenous change of data) in the sequence (lines 6–7). Then the flights that need to be allocated or re-allocated are ordered according to some logic as described in Section 4.4. Finally, Algorithm 2 is called for each flight to be allocated/reallocated. While allocating a flight, Algorithm 2 may eject another flight from its slot, and for that reason the allocation may need to be repeated. Note that the flight allocation rules have to guarantee that the loop in lines 9–11 will terminate.

Algorithm 1: Allocations update procedure.

input : Flights F, runway records R, allocations A', and a copy F' of the flights since the last run of the procedure
parameters : Slot generation time window w
output : Updated allocations A

1 Generate a set S of runway slots for the period $[\text{now} - w, \text{now} + w]$ excluding the runway closure periods; each $s \in S$ is a triple $(t_{\text{from}}, t_{\text{to}}, B)$ such that $t_{\text{from}} - t_{\text{start}}(r) = \frac{1\text{ hour}}{c(r)} i$, where i is a non-negative integer, r is the runway record active at t_{from}, $t_{\text{to}} = \min\{t_{\text{start}} + \frac{1\text{ hour}}{c(r)},\ t_{\text{end}}\}$ and $B = \emptyset$ is the set of allocations in that slot (note that $|B| \leq 1$ in any circumstances);
2 Let Q be an empty queue of flights for allocation/reallocation;
3 Let $A \leftarrow A'$;
4 For each $a \in A$ such that $f(a) \notin F$ and remove a by $A \leftarrow A \setminus \{a\}$;
5 For each $f \in F$ such that $a(f) =$ 'none', enqueue f to Q and set $m(f(a)) \leftarrow 0$;
6 For each $a \in A$ such that $t_{\text{TOBT}}(f) \neq t_{\text{TOBT}}(f')$ or $t_{\text{CTOT}}(f) \neq t_{\text{CTOT}}(f')$ or $t_{\text{EXOT}}(f) \neq t_{\text{EXOT}}(f')$, where $f = f(a)$ and $f' \in F'$, $n(f') = n(f)$, remove a by $A \leftarrow A \setminus \{a\}$, enqueue f to Q and reset $m(f(a)) \leftarrow 0$;
7 For each $a \in A$, find $(t_{\text{from}}, t_{\text{to}}, B) \in S$ such that $t_{\text{TTOT}}(a) = t_{\text{from}}$; if no such slot exists or $B \neq \emptyset$, then remove a by $A \leftarrow A \setminus \{a\}$, enqueue $f(a)$ to Q and reset $m(f(a)) \leftarrow 0$; otherwise let $B = \{a\}$;
8 Order the flights in Q according to the prioritisation rules (see Section 4.4);
9 **while** $Q \neq \emptyset$ **do**
10 Dequeue f from Q;
11 Call *allocate*(f, A, S, Q) (Algorithm 2);
12 **return** A

4.4 Adjustability of the TSAT Generator

In order to be part of a widely-used system, the TSAT Generator needs to be easily adjustable for the needs of each particular airport. This is achieved by thorough parameterising of all the aspects of its operation and leaving the most important decisions up to some declarative rules that can flexibly define the desired behaviour of the system without affecting the core functions responsible for maintaining data integrity.

Algorithm 2: Function *allocate*(f, A, S, Q) allocating an added/modified flight to the slots.

input : Flight f, allocations A, runway time slots S the queue Q of flights to be allocated, slot filters SUITABLE(1) and SUITABLE(2)
parameters : Slack size σ for creating a pool of aircraft at the runway
output : Updated allocations A, updated slots S and updated flights queue Q

1 **foreach** *pass, p, of the algorithm* **do**
2 Compute the set of slots $S' \subseteq S$ that pass SUITABLE(p), and so can be used for allocating flight f in this pass of the algorithm;
3 **if** $S' \neq \emptyset$ **then**
4 Find $(t_{\text{from}}, t_{\text{to}}, B) \in S'$ that minimises t_{from};
5 **if** $B \neq \emptyset$ **then**
6 Remove the allocation: $A \leftarrow A \setminus B$;
7 Enqueue $f(a')$ to Q, where $\{a'\} = B$;
8 Update the moves counter: $m(f(a')) \leftarrow m(f(a')) + 1$;
9 Create a new allocation a and set $f(a) \leftarrow f$, $t_{\text{TTOT}}(a) \leftarrow t_{\text{from}}$ and $t_{\text{TSAT}}(a) \leftarrow t_{\text{from}} - t_{\text{EXOT}}(f) - \sigma$;
10 Update the allocations: $A \leftarrow A \cup \{a\}$;
11 Update the time slot: $B \leftarrow \{a\}$;
12 **return**

The declarative rules are used for two purposes. Firstly, to give a priority order to the flights when allocating or re-allocating several of them in the same run of the re-sequencing procedure (line 8 of Algorithm 1). Secondly, to find the slot for allocating a flight (see Algorithm 2), and which implicitly define a 'allocation priority' ordering in the sense of which flight have the power to eject out some other flight from its slot.

The criteria that can be used to order the flights before allocation include:

1. The allocated CTOT (for regulated flights). Prioritising by the CTOT will ensure that flights get allocated in an order which reflects the order of their allocated CTOT. As long as all aircraft which are added as a set can meet their CTOTs, this will ensure that they do so.
2. The move counter $m(f)$, i.e. the number of times flight f has been moved. Moving a flight means changes for the airline and/or ground handlers and may mean that scarce resources (e.g. the tugs which are needed for pushing aircraft back from the gates) have to be reallocated. This is obviously undesirable, so can be explicitly penalised. Penalising this factor super-linearly means larger move counts are relatively heavily penalised per move, and so encourages that the changes are more evenly spread across aircraft. Although it could result in a number of smaller changes rather than one larger (less equitable) change, small changes are often easier to accommodate, especially if aircraft still push back in the same relative sequence.
3. The time at which the current TOBT was declared. Giving a higher priority to earlier TOBTs biases the schedule towards a first-come-first-served system whereby airlines which inform the airport about the TOBT earlier are more likely to get the earlier take-off slot. This is useful to discourage late changes to TOBTs, which can have adverse effects on a sequence and force late changes upon other aircraft to compensate for changes.

Slot	ID	Reg.	Moves
9:00–9:02			
9:02–9:04	N1	No	0
9:04–9:06	N2	No	0
9:06–9:08			
9:08–9:10			

(a) Two non-regulated flights are slotted. The earliest suitable slots for both N1 and N2 are 9:02–9:04.

Slot	ID	Reg.	Moves
9:00–9:02			
9:02–9:04	R3	Yes	0
9:04–9:06	N2	No	0
9:06–9:08			
9:08–9:10			

(b) A regulated flight R3 with the first suitable slot 9:02–9:04 enters the system. Being of a higher priority than N1, it ejects N1.

Slot	ID	Reg.	Moves
9:00–9:02			
9:02–9:04	R3	Yes	0
9:04–9:06	N2	No	0
9:06–9:08	N1	No	1
9:08–9:10			

(c) The priority of N1 is lower than that of R3 and it is not higher than that of N1. Hence, it is slotted to the 9:06–9:08 slot.

Fig. 2: Example of prioritisation of regulated over non-regulated flights only. This scheme enforces minimal perturbation, as only one non-regulated flight is moved. The moves counters in this example are not involved in the prioritisation.

4. The TOBT. Ordering by the TOBT can be considered to be fair in that the aircraft which will be ready first will get the earlier slot.

The flight allocation rules define the time slots suitable for a flight and the priority of an allocating flight against the already allocated flights.

For example, we can define two algorithm passes (see Algorithm 2) for a regulated flight f:

1. SUITABLE(1) allows only slots $(t_{\text{from}}, t_{\text{to}}, B)$ such that

$$\max\{t_{\text{CTOT}} - 5,\ t_{\text{TOBT}}(f) + t_{\text{EXOT}}(f)\} \le t_{\text{from}} \le t_{\text{CTOT}} + 10 \text{ and } B = \emptyset.$$

In other words, flight f does not have 'allocation priority' over any flights at this stage; the algorithm attempts to allocate f to a vacant slot.

Slot	ID	Reg.	Moves
9:00–9:02			
9:02–9:04	N1	No	0
9:04–9:06	N2	No	0
9:06–9:08			
9:08–9:10			

(a) Two non-regulated flights are slotted. The earliest suitable slots for both N1 and N2 are 9:02–9:04.

Slot	ID	Reg.	Moves
9:00–9:02			
9:02–9:04	R3	Yes	0
9:04–9:06	N2	No	0
9:06–9:08			
9:08–9:10			

(b) A regulated flight R3 with the first suitable slot 9:02–9:04 enters the system. Being of a higher priority than N1, it ejects N1. The N1 flight moves counter turns 1.

Slot	ID	Reg.	Moves
9:00–9:02			
9:02–9:04	R3	Yes	0
9:04–9:06	N1	No	1
9:06–9:08			
9:08–9:10			

(c) The priority of N1 is lower than that of R3. However, due to its moves counter, its priority is higher than that of N2. Hence, it ejects N2. The moves counter of N2 turns 1.

Slot	ID	Reg.	Moves
9:00–9:02			
9:02–9:04	R3	Yes	0
9:04–9:06	N1	No	1
9:06–9:08	N2	No	1
9:08–9:10			

(d) The priority of N2 is lower than that of R3 and it is not higher than that of N1. Hence, it gets slotted to 9:06–9:08.

Fig. 3: Example of prioritisation of regulated over non-regulated flights, with ties broken by the number of moves. This scheme enforces fairness, as both non-regulated (low-priority) flights get equally delayed.

2. SUITABLE(2) allows only slots $(t_{\text{from}}, t_{\text{to}}, B)$ such that

$$\max\{t_{\text{CTOT}} - 5,\ t_{\text{TOBT}}(f) + t_{\text{EXOT}}(f)\} \le t_{\text{from}}$$

and

$$\text{either } B = \emptyset \text{ or } \big(B = \{a\} \text{ and } t_{\text{CTOT}}(f(a)) = \text{‘none’}\big).$$

In other words, flight f has ‘allocation priority’ over any non-regulated flight.

With the above rules, the first pass of the algorithm will attempt to allocate f to some vacant slot within the CTOT window. If failed, the second pass will attempt to allocate the flight to the earliest slot whether it is empty or occupied by a non-regulated flight.

Some airports, however, may prefer to have the regulated flights at the runway as early as possible in any circumstances. Then the first rule should be omitted.

For a flight which is not regulated, it may be appropriate to replace the first allocation pass by:

1. SUITABLE(1) allows only slots $(t_{\text{from}}, t_{\text{to}}, B)$ such that

$$t_{\text{from}} \geq t_{\text{TOBT}}(f) + t_{\text{EXOT}}(f) \text{ and } B = \emptyset.$$

   In other words, non-regulated flights can only be allocated to vacant slots.
   This strategy enforces the minimum perturbation of the sequence, see example in Figure 2.

An alternative rule for a regulated flight may be:

1. SUITABLE(1) allows only slots $(t_{\text{from}}, t_{\text{to}}, B)$ such that

$$t_{\text{from}} \geq t_{\text{TOBT}}(f) + t_{\text{EXOT}}(f)$$

   and

$$\text{either } B = \emptyset \text{ or } \big(B = \{a\} \text{ and } t_{\text{CTOT}}(f) = \text{`none'} \text{ and } m(f) > m(f(a))\big).$$

   In other words, the algorithm accepts the first slot which is either vacant or is occupied by a non-regulated flight with a lower number of moves.
   This strategy enforces fairness as the algorithm tends to avoid moving flights that were already moved many times, see example in Figure 3.

The above examples demonstrate that the designed TSAT Generator can be adapted for a wide variety of specific airports and operator preferences, while continuing to use clearly understandable rules, and maintaining the clarity and acceptability for airport operators. Table 1 depicts how each of the requirements defined in Section 2.3 are met by our system.

5 Potential enhancements and future benefits

The system which has been implemented based upon this research has been integrated into a medium sized International European airport, providing both take-off time predictions and pushback time allocations. As inter-airport coordination improves and new facilities come on board, further uses could be found for the predicted take-off times, and further feeds may need to be considered within the allocation rules. Two such potential links are considered in this section: a link with a DMAN and a link with the strategic slot allocation system for airports.

The similarities and differences between a DMAN and PDS system were discussed in Section 2.2. In general, a DMAN will be needed at an airport where the departure sequencing problem is too complex for controllers to easily solve it manually. In small to medium sized mixed-mode airports (where the runways are used for both take-offs and landings) it may be simple enough to manually solve the sequencing problem (see Section 2.1). However, at large multi-runway airports, especially where complex take-off sequencing constraints are present, where there are constraints on some departure routes, or where runways are used in segregated mode (e.g. London Heathrow has all three of these problems, Atkin et al (2007)), sequencing help, such as from a DMAN, may be appropriate. When a PDS is

Produce a predicted take-off time for each aircraft, for use by Eurocontrol and for passing on to other parties (e.g. airports) to aid overall system optimisation.	The TSAT Generator predicts the TTOTs by producing an approximate departure sequence and maintaining it reflecting the flight and runway status updates. The updates are communicated to the other components of the system.
Ensure that all take-off times meet the allocated CTOT timeslots for regulated flights.	The ordering of flights in the allocation queue and the implementation of the SUITABLE function can be adjusted to give priority to regulated flight over non-regulated flights. The sample implementation of the SUITABLE function (see Section 4.4) guarantees that, if an appropriate runway time slot exists, the regulated flight will be allocated within the CTOT window unless all such slots are occupied by other regulated flights.
Ensure that the sequence responds appropriately to changing circumstances.	By comparing the new flight database to its last seen version, Algorithm 1 reliably extracts information on the recent data changes, quickly responding to any events. The runway data changes are handled by regeneration of the runway time slots and further matching of the flight allocations to the new slots.
Ensure that the prioritisation (slot-allocation) is both transparent (using known rules) and justifiable to all parties.	The sequencing procedure is transparent and deterministic. The response of the system to any change of the data is predictable by any of the stakeholders. Moreover, interference between changes is minimised due to the system's attempt to minimise perturbations, which further improves predictability.
Ensure that the system is easily adjustable to the needs and practices of a particular airport and maintainable.	The behaviour of the system can be adjusted by modifying the order and allocation prioritisations, see Section 4.4. The multi-pass algorithm allows flexible control over the preferred slots for each allocation. The declarative nature of all the adjustable components provide the necessary transparency and maintainability.
Ensure reliability of the system in circumstances such as unstable connection with the other components, power cuts off, etc.	The one-way synchronisation mechanism (Section 4.1) employed to communicate data changes between the components effectively tackles unstable connection and power cut off issues and prevents errors accumulation.

Table 1: Ways in which our system meets the requirements.

present, there will usually be fewer aircraft at the runway (the PDS will hold back at the gates those which would otherwise have longer waits), which will increase the practicality of solving the remaining sequencing problem manually, making a full DMAN less necessary.

At airports where both a PDS and DMAN are present, it is possible to link the two. The simplest way to link a PDS and DMAN together is to apply the PDS first, to determine the TSATs and obtain the early predictions of the take-off times. When the aircraft reach the runway a subsequent DMAN can then determine the final take-off sequence. However, it should be noted that the set of aircraft which the DMAN has available to sequence will depend upon the results from the PDS system (which will determine when to release aircraft from the gates). At some airports, where the throughput is not very sensitive to the set of aircraft which are available, an unlinked system would work well. At others, it may be necessary for

the PDS system to actually understand the detailed runway separation rules in order to ensure that an appropriate pool of aircraft is available. For example, the TSAT allocation system at London Heathrow (Atkin et al, 2013) actually models the runway sequencing constraints due to the ease with which capacity can be lost at Heathrow.

When a PDS is considering all of the separation constraints already, there is an obvious question of whether the system could do both tasks. For example, a cut-down version of the DARTS DMAN can be utilised as a PDS (delair air traffic systems, 2013), and the Heathrow TSAT generator (Atkin et al, 2013) could easily be used as a DMAN, since it continues to refine take-off time predictions for aircraft even after they push back (although these changes are not sent externally from the airport) in order to ensure that the changing airport situation is reflected in later TSATs which it allocates. However, the major problem difference between these two systems is the time at which decisions are made. Due to the fact that aircraft may not be able to push back in time (for reasons which are beyond the control of an airline in many cases, e.g. delayed passengers) and the variability in taxi times, it is impractical to actually commit to definitive take-off times or sequences which have been determined by a system at least 30-60 minutes before take-off (the sort of time at which a PDS will be estimating take-off times). Even if a single system performed both tasks, they should happen separately (or on an ongoing basis), so that the take-off sequence and times that are suggested to controllers take account of the situation at the time, rather than an hour earlier when the TSATs were generated.

In addition to the CTOT slots and the allocated runway slots, that have already been discussed in this paper, other types of slots are often allocated at airports. Airports have theoretical maximum capacities for various things (such as runway usage). Unfortunately, the load on an airport is not usually level throughout the day, with significant peaks, and the allocation of 'slots' to airlines indicating an allocation of the resources to the airline and a time at which the resource can be used is common at the busier airports[5]. Runway slots (which will here be called 'airline runway slots' to differentiate them from the runway slot times which the TSAT system allocates) are an example of these. Airline runway slots can have a very high value at some airports[6], and the primary purpose is to ensure that an airline keeps to the times to which they have committed.

Although compliance with these airline runway slots is not currently considered when the TSAT system allocates runway slot times, this could be utilised as an objective, potentially in the same way as CTOT compliance is considered. However, since the TSAT system is under the control of the airport, not the airline, and is attempting to tactically handle, in a fair manner, the situation where the demand for runway capacity exceeds the supply, there are some potential problems with such an approach, in that it may result in a 'less fair' slot allocation. For example, an airline which had been allocated a slot, but was delayed, may benefit by being prioritised in order to meet its airline runway slot; a situation which is far from the initial intention for such allocations.

[5] See IATA slot allocation process page, `http://www.iata.org/policy/slots/Pages/index.aspx`

[6] For example, "Heathrow slots fetch £20m", Sunday times: `http://www.thesundaytimes.co.uk/sto/business/Companies/article1425242.ece`

New systems which would measure the compliance of airlines to allocated slots are being considered[7] could take the predicted take-off times from airport systems (including the TSAT system discussed here) into consideration when examining compliance with airline runway slots, in which case the interaction of the two types of runway slots may need to be reconsidered, and a two-way communication (i.e. feedback) mechanism may also prove useful, subject to the approval of all stakeholders involved.

6 Conclusions

In this paper we have described the aims of a Pre-Departure Sequencer for allocating TSATs and the environment within which it must operate. We have explained the goals, requirements and algorithms which we utilised in the development of this real decision support system. The system has been built to be integrated into the Ultra Electronics systems which are running at a number of airports worldwide, and will enter acceptance trials at the first airport soon. It has been designed in collaboration with experts at the airport and within Ultra Electronics Airport Systems and a number of lessons were learned in the process about how to actually meet user requirements in this kind of environment. The main research conclusions from this process are summarised below.

Besides the obvious sequencing constraints, one of the most important aspects of the PDS is the interaction with humans, for example, with decision makers in the ATC and ground operations. This has the immediate consequence that the system decisions should not 'churn': TSAT values should not be changed more than necessary, since constant updates lead to difficult and inefficient operations. This is an important criterion in the algorithm design.

It was also important that the human aspects required the PDS decisions to be predictable, repeatable, and potentially explainable to people that are experts in ATC, but not experts in algorithms or search. If the PDS is stochastic, then the exact outcome is unpredictable, which can be very disconcerting for operators and also makes the software testing phase both onerous and complex, or impossible to guarantee. The requirement of ensuring that decisions could always be given explanations that make sense to the human experts limited the choice of algorithms that would be appropriate for this problem. For example, a standard stochastic local search would be a last resort, since it tends to be non-repeatable, and also very difficult to explain or justify the final decisions, which effectively arose from making some kind of global improvement, so the identification and justification of specific local improvements can be hard to achieve. As has been shown, the developed algorithm is flexible enough to not only allow decisions to be explained, but to conform to many different decision maker priorities.

Overall, this led us to an event-driven rule-based approach; though with multiple passes through carefully designed sets of rules, and various triggers corresponding to circumstances such as the runway capacity changing. The basic algorithm presented here is based on splitting the runway resource into time slots of the same lengths, which are computed from maximum number of take-offs per hour

[7] see for example, Eurocontrol's Flight Plan and Airport Slot Consistency Service (FAS) web page: `https://www.eurocontrol.int/services/cs1-flight-plan-and-airport-slot-consistency-service-fas`

as provided by the ATC. In general the PDS algorithm will be working with a sequence which it has previously generated, and it then has to fit in a new flight allocation that has just become available, remove one which is no longer available, or make some change to an existing flight allocation. The sequence is updated in reaction to events such as 'a new flight is declared', 'EXOT is changed' or 'controller reallocated a flight'. Such a system avoids unnecessary alteration of flights and has an easy to understand behaviour.

A PDS system can significantly improve many aspects of airport operations, by providing both predicted take-off times and allocating TSATs to airlines well in advance of pushback. Apart from obeying the basic constraints of take-off sequencing, the system keeps the number of changes in the pre-departure sequence to the minimum and has easily predictable and explainable behaviour. The traditional focus of OR optimisation projects is on the problem alone; however, one of our main lessons was that the "meta-problem" of the human context, with the need for development of high trust levels in the autonomous operations, had an important influence on the user acceptability of different algorithms in this research. The lessons learned here, may well also be useful to other problems in OR that have a combination of a dynamic, interactive, online system, with the need to provide a decision support system that can handle the complexity but do so using a rule-based so as to remain explainable and transparent to the end users.

Acknowledgements We would like to thank Ultra Electronics Airport Systems for their collaboration with us in building the automated pre-departure sequencer and for integrating it into their system.

References

Atkin JAD (2008) On-line decision support for take-off runway scheduling at London Heathrow airport. PhD thesis, University of Nottingham

Atkin JAD, Burke EK, Greenwood JS, Reeson D (2007) Hybrid meta-heuristics to aid runway scheduling at London Heathrow airport. Transportation Science 41(1):90–106

Atkin JAD, Burke EK, Greenwood JS, Reeson D (2008) On-line decision support for take-off runway scheduling with uncertain taxi times at London Heathrow airport. The Journal of Scheduling 11(5):323–346

Atkin JAD, Burke EK, Greenwood JS (2010a) The TSAT allocation system at London Heathrow: The relationship between slot compliance, throughput and equity. Public Transport 2(3):173–198

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

Atkin JAD, De Maere G, Burke EK, Greenwood JS (2013) Addressing the push-back time allocation problem at Heathrow airport. Transportation Science 47(4):584–602

Balakrishnan H, Chandran BG (2010) Algorithms for scheduling runway operations under constrained position shifting. Operations Research 58(6):1650–1665

Beasley JE, Krishnamoorthy M, Sharaiha YM, Abramson D (2000) Scheduling aircraft landings – the static case. Transportation Science 34:180–197

Beasley JE, Sonander J, Havelock P (2001) Scheduling aircraft landings at London Heathrow using a population heuristic. Journal of the Operational Research Society 52:483–493

Beasley JE, Krishnamoorthy M, Sharaiha YM, Abramson D (2004) Displacement problem and dynamically scheduling aircraft landings. Journal of the Operational Research Society 55(1):54–64

Bennell J, Mesgarpour M, Potts C (2011) Airport runway scheduling. 4OR: A Quarterly Journal of Operations Research 9:115–138

Bianco L, Dell'Olmo P, Giordani S (1999) Minimizing total completion time subject to release dates and sequence-dependent processing times. Annals of Operations Research 86:393–415

Bolander MA (2000) Scheduling and control strategies for the departure problem in air traffic control. PhD thesis, University of Cincinnati

Dear RG, Sherif YS (1989) The dynamic scheduling of aircraft in high density terminal areas. Microelectronics and Reliability 29(5):743–749

delair air traffic systems (2013) DARTS, arrival and departure management. `http://www.delair.de/darts.html?file=tl_files/delair/pdf/departure%20manager.pdf` (last accessed July 2015).

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

Ernst AT, Krishnamoorthy M, Storer RH (1999) Heuristic and exact algorithms for scheduling aircraft landings. Networks 34:229–241

EUROCONTROL, ACIEurope (2010) A-CDM Airport Collaborative Decision Making. Brochure available at: `http://www.euro-cdm.org/library/cdm_brochure.pdf`

H Balakrishnan BC (2010) Algorithms for scheduling runway operations under constrained position shifting. Operations Research 58:1650–1665

München F (2006) Airport CDM at Munich airport. `http://www.euro-cdm.org/library/eurocontrol/tf12/tf12_cdm_status_munich.pdf` (last accessed July 2015).

Newell G (1979) Airport capacity and delays. Transportation Science 13(3):201–240

Psaraftis HN (1990) A dynamic programming approach for sequencing groups of identical jobs. Operations Research 28:1347–1359

SESAR Joint Undertaking (2004) Basic DMAN operational service and environment definition (OSED). `http://www.sesarju.eu/sites/default/files/solutions/4_DMAN_baseline_OSED.pdf` (last accessed July 2015).

Trivizas DA (1998) Optimal scheduling with maximum position shift (MPS) constraints: A runway scheduling application. Journal of Navigation 51:250–266

Wu C, Caves R (2002) Research review of air traffic management. Transport Reviews 22:115–132