Submitted to *Transportation Science*
manuscript (Please, provide the manuscript number!)

A New Simheuristic Approach for Stochastic Runway Scheduling

Rob Shone
Lancaster University, r.shone@lancaster.ac.uk,

Kevin Glazebrook
Lancaster University, k.glazebrook@lancaster.ac.uk,

Konstantinos G. Zografos
Lancaster University, k.zografos@lancaster.ac.uk,

We consider a stochastic, dynamic runway scheduling problem involving aircraft landings on a single runway. Sequencing decisions are made with knowledge of the estimated arrival times (ETAs) of all aircraft due to arrive at the airport, and these ETAs vary according to continuous-time stochastic processes. Time separations between consecutive runway landings are modeled via sequence-dependent Erlang distributions and are affected by weather conditions, which also evolve continuously over time. The resulting multi-stage optimization problem is intractable using exact methods and we propose a novel simheuristic approach, based on the application of methods analogous to variable neighborhood search (VNS) in a high-dimensional stochastic environment. Our model is calibrated using flight tracking data for over 98,000 arrivals at Heathrow Airport. Results from numerical experiments indicate that our proposed simheuristic algorithm outperforms an alternative based on deterministic forecasts under a wide range of parameter values, with the largest benefits being seen when the underlying stochastic processes become more volatile and also when the on-time requirements of individual flights are given greater weight in the objective function.

Key words: Simulation optimization, stochastic processes, runway scheduling, aviation
History:

1. Introduction

Imbalances between demand and capacity at the world's busiest airports continue to pose problems for schedule coordinators, air traffic controllers, airspace users and other stakeholders. By the end of June 2023, daily flight numbers in Europe had recovered to 93% of their pre-pandemic levels and major hubs such as London Heathrow, Paris Charles de Gaulle and Amsterdam Schiphol were again processing more than 1000 runway movements (i.e. take-offs or landings) per day on average

(Eurocontrol (2023)). Flight delays, however, are also increasing in frequency following the record lows seen during 2020 (Eurocontrol (2022)). The question of how to mitigate air traffic congestion while satisfying the ever-increasing demand for air transport services has been examined extensively at the strategic, tactical and operational levels (Zografos et al. (2017), Jacquillat and Odoni (2018), Cavusoglu and Macario (2021)).

Air traffic delays can be mitigated by improving the efficiency of airport operations, including the sequencing of aircraft using the runways. Optimizing runway sequences is a critical step in making efficient use of scarce airport capacity and minimizing delays in the airport network as a whole. Following the developments of research programmes such as SESAR and NextGen in recent years, decision support tools such as the Arrival Manager (AMAN) are now being extended in order to allow the management of incoming flights at much earlier points in their journeys, in order to absorb delays in the 'enroute' stage and reduce the need for holding patterns close to the airport (SESAR (2023a,b)). Thus, there is a growing need to consider sequencing decisions over longer time horizons in order to make the best use of available technology. However, this task is made more challenging by the fact that an aircraft's estimated arrival time (ETA) is subject to greater uncertainty when the prediction is made at an earlier stage of its journey (Khassiba et al. (2020)).

In this paper we consider a stochastic, dynamic runway scheduling problem. The focus is on tactical decision-making[1], in the sense that we aim to optimize the real-time decisions made by air traffic controllers in response to the latest available information on a particular day of operations, with the schedule of aircraft take-offs and landings having been determined in advance. Our problem formulation draws upon different areas of the literature that have evolved along quite separate lines in the last few decades. On one hand, we model the stochastic nature of airport runway operations using a queueing theory approach, whereby the times that aircraft arrive at the runway threshold and complete 'service' (i.e. usage of the runway) are subject to uncertainty. Several previous studies have used stochastic queueing formulations to model operational delays at airports (Koopman (1972), Stamatopoulos et al. (2004), Hansen et al. (2009), Pyrgiotis and Odoni (2016), Jacquillat et al. (2017)). The decision-making aspect of our problem relates to the sequencing of aircraft landing at a single runway during a congested period. In this respect, we build upon the literature on aircraft sequencing problems, which began with the formulation of static, deterministic optimization problems (Psaraftis (1978)) but has more recently expanded into the area of stochastic optimization (Solveling et al. (2011), Heidt et al. (2016), Liu et al. (2018), Solak et al. (2018), Khassiba et al. (2020)).

In queueing models of airport runway operations, a common approach is to use an estimate for the airport's capacity in order to derive 'service rates' for the queues. The capacity of an airport

[1] The term 'operational decision-making' may be preferred by some authors.

or single runway may be defined as the expected number of runway movements per unit time that can be operated under conditions of continuous demand (de Neufville and Odoni (2013)). This definition implies that an airport's capacity actually varies with time, as it depends on various controllable and non-controllable factors, including runway configurations and weather conditions. Gilbo (1993) introduced the concept of a 'capacity envelope' to represent the set of feasible pairs of service rates for arrivals and departures at a single airport, and subsequently this approach has been used to formulate stochastic, dynamic optimization problems based on the control of service rates at discrete time epochs (Jacquillat and Odoni (2015), Jacquillat et al. (2017), Shone et al. (2019)). The use of a capacity envelope to select service rates for aircraft queues may be seen as somewhat macroscopic in nature, as it does not explicitly allow for fine-grain aircraft sequencing and the time savings that air traffic controllers can achieve by taking into account separation requirements between different types of aircraft; instead, it assumes that different possible traffic mixes and other considerations can be implicitly accounted for by configuring service time variances (Shone et al. (2021)). The model proposed in this paper assumes that an aircraft's service time follows a random distribution which depends on its own weight class and also that of its immediate predecessor in the runway queue. Thus, we allow for the effects of weight classes on separation times, while preserving the stochastic modeling of runway service times.

Two-stage stochastic optimization has become popular in recent years as a means of incorporating uncertainty into runway scheduling problems. Solveling et al. (2011) introduced a two-stage model in which a sequence of aircraft weight classes is determined in the first stage, and specific flights are assigned to positions in the sequence (subject to weight class compatibility) in the second stage. Solak et al. (2018) subsequently enhanced this model by introducing costs based on exact timings of runway operations. Liu et al. (2020) used a similar formulation, but allowed for limited or ambiguous information about the model parameters. Khassiba et al. (2020) considered the optimization of a sequence of aircraft arriving at an initial approach fix (IAF) and used chance constraints to mitigate the risk of separation time violations.

While two-stage optimization methods clearly have useful applications in runway scheduling problems, in this paper we take a different approach in order to model the problem faced by air traffic controllers who need to make decisions in information-rich and fast-changing dynamic environments. We consider multi-stage problems involving hundreds of aircraft and use continuous-time stochastic processes to model the evolution of uncertainty over time, with each decision being made under the latest set of information available. The high-dimensional nature of our problem precludes the use of exact solution methods, and instead we opt for a simulation-based approach in which many possible runway sequences are compared using randomly-sampled system trajectories. Our approach enables the continuous availability of a recommended runway sequence that air traffic controllers can use to direct the movements of aircraft near the terminal airspace; please see Section 3.4 for further explanation of how our methods can be implemented in practice.

The main contributions of our paper are as follows:

- We provide a new formulation for the multi-objective stochastic runway scheduling problem which includes three different types of dynamic uncertainty: (i) estimated times of arrival (ETAs) for aircraft evolve according to continuous-time stochastic processes; (ii) sequence-dependent aircraft separation times follow Erlang distributions; (iii) expected changepoints in weather conditions also vary according to continuous-time stochastic processes.
- We propose a novel solution methodology for this problem, based on the application of simheuristic search techniques in a stochastic and rapidly-changing environment.
- We configure our model using data from over 98,000 flights landing at Heathrow Airport in 2018 and 2019 and show, using numerical experiments, that the solutions given by our simheuristic approach consistently outperform those given by alternative heuristics with respect to a weighted objective function based on schedule punctuality and air-holding times.

Our model formulation is provided in Section 2. Details of our heuristic approaches are presented in Section 3. The use of flight tracking data to configure our model parameters is described in Section 4, and we provide details of results from our computational experiments in Section 5. Our concluding remarks are given in Section 6.

2. Model formulation

In this paper we restrict attention to arriving flights (landings) on a single runway at an airport, and assume that this runway is not used or affected by the stream of outbound traffic (departures). It is common for large airports to use one or more runways for arrivals only; for example, Heathrow Airport operates two runways in 'segregated mode' during typical hours of operation. Let $\mathcal{F}$ be the set of arriving flights scheduled to use the runway during a particular time interval, denoted by $\mathcal{T} := [0,T]$. For each flight $i \in \mathcal{F}$ we associate a scheduled arrival time $a_i \in \mathcal{T}$ at the destination airport and a scheduled departure time $d_i \in (-\infty, a_i)$ from its origin airport. For clarity, we emphasize that the destination airport is the same for all flights in $\mathcal{F}$, but the origin airport is flight-specific. We also use w_i to denote the weight class of aircraft $i \in \mathcal{F}$, belonging to a set of weight classes $\mathcal{W}$, and define $g_i \in [0,1]$ as a relative cost parameter associated with delays to flight i, which might depend on the number of passengers carried and other factors.

The actual times that flights land on the runway are affected not only by sequencing and scheduling decisions, but also by the uncertainty affecting (i) their departure and flight times, (ii) landing time separations with preceding aircraft and (iii) weather conditions at the destination airport. In Sections 2.1-2.3 we explain how these different sources of uncertainty are modeled, and in Section 2.4 we present our decision-making framework and objective function.

2.1. Unconstrained landing times

The first source of uncertainty in our model is related to the earliest time that a flight would be able to land in the absence of congestion effects or adverse weather conditions at the destination airport. Following previous studies (Bennell et al. (2017), Khassiba et al. (2020)), we refer to this as the *unconstrained landing time* and denote it by A_i for flight $i \in \mathcal{F}$. Noting that A_i may depend on many unpredictable factors and control interventions at various different stages of flight i's progress (including the pre-departure stage), we propose to make a distinction between *pre-tactical uncertainty* and *tactical uncertainty* and write

$$A_i = a_i + \Delta_i^{\text{pre}} + \Delta_i^{\text{tac}}, \tag{1}$$

where Δ_i^{pre} and Δ_i^{tac} are pre-tactical and tactical delays (possibly negative-valued), respectively. Here, 'pre-tactical' delays are those which can already be foreseen well in advance of a flight's departure, but would not have been known when the airport arrival schedule was originally produced (typically several months in advance of operations). Examples of delays which may fall into the 'pre-tactical' category are those caused by airline crew unavailability, local airspace restrictions or global upper wind conditions. On the other hand, 'tactical' delays are those which evolve dynamically during actual operations. These might include take-off delays caused by taxiway congestion or enroute delays caused by the need to avoid potential air traffic conflicts.

The pre-tactical delay Δ_i^{pre} is a semi-bounded continuous random variable in our model, while Δ_i^{tac} is also stochastic but additionally depends on sequencing decisions. In our numerical experiments later, we rely on gamma distributions for modeling the pre-tactical delays, with flight-specific parameters estimated from historical data; further details are given in Section 4. For each $i \in \mathcal{F}$, we assume that Δ_i^{pre} is already 'realized' in advance of flight i's scheduled departure time and remains constant throughout the remainder of $\mathcal{T}$. On the other hand, Δ_i^{tac} is not known until flight i actually lands, and until then we are only able to predict its value.

To model tactical uncertainty we use $X_i(t)$ to denote the *estimated time of arrival* (ETA) for flight i at time $t \in \mathcal{T}$, which depends on the latest information available at t. The ETA $X_i(t)$ may vary considerably in short intervals of time and is generally not a monotonic function of t. We assume $X_i(t)$ remains equal to $a_i + \Delta_i^{\text{pre}}$ (an adjusted ETA after realization of the pre-tactical delay Δ_i^{pre}) until t is within a certain proximity q_i of the scheduled departure time d_i, at which point tactical uncertainty (which may include some pre-departure uncertainty associated with taxi-out times, for example) begins to take effect and $X_i(t)$ varies according to a Brownian motion (BM) process. For convenience, let $h_i = d_i - q_i$. Then we define

$$X_i(t) = \begin{cases} a_i + \Delta_i^{\text{pre}}, & \text{if } t \le h_i, \\ a_i + \Delta_i^{\text{pre}} + B_i(t) + \rho_i(t), & \text{if } t > h_i, \end{cases} \tag{2}$$

where $B_i(t) \sim \mathrm{N}(0,\, \sigma_i^2(t-h_i))$ and $\rho_i(t) \geq 0$ denotes any additional 'airborne holding time' incurred by flight i as part of the tactical decision-making process, i.e. the aircraft sequencing. We elaborate further on this holding time in Section 2.4, but in the remainder of this subsection we assume (for ease of exposition) that $\rho_i(t) \equiv 0$. The unconstrained landing time A_i is given by

$$A_i = \min\Big\{t > h_i \,:\, X_i(t) \leq t\Big\}. \tag{3}$$

That is, A_i is the earliest point at which $X_i(t)$ is exceeded by the current time. Figure 1 shows how A_i and Δ_i^{tac} are determined by the BM trajectory for a particular flight i, given some parameters a_i, d_i, q_i and a fixed realization of Δ_i^{pre}. Note that we have the logical property that as t increases, $X_i(t)$ becomes an increasingly accurate forecast of A_i; in other words, the unconstrained landing time becomes more predictable with time.

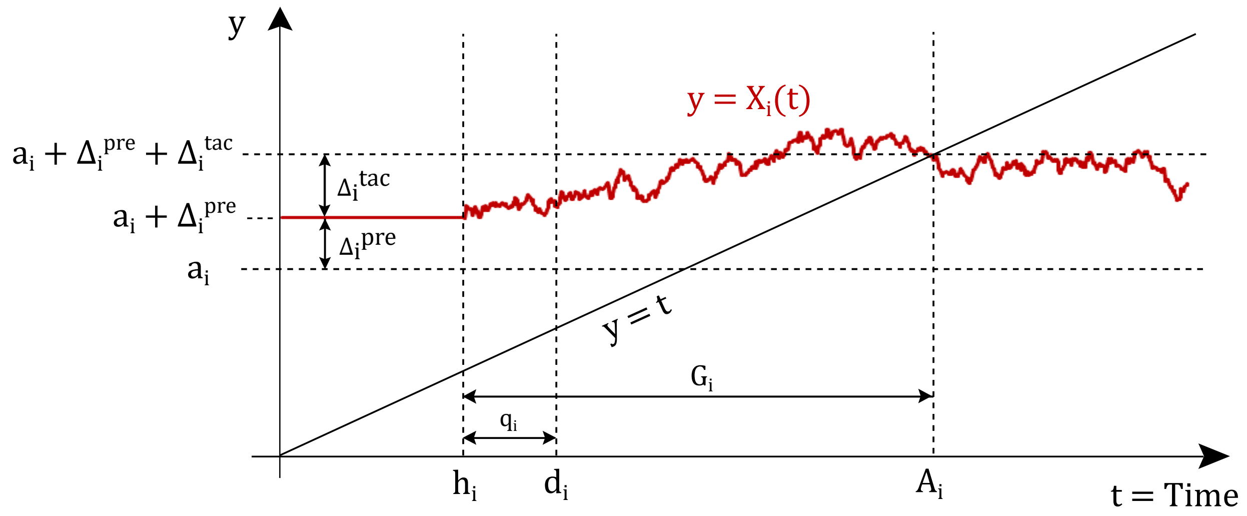

Figure 1 **The unconstrained landing time A_i is obtained as the earliest point at which the trajectory $y = X_i(t)$, which behaves randomly for $t > h_i$, intersects the straight line $y = t$. The pre-tactical and tactical delays, Δ_i^{pre} and Δ_i^{tac}, are shown as distances on the vertical scale.**

From (3), it follows that

$$A_i \overset{\mathrm{dist}}{=} \min\Big\{t > h_i \,:\, B_i(t) + t \geq a_i + \Delta_i^{\mathrm{pre}}\Big\} \tag{4}$$

$$= h_i + \min\Big\{u > 0 \,:\, B_i(u + h_i) + u \geq a_i + \Delta_i^{\mathrm{pre}} - h_i\Big\} \tag{5}$$

$$\overset{\mathrm{def}}{=} h_i + G_i, \tag{6}$$

where we have used the fact that $-B_i(t)$ has the same distribution as $B_i(t)$ in (4), and the change of variables $u = t - h_i$ is used in (5). Noting that $B_i(u + h_i) \sim \mathrm{N}(0,\, \sigma_i^2 u)$, we see that A_i can be interpreted as the 'first hitting time' for a BM process with a positive linear drift (represented by the additive term u in (5)). Using known theoretical results for BM processes with drift (Folks and Chhikara (1978)), it follows that the conditional density function of the random variable G_i defined in (6), given some realization of Δ_i^{pre}, is the inverse Gaussian function

$$f_{G_i}(u \mid \Delta_i^{\text{pre}} = \delta) = \frac{a_i + \delta - h_i}{\sigma_i \sqrt{2\pi u^3}} \exp\left(-\frac{(a_i + \delta - h_i - u)^2}{2\sigma_i^2 u}\right) \quad (u > 0), \tag{7}$$

which has mean $a_i + \delta - h_i$ and variance $\sigma_i^2(a_i + \delta - h_i)$. The tactical delay Δ_i^{tac} can be expressed as

$$\Delta_i^{\text{tac}} = G_i - (a_i + \Delta_i^{\text{pre}} - h_i),$$

which can be recognized from Figure 1 as the difference between the duration of the Brownian motion, $A_i - h_i$, and the duration that would occur if $X_i(t)$ remained at its initial level $a_i + \Delta_i^{\text{pre}}$ throughout $\mathcal{T}$. Naturally, this has a mean of zero. We also note, using (1), that

$$\mathbb{E}[A_i] = a_i + \mathbb{E}[\Delta_i^{\text{pre}}] \tag{8}$$

and, using the law of total variance,

$$\begin{aligned} \text{Var}(A_i) &= \text{Var}(G_i) \\ &= \text{Var}(\mathbb{E}[G_i \mid \Delta_i^{\text{pre}}]) + \mathbb{E}[\text{Var}(G_i \mid \Delta_i^{\text{pre}})] \\ &= \text{Var}(\Delta_i^{\text{pre}}) + \sigma_i^2 \left(a_i + \mathbb{E}[\Delta_i^{\text{pre}}] - h_i\right). \end{aligned} \tag{9}$$

While it is possible to obtain data on the timeliness of aircraft arrivals and departures at airports around the world, it is not easy to find reliable information about how an aircraft's ETA varies dynamically during (and prior to) its flight. Hence, parameters such as σ_i in our model are difficult to estimate accurately. In our numerical experiments in Section 5, we consider a range of possible values of σ_i, but for each value of σ_i we adjust the parameters of the pre-tactical delay distribution in such a way that the value of $\text{Var}(A_i)$ given by (9) corresponds as closely as possible to an estimate of $\text{Var}(A_i)$ obtained from historical data. In other words, we keep $\text{Var}(A_i)$ close to a fixed, data-calibrated value, but experiment with different 'weightings' for the pre-tactical and tactical components of the variance in (9).

2.2. Landing separation times

In our model, the actual time that flight $i \in \mathcal{F}$ lands on the runway depends not only on the unconstrained landing time A_i but also on the traffic congestion and weather conditions at the destination airport. In reality, pairs of aircraft landing consecutively on the same runway are required to maintain certain time separations, and these separations depend on the weight classes of the aircraft involved. Many previous studies have incorporated class-dependent separation requirements in mathematical formulations of aircraft sequencing problems, although these problems are often of a deterministic nature (Psaraftis (1978), Beasley et al. (2000, 2004), Bennell et al. (2017)).

Congestion in airport terminal areas naturally causes 'queues' to form as aircraft await their turn to use the runway. There is a long-established tradition of using time-dependent queueing models such as $M(t)/E_k(t)/1$ to model on-time performance at airports (Koopman (1972), Stamatopoulos et al. (2004), Hansen et al. (2009), Pyrgiotis and Odoni (2016), Jacquillat et al. (2017)). Here, E_k denotes an Erlang-k distribution, which is a form of gamma distribution. This type of distribution is often favored in queueing models of air traffic due to its convenience and configurability; indeed, the parameters of such distributions can be adjusted in order to ensure close resemblance to empirical distributions that might be observed in practice.

To the best of our knowledge, there is no precedent in the literature for the incorporation of class-dependent, Erlang-distributed separation times in an aircraft sequencing or runway scheduling problem. We propose that such an approach makes sense, given that both class-dependent separation times and Erlang queue service times have (separately) been common features in previous air traffic models. Let us define L_i as the actual time that flight $i \in \mathcal{F}$ touches down on the runway (it is implied that $L_i \geq A_i$). Suppose that flight $j \in \mathcal{F}$ immediately follows i in the landing sequence, and let e_{ij} denote the recommended time separation between a leading aircraft of class $w_i \in \mathcal{W}$ and a following aircraft of type $w_j \in \mathcal{W}$. Then the random variable M_{ij}, interpreted as the time between the landings of i and j if these landings occur during a 'congested period', is assumed to be Erlang-distributed with the probability density function

$$f_{M_{ij}}(t) = \frac{k\mu_{ij}(k\mu_{ij}t)^{k-1}\exp(-k\mu_{ij}t)}{(k-1)!} \quad (t > 0),$$

where $\mu_{ij} = e_{ij}^{-1}$ and k is an integer-valued shape parameter. The mean of this distribution is e_{ij} and the variance is e_{ij}^2/k; hence, large values of k result in time separations that conform closely to the recommended values. For clarity, we emphasize that our model allows the possibility of actual time separations being smaller than their recommended values (due to random variation), although large k values would result in only small deviations. In our numerical experiments in Section 5, we experiment with different values of k but choose e_{ij} according to actual air traffic regulations.

The actual landing time of flight j is then given by

$$L_j = \begin{cases} A_j, & \text{if } j \text{ is the first plane to land during } \mathcal{T}, \\ \max\{A_j,\, L_i + M_{ij}\}, & \text{if } j \text{ is preceded by } i \text{ in the landing sequence.} \end{cases}$$

That is, flight j cannot land before its unconstrained landing time A_j, but also must be appropriately separated from the preceding aircraft i. If flight j happens to arrive in the terminal area much later than flight i's landing time then the random separation M_{ij} becomes irrelevant. This is why we refer to M_{ij} as the time between landings during a 'congested period'.

In fact, the required separation e_{ij} between two consecutively-arriving flights i and j may also have a time dependence in our model due to the effects of weather conditions in the terminal area at the time of flight j's final approach. We have thus far written e_{ij} (without any time dependence) in order to simplify the notation, but this should be modified if weather variations are to be included. Further details are given in Sections 2.3 and 2.4.

2.3. Weather conditions in the terminal area

Poor weather conditions at an airport can reduce its operational capacity by enforcing longer time separations between consecutive take-offs and landings (Odoni et al. (2011)). In our model, it makes sense to consider a time horizon $\mathcal{T}$ that does not span more than one day, and we may therefore assume that any periods of bad weather (and their timings) can be predicted with a high level of precision at the beginning of $\mathcal{T}$. However, there may still be some uncertainty in the forecasts. We consider two possible cases: (i) conditions are certain to remain 'fine' throughout $\mathcal{T}$; (ii) conditions will be fine except for a single period of bad weather, denoted $\mathcal{U} \subset \mathcal{T}$, during which longer time separations are required. (To use the correct terminology, we note that 'fine weather' implies visual meteorological conditions (VMC), whereas bad weather implies instrumental meteorological conditions (IMC); see Jacquillat and Odoni (2015) for further details.) In the first case, the time separations e_{ij} referred to in Section 2.2 do not require any time dependence. In the second case, however, we use R_j to denote the time that flight $j \in \mathcal{F}$ begins the final stage of its journey to the runway and define the required separation between $i, j \in \mathcal{F}$ by

$$E_{ij}(R_j) = \begin{cases} e_{ij}, & \text{if } R_j \in \mathcal{T} \setminus \mathcal{U}, \\ \phi e_{ij}, & \text{if } R_j \in \mathcal{U}, \end{cases} \tag{10}$$

where $\phi > 1$ and e_{ij} is interpreted as the 'fine weather' value. Then, similarly to before, M_{ij} is Erlang-distributed with parameters $\mu_{ij}(R_j) = 1/E_{ij}(R_j)$ and $k \in \mathbb{N}$.

The beginning and ending times of $\mathcal{U}$ are subject to dynamic uncertainty. Suppose that, at the beginning of $\mathcal{T}$, we expect that $\mathcal{U}$ will begin at time t_0 and end at some later time t_1 ($t_0, t_1 \in \mathcal{T}$). However, during $\mathcal{T}$ we continuously revise these estimates according to the latest forecast. Let $T_0(t)$ and $T_1(t)$ be defined by Brownian motion trajectories as follows:

$$T_0(t) \sim \mathrm{N}(t_0, \nu^2 t), \quad T_1(t) \sim \mathrm{N}(t_1, \nu^2 t),$$

where $\nu > 0$ is a variance parameter. We will assume independence between $T_0(t)$ and $T_1(t)$ for simplicity, although a dependence structure could be incorporated if needed. Next, we define

$$U_0 = \min\{t > 0 \mid T_0(t) \le t\}, \quad U_1 = \min\{t > 0 \mid T_1(t) \le t\},$$

and the interval $\mathcal{U}$ is then given by

$$\mathcal{U} = \begin{cases} [U_0, U_1], & \text{if } U_0 < U_1, \\ \emptyset, & \text{otherwise.} \end{cases}$$

At any time t satisfying $t < \min\{U_0, U_1\}$, our prediction is that $\mathcal{U}$ will begin at $T_0(t)$ and end at $T_1(t)$, assuming that $T_0(t) < T_1(t)$. If $T_0(t) \geq T_1(t)$, then the prediction is that no period of bad weather will occur. Our approach can also be generalized to incorporate multiple periods of bad weather; see Appendix G for further explanation and experimental results.

2.4. Decision-making framework and objective function

In this subsection we describe how our stochastic runway scheduling problem can be formulated as a stochastic, dynamic optimization problem by providing details of the state space, action space, cost mechanism and objective function. First, however, we must provide some extra information about the statuses of flights in $\mathcal{F}$ and how these change during $\mathcal{T}$.

As explained in Section 2.1, we assume that the pre-tactical delays Δ_i^{pre} are known for all $i \in \mathcal{F}$ at the beginning of $\mathcal{T} = [0, T]$. We scale the units of time in such a way that $0 < a_i + \Delta_i^{\text{pre}} < T$ for all $i \in \mathcal{F}$; that is, all flights are expected to land during $\mathcal{T}$ following the realization of pre-tactical uncertainty. It is possible that $h_i < 0$ for some $i \in \mathcal{F}$, indicating that $X_i(t)$ is already varying according to Brownian motion and flight i may (or may not) be airborne at $t = 0$. For any such flights, we generate an initial ETA $X_i(0)$ by sampling from a Normal distribution with mean $a_i + \Delta_i^{\text{pre}}$ and variance $\sigma_i^2 |h_i|$ in order to be consistent with (2).

For flight $i \in \mathcal{F}$, let Q_i be defined by

$$Q_i = \min\Big\{ t > 0 \mid X_i(t) - \tau \leq t \Big\},$$

where $\tau > 0$ is a fixed parameter that defines the stage of an aircraft's journey at which it becomes eligible for tactical sequencing; in other words, sequencing decisions are made for aircraft that are within τ time units of their expected unconstrained landing time. In practice, $\tau = 30$ minutes might be an appropriate value (Bennell et al. (2017), Khassiba et al. (2020)). At time Q_i, flight i is said to enter a 'pool' of aircraft waiting to be sequenced, i.e. given a position in the runway landing order. The 'pool' does not represent any particular physical location or region; instead, it merely represents a collection of aircraft that are expected to be able to land within τ time units. In particular, it should not be confused with a holding stack, as planes in a holding stack must generally exit from the stack in a fixed order, whereas the 'pool' in our model represents the last stage of an aircraft's journey at which its sequence position remains undetermined.

The sequencing decision for flight i does not necessarily need to be made at time Q_i. Instead, flight i can be retained in the pool until some later time $R_i > Q_i$, at which point it is 'released' and proceeds with the final stage of its journey. During the time spent in the pool, the flight's progress is effectively 'paused' and its ETA increases linearly, i.e. $X_i(t) = X_i(Q_i) + t - Q_i = t + \tau$ for $t \in [Q_i, R_i]$, indicating that it is not making further progress towards the runway during this time. It may be performing path-stretching maneuvers, for example, in order to elongate its journey (Newell (1979), Montlaur and Delgado (2017)). We define $\rho_i(t) = \min\{t, R_i\} - Q_i$ for $t > Q_i$ as the 'holding time' incurred up to time t, satisfying equation (2). It will also be convenient to define $\rho_i := \lim_{t\to\infty} \rho_i(t) = R_i - Q_i$ as the total amount of time spent in the pool.

At time R_i, flight i enters a 'queue' of aircraft waiting to use the runway. The queue is strictly first-come-first-served, so that aircraft are required to land in the same order that they are released from the pool. During periods of heavy congestion, the queue is likely to include planes circling in a holding stack, but planes on their final descent towards the runway are also notionally part of the 'queue' in our model. We note that

$$A_i = \min\left\{t > R_i \mid t \geq R_i + \tau + \tilde{B}_i(t)\right\}, \tag{11}$$

where $\tilde{B}_i(t) \sim \mathrm{N}(0, \sigma_i^2(t - R_i))$. This ensures consistency with the details in Section 2.1. The actual landing time L_i also depends on the required time separation between flight i and its predecessor in the queue, as explained in Section 2.2, and this separation depends on the weather conditions at time R_i as described in Section 2.3. Figure 2 illustrates our model by showing aircraft at different stages of their journeys at some arbitrary point in time $t \in \mathcal{T}$.

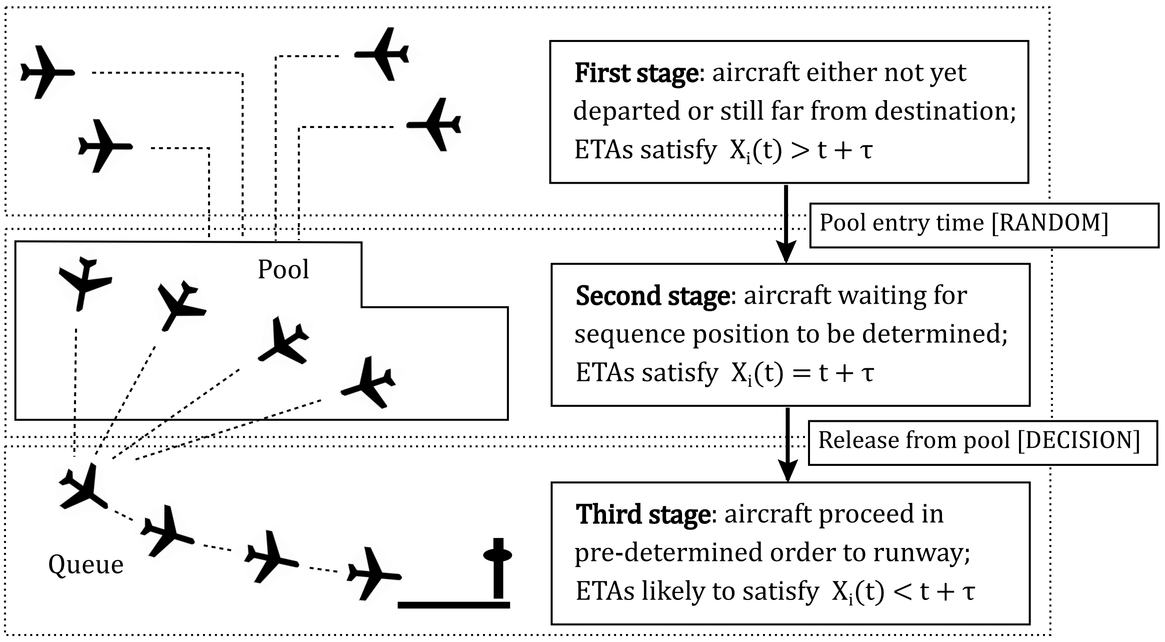
First stage: aircraft either not yet departed or still far from destination; ETAs satisfy $X_i(t) > t + \tau$
Pool entry time [RANDOM]
Pool
Second stage: aircraft waiting for sequence position to be determined; ETAs satisfy $X_i(t) = t + \tau$
Release from pool [DECISION]
Queue
Third stage: aircraft proceed in pre-determined order to runway; ETAs likely to satisfy $X_i(t) < t + \tau$

Figure 2 **Aircraft at different stages of their journeys at an arbitrary point in time $t \in \mathcal{T}$.**

We note that, due to the variable nature of the BM trajectories, it is possible for a flight's expected remaining flight time to exceed τ even if it has already entered the pool at an earlier time and has also been released from the pool; that is, we may have $X_i(t) > t + \tau$ for some $i \in \mathcal{F}$ even if $t > R_i$. In this situation, we assume that flight i's position in the queue (and, therefore, its position in the landing sequence) remains unaffected. It is not required to enter the pool again. Furthermore, any flights behind it in the queue must wait until it has landed before landing themselves, even if they have earlier unconstrained landing times.

The essence of our decision-making problem is to decide the release times R_i for flights $i \in \mathcal{F}$. At any time $t \in \mathcal{T}$, we are aware of the following dynamic information:

- The latest ETAs $X_i(t)$ of all flights $i \in \mathcal{F}$ that have not yet landed, including those that are in the queue, those that are in the pool and those that are yet to arrive in the pool;
- The ordering of any planes that are in the queue;
- Any weather forecast information that remains relevant at time t, which may be summarized by $\{T_0(t), T_1(t)\}$ if $t < U_0$ and by $T_1(t)$ if $U_0 < t < U_1$;
- If a flight j is currently 'in service', meaning that $t \geq A_j$ but $t < L_i + M_{ij}$ (where i is the predecessor of j in the queue), then we are aware of how much time j has spent in service so far.

The above information comprises the *system state* in our problem. We are also aware of the weight classes w_i and cost parameters g_i for all $i \in \mathcal{F}$. Actions can be taken at any time $t \in \mathcal{T}$ for which the pool is non-empty, implying that the set of decision epochs is uncountable; in other words, the decision-maker is able to continuously update their policy according to the latest available information, which also evolves continuously. The 'action' chosen at decision epoch t is an ordered list (i.e. a tuple) of aircraft to release from the pool. Passive actions (empty tuples) are likely to be chosen in cases where it may be advantageous to wait before deciding which aircraft to release next. As an example, suppose the set of flights in the pool at time t is $\{2, 5, 6, 7, 9\}$, where these flights have been indexed according to their positions in the original landing schedule. Then possible actions at time t include $(6, 9, 2, 7, 5)$, $(6, 7)$, (2) and $\emptyset$.

It remains for us to specify our cost mechanism and objective function. We suppose that, in making decisions about when to release aircraft from the pool (and thereby determining the runway landing sequence), we have two broad objectives in mind: (i) flights should be able to land as near as possible to their scheduled landing times; (ii) flights should have their total flight times minimized. These two objectives are not entirely unrelated, but they are quite different in nature. The first objective is related to punctuality of air transport operations and the need to avoid disruption to airline schedules (taking into account the transfer times needed between aircraft flight legs, etc.). The second objective is based on the need to avoid overly long flight times, which would compromise safety and increase fuel costs. For convenience, we tend to refer to the first and second objectives as (minimization of) *schedule delays* and *operational delays* respectively in later sections.

Specifically, for flight $i \in \mathcal{F}$, we define

$$C_i^{[S]} = \left(\left[L_i - a_i - \gamma^{[S]} \right]^+ \right)^2, \quad C_i^{[W]} = \left(\left[L_i - (A_i - \rho_i) - \gamma^{[W]} \right]^+ \right)^2, \tag{12}$$

where $\gamma^{[S]} \geq 0$ and $\gamma^{[W]} \geq 0$ are parameters given as model inputs. The interpretation is that, in the first objective, a delay of up to $\gamma^{[S]}$ time units between the actual landing time and the scheduled landing time is 'acceptable', but beyond this we incur a penalty which increases quadratically with the amount of delay. To interpret the second objective, note that $A_i - \rho_i$ is the time that flight i would land on the runway if it were released from the pool immediately and did not incur any queueing delays. We calculate the amount of 'extra airborne time' incurred as the difference between the actual landing time, L_i, and $A_i - \rho_i$. This 'extra airborne time' can be incurred as a result of being held in the pool *or* being delayed in the queue, so it is not necessarily the case that keeping flight i waiting in the pool leads to an increase in $C_i^{[W]}$.

An advantage of our simulation-based solution approach is that the cost functions $C_i^{[S]}$ and $C_i^{[W]}$ can be made almost arbitrarily complicated without affecting tractability; they do not necessarily need to take quadratic or even polynomial forms, for example. By considering only the positive part of $L_i - a_i - \gamma^{[S]}$ we avoid penalizing early landing times, but penalties for earliness could easily be incorporated into our model if needed. We also note that there is an obvious choice available for $\gamma^{[S]}$, as 15 minutes is often regarded as a threshold for the purpose of classifying delays in the aviation industry (Ball et al. (2010)).

The objective of the problem is to minimize the weighted objective function

$$\sum_{i \in \mathcal{F}} g_i \times \left[\theta^{[S]} C_i^{[S]} + \theta^{[W]} C_i^{[W]} \right], \tag{13}$$

where $\theta^{[S]}$ and $\theta^{[W]}$ are positive-valued weights, normalized so that $\theta^{[S]} + \theta^{[W]} = 1$. Traditionally one would approach a stochastic, dynamic optimization problem by aiming to find an optimal 'policy', mapping states to actions. In this case we have a problem with a high-dimensional, continuous state space which is obviously beyond the scope of exact solution by approaches such as dynamic programming. We propose to take a simheuristic approach and use simulation methods to update our 'belief' of the optimal landing sequence as time and uncertainty evolve.

3. Solution methodology

The main solution approach of interest in our study is a simheuristic approach[2], implemented in real time, in which we continuously update performance estimates for a small number of 'candidate'

[2] See Juan et al. (2015) for a useful review of simheuristic methods for stochastic optimization problems.

solutions (sequences), aiming to discover new strongly-performing solutions and discard weaker ones as time progresses. Within the taxonomy of metaheuristic algorithms, our approach may be compared to a Variable Neighborhood Search (VNS), as we occasionally force the algorithm to migrate to a different region of the solution space if it seems to be making no further progress in finding improved solutions in its current neighborhood.

We summarize our simheuristic algorithm in Section 3.1. In Section 3.2 we describe an alternative, simpler approach, based on considering expected values of random variables rather than using simulation. In Section 3.3 we suggest some additional policies, including the simple 'first-come-first-served' rule, that can be used as benchmarks for our other heuristics. In Section 3.4 we provide further explanations as to how our methods can be implemented in practice.

3.1. The simheuristic approach

Figure 3 shows an outline of the main steps in our simheuristic algorithm, which we refer to as 'SimHeur' for convenience. In the remainder of this subsection we provide an overview of these key steps. Full details of the implementation can be found in Appendix A.

In step 1 we set $t=0$ and create an initial population $\mathcal{S}_0$ of S sequences, with each sequence consisting of up to l flights ($S, l \in \mathbb{N}$) that have yet to be released from the pool (that is, they are either in the pool or yet to arrive in the pool). Each sequence represents a possible selection and ordering of the next l flights to land on the runway. The parameter l is generally much smaller than $|\mathcal{F}|$ (we might consider $l=15$, for example), so we initially consider sequencing decisions over a limited time horizon. The sequences in the population are frequently changed while the algorithm is in progress (see steps 2B, 4A-4C).

In step 2A we update performance estimates of all sequences in the current population by generating random sample trajectories, with each trajectory consisting of realizations of future random events (e.g. pool arrival times, service times, weather changes) obtained by sampling from the relevant probability distributions described in Section 2. We revisit this step many times during the algorithm and generate many random trajectories, with the n^{th} trajectory (for $n=1,2,...$) being associated with a set of cost estimates $\{J_s^{(n)}\}_{s\in\mathcal{S}}$ for sequences $s\in\mathcal{S}$ (where $\mathcal{S}$ is the current population). These estimates $J_s^{(n)}$ are used to update the performance measures

$$V_s^{(n)} := (1-\psi_n)V_s^{(n-1)} + \psi_n J_s^{(n)}, \tag{14}$$

$$W_s^{(n)} := (1-\psi_n)W_s^{(n-1)} + \psi_n \left(J_s^{(n)}\right)^2, \tag{15}$$

where the step size ψ_n may be chosen based on a simple averaging rule (i.e. $\psi_n = 1/n$) or by employing a reinforcement learning-style rule in which the more recently-acquired cost estimates are given greater weight than the older ones, to reflect the fact that they are obtained using

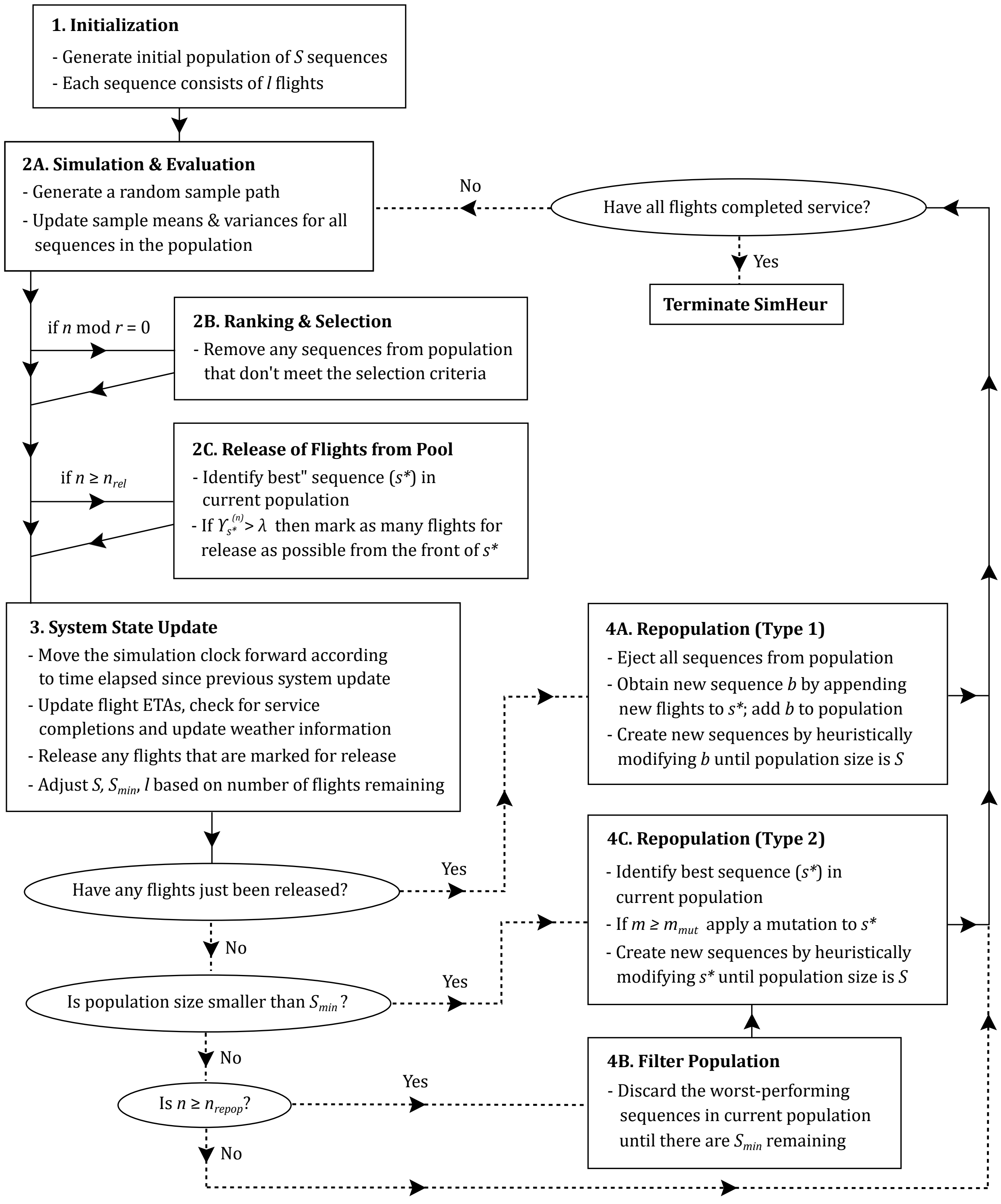
1. Initialization
- Generate initial population of S sequences
- Each sequence consists of l flights
2A. Simulation & Evaluation
- Generate a random sample path
- Update sample means & variances for all sequences in the population
No
Have all flights completed service?
Yes
Terminate SimHeur
if n mod r = 0
2B. Ranking & Selection
- Remove any sequences from population that don't meet the selection criteria
if n ≥ n_rel
2C. Release of Flights from Pool
- Identify best" sequence (s*) in current population
- If Y_s*^(n) > λ then mark as many flights for release as possible from the front of s*
3. System State Update
- Move the simulation clock forward according to time elapsed since previous system update
- Update flight ETAs, check for service completions and update weather information
- Release any flights that are marked for release
- Adjust S, S_min, l based on number of flights remaining
4A. Repopulation (Type 1)
- Eject all sequences from population
- Obtain new sequence b by appending new flights to s*; add b to population
- Create new sequences by heuristically modifying b until population size is S
4C. Repopulation (Type 2)
- Identify best sequence (s*) in current population
- If m ≥ m_mut apply a mutation to s*
- Create new sequences by heuristically modifying s* until population size is S
Have any flights just been released?
Yes
No
Is population size smaller than S_min?
Yes
No
Is n ≥ n_repop?
Yes
No
4B. Filter Population
- Discard the worst-performing sequences in current population until there are S_min remaining

Figure 3 **The SimHeur algorithm.**

more recently-updated system information. Subsequently, in step 2B, we use an online ranking and selection method and discard any sequences s which fail to satisfy the criterion

$$V_s^{(n)} \leq V_{s'}^{(n)} + Z_{s,s'} \quad \forall s' \in \mathcal{S} \setminus \{s\}, \tag{16}$$

where the threshold $Z_{s,s'}$ is given by

$$Z_{s,s'} = \left(\frac{z_{1-\eta/2}^2}{n-1} \left(W_s^{(n)} + W_{s'}^{(n)} - \left[V_s^{(n)}\right]^2 - \left[V_{s'}^{(n)}\right]^2 \right) \right)^{1/2} \tag{17}$$

and $z_{1-\eta/2}$ is a quantile of the standard normal distribution. This criterion is based on standard methods for ranking and selection in simulation optimization (Nelson (2013)).

In step 2C we identify the sequence s with the smallest $V_s^{(n)}$ value in the current population and, provided that the number of cost estimates generated meets a certain threshold (denoted n_{rel}), mark certain flights from the front of this sequence to be released from the pool (although their actual release occurs in step 3). This step also includes a mechanism for deciding whether it would be advantageous to delay the release of flights in order to gather more information (through system state updates) before committing flights to the queue. The mechanism is based on evaluating the likelihood of a flight j landing immediately upon realizing its unconstrained arrival time (i.e. $L_j = A_j$), as opposed to having to wait in the queue for other flights to land before landing itself (i.e. $L_j = L_i + M_{ij}$ for some i). If the latter scenario is much more likely than the former, then there may be no advantage in releasing j immediately, as an immediate release would only cause it to spend longer waiting in the queue and would not enable an earlier landing time; thus, in this case it may be better to postpone the decision of which flight(s) to release next.

In step 3 we move the simulation 'clock' forward by an increment that depends on the actual amount of CPU time spent since the previous visit to this step and update the system state accordingly. This means we update the latest ETAs $X_i(t)$ for flights i that have not yet realized their unconstrained arrival times A_i and also check for service completions since the previous update. We also update the weather forecast information (if applicable). Any flights marked for release in step 2C are also released from the pool in this step.

If any flights are released in step 3, then in step 4A we generate a completely new population with a new set of sequences to be evaluated. The method for generating these sequences is based on making random changes to the 'old' best-performing sequence. Step 4B is encountered if the total number of cost estimates generated for the sequences in the current population has reached a certain threshold (denoted n_{repop}). At this point, we remove the worst-performing sequences from the current population and move to step 4C, in which these are replaced by new randomly-generated sequences in order to continue exploration of the solution space. As shown in Figure 3, step 4C can also be encountered if the population size S has dropped below a certain minimum level (denoted $S_{\min}$) due to removal of sequences in the ranking and selection step (2B).

Notably, step 4C also includes a 'mutation' step, which is implemented if and only if a certain number (denoted m_{mut}) of 'repopulations' have already been carried out without discovery of any new sequences that perform better than the best sequence in the current population. The purpose of this mutation step is to force the algorithm to explore a different region of the solution space if it seems to be making no further progress in the current area of exploration, and this is consistent with the general VNS methodology (Mladenovic and Hansen (1997)).

This completes our high-level overview of the steps in the SimHeur algorithm. Please find more complete details of these steps in Appendix A.

3.2. An alternative approach based on expected values

In order to evaluate the performance of the SimHeur algorithm described in Section 3.1 we will compare it to some alternative approaches. One such alternative is to estimate the costs associated with different sequences by assuming that random variables such as Q_i, M_{ij}, U_0, U_1 etc. conform to their expected values, conditioned on the latest system state information available. Computationally, this is much less demanding than generating random sample trajectories and updating performance measures of candidate runway sequences as described in step 2A of SimHeur, but it is also less accurate. We refer to this simpler algorithm as 'DetHeur' and the steps involved are similar to those described in Section 3.1 for SimHeur, but with some simplifications due to the fact that the sample trajectories in step 2A are generated using expected values rather than random sampling. In DetHeur, the ranking and selection step (2B) is no longer required, as we use only a single trajectory to evaluate performances of different sequences and then allow the removal of poor-performing sequences to be handled entirely within steps 4A-4C. Please find further details of how the DetHeur algorithm works in Appendix B.

Like SimHeur, the DetHeur algorithm always uses the latest available system state information (obtained in step 3) in order to make decisions, but it differs from SimHeur by making decisions based on deterministic forecasts. An important advantage of DetHeur is that it enters the repopulation step 4C much more frequently than SimHeur (due to the removal of the condition $n \geq n_{\text{repop}}$) and this implies that it is able to spend more time searching the solution space than SimHeur, with more random mutations. However, we conjecture that this advantage diminishes as the amount of computational power increases, because the SimHeur algorithm only needs to be able to explore 'enough' of the solution space to be able to find the best solution. If the amount of computational effort spent on exploration of the solution space is already sufficient to find the best solution, then there is no advantage to be gained by exploring further.

We also note that DetHeur lacks the mechanism described in Step 2C of SimHeur for delaying release of flights from the pool in order to acquire more information from system state updates. The mechanism used by SimHeur is based on probabilistic assessments of whether or not flights will be delayed in the queue after being released, and DetHeur cannot emulate such a mechanism in any effective way. We elaborate further on this in Appendix B.

We note that in the stochastic programming literature, the idea of comparing the optimal solution to a stochastic program with the optimal solution to a corresponding 'expected value' problem is widely used (see Birge and Louveaux (2011)). One can compare the performances of the two solutions under stochastic conditions in order to evaluate the 'value of the stochastic solution' (VSS). A similar approach can be used in our problem, by comparing the performances of SimHeur and DetHeur in order to investigate the benefits of being able to simulate random events based on knowledge of the underlying probability distributions.

3.3. Other sequencing strategies and benchmarks

In our numerical experiments in Section 5 we make use of some additional benchmarks for evaluating the performances of SimHeur and DetHeur. The first of these is the cost associated with a simple 'first-come-first-served' (FCFS) rule, in which flights are released immediately when they arrive at the pool ($R_i = Q_i$ for $i \in \mathcal{F}$). The FCFS rule is simple to implement, but it pays no attention to scheduled landing times or the time separations required between different weight class combinations, so the resulting landing sequence may be far from optimal.

We also consider another policy obtained from a static, deterministic optimization procedure, referred to as 'DStat' for short. The DStat policy can be expressed in the form of a single sequence, i.e. an ordering of the flights $i \in \mathcal{F}$, which is computed *once* at the beginning of the interval $\mathcal{T}$ (assuming knowledge of the pre-tactical delays Δ_i^{pre}) and not updated at any future time points. More specifically, at the beginning of $\mathcal{T}$, we assume that all random variables, including unconstrained arrival times, service times and weather transitions conform to their expected values and then aim to find the runway sequence that optimizes the objective function (13) under such conditions. Although the resulting optimization problem is static and deterministic, it still has very high combinatorial complexity, and we therefore aim to solve the problem heuristically by making successive improvements to a first-come-first-served sequence until no further improvements can be found; further details are given in Appendix C. After the DStat sequence has been obtained, its performance under the stochastic conditions of our model can be evaluated.

Although the DStat policy is similar to DetHeur in that both algorithms treat the problem as deterministic and rely upon expected values, the DStat policy is much more simplistic than DetHeur as it is not a *dynamic* policy; that is, it lacks the ability to update sequencing decisions during $\mathcal{T}$ in response to the latest observed information.

3.4. Implementation in practice

Here we address the question of how the sequencing methods proposed in Sections 3.1-3.3 could be implemented by air traffic controllers (ATCOs) in practice. Typically, when aircraft are within about 200 nautical miles (roughly 40 minutes) of their destination airport, they enter a region of controlled airspace known as the Extended Terminal Maneuvering Area (E-TMA). At this point, their movements are directed by ATCOs using decision support tools such as the Arrival Manager (AMAN) in Europe, or the Traffic Management Advisor in the US (Khassiba et al. (2020)). ATCOs are able to coordinate aircraft movements in order to achieve an intended sequence of landings, using maneuvers such as vectoring, holding and path stretching (Bennell et al. (2011), Errico and Di Vito (2019)). The time window available for sequencing is limited, however, as aircraft that are already within a certain time threshold of expected landing (often referred to as a *freeze horizon*; see Moser and Hendtlass (2007), Murca and Muller (2015), Bennell et al. (2017)) may no longer have their positions in the landing sequence adjusted.

In recent years, ATM research programmes such as SESAR and NextGen have enabled greater information sharing between airspace users, implying that tools such as AMAN could be extended in order to provide decision support over much longer operational horizons. For example, aircraft up to 500 nautical miles (roughly 2 hours) away from landing could be advised to make speed or trajectory changes in order to achieve a desirable incoming traffic flow (Tielrooij et al. (2015), Khassiba et al. (2020)). This represents a form of *pre-sequencing* that reduces the need for holding patterns and enables savings in fuel consumption (SESAR (2023a,b)).

From an ATCO's perspective, the critical task is to ensure that the movements of aircraft within the E-TMA (or in sufficient proximity to it) are consistent with the need to achieve a certain desired sequence of landings. In line with other studies in this area, we do not aim to specify the detailed movements of aircraft (in terms of heading or altitude changes, etc.) that are required, but instead assume that the specification of an intended landing sequence at any given time is sufficient for ATCOs to arrange aircraft movements accordingly. In our particular model, the 'pool' (see Section 2.4) includes aircraft that have entered the E-TMA but have not yet passed beyond the freeze point, and therefore our decisions are focused on these.

As in previous studies on dynamic runway scheduling, our optimization approach must be implementable in real time, without any delays in prescribing actions. Previous studies have addressed this need by employing a 'rolling horizon' approach, in which an optimization problem is solved at regular time intervals (e.g. every 5 minutes) in order to find the optimal ordering of planes that are currently 'schedulable' (see Sama et al. (2013), Murca and Muller (2015), Solak et al. (2018)). The optimization problem may be solved via two-stage stochastic programming or via a metaheuristic algorithm, for example. Although the rolling horizon approach entails some time delay in between successive optimization runs, it nevertheless fulfils the need of the dynamic problem by ensuring that a perceived 'optimal' sequence (specifically, the sequence given by the most recent optimization run) is always available for air traffic controllers to act upon. In our model, the nature of uncertainty is quite different as it is based on continuous-time stochastic processes. However, our SimHeur and DetHeur algorithms work effectively as alternatives to the rolling horizon approach because they rely on a VNS-based method to continually search the solution space based on the latest available state information. Importantly, the 'anytime' nature of the VNS method ensures that a strong-performing solution is always available. As with the rolling horizon approach, ATCOs can act based on the latest-generated optimal sequence, without needing to wait for the results of any further computational procedures.

In order to implement SimHeur, one must evaluate the performances of candidate sequences by continuously running background simulations; thus, it is essential for these simulations to be based on a mathematical model that accurately represents the real-world dynamics. The DetHeur algorithm, on the other hand, is slightly less demanding in this respect, since it relies only upon expected values rather than full probability distributions. In the case of SimHeur, the need for a simulation model that accurately represents the 'ground truth' could potentially be a barrier to implementation. However, it is worth noting that simulation is already recognized as a powerful technique for optimizing air traffic control outcomes in a real-time context, and heuristic methods based on simulated trajectories of future events have been deployed in practice. For example, NATS Ltd (`https://www.nats.aero/`), which offers air traffic control services to some of the busiest airports worldwide, has developed a heuristic event-based model that makes use of stochastic input data to evaluate a range of operational metrics including on-time performance, demand on airport infrastructure and environmental impacts (NATS Ltd (2023a,b)). Thus, there is already a precedent for simulation to be used as a decision support tool for air traffic controllers.

To summarize, our paper follows previous studies by recommending that the movements of aircraft within or near the E-TMA are managed by ATCOs with reference to a target landing sequence that is always available and frequently updated. Unlike previous studies, however, we model uncertainty using continuous-time stochastic processes and (in the case of the SimHeur algorithm) recommend simheuristics as a means of optimization. Model calibration is clearly an important issue, and the next section describes how the parameters of our model may be calibrated according to actual data from Heathrow Airport.

4. Data acquisition and model calibration

In this section we describe our use of on-time performance data for arrivals at London Heathrow Airport to estimate parameter values for the model described in Section 2 and design an appropriate flight schedule for model testing purposes.

As mentioned in Section 2.1, Heathrow Airport usually operates with one of its two runways used exclusively for arrivals. We decided to select a particular day of operations and look up historical on-time performance data for the arriving flights scheduled during a particular part of that day. Our chosen day was August 1st, 2019 and we considered the eight-hour period from 6:00AM (inclusive) to 2:00PM (non-inclusive), during which there were 323 arriving flights scheduled. We take these 323 flights as our set of flights $\mathcal{F}$. We note that our chosen date ensures that the flight schedule and historical dataset are not affected by the disruption caused by the Covid-19 pandemic.

Figure 4 shows the numbers of arrivals scheduled within each half-hour interval during the period 6:00AM-2:00PM, with a breakdown of aircraft weight classes in each interval also provided. Aircraft can be divided into different weight classes according to their ‘maximum take-off mass’ (MTOM), measured in kilograms (UK Aeronautical Information Services (2019)). We found that the vast majority of aircraft arriving at Heathrow on the day of interest belonged to the ‘heavy’ and ‘lower medium’ categories, with only a small minority belonging to the ‘upper medium’ and ‘small’ categories. Larger aircraft tend to be used for long-haul, intercontinental flights, whereas smaller aircraft usually arrive at Heathrow from other European airports.

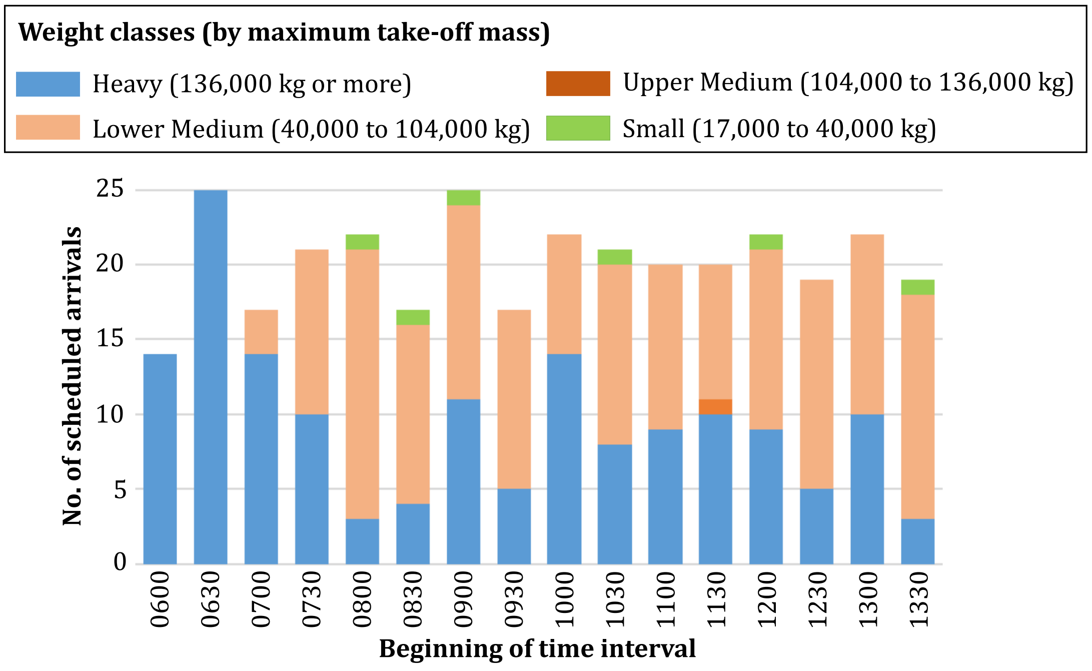

Figure 4 Numbers of arrivals scheduled by half-hour time interval between 6:00AM and 2:00PM at Heathrow Airport on August 1st, 2019.

In order to specify the required time separation e_{ij} between a leading aircraft of type i and a following aircraft of type j (where $i, j \in \mathcal{W} = \{\text{Heavy, Upper Medium, Lower Medium, Small}\}$) we use the matrix of separation times from Bennell et al. (2017), which has been calibrated according to observed data from Heathrow Airport. This matrix is shown in Table 1.

For each of the 323 arrivals at Heathrow during the eight-hour period of interest, we used historical data available at `http://www.flightradar24.com` to find the exact arrival times (to the

Table 1 Required separation times in seconds for different leader-follower aircraft weight class pairs (H = Heavy, UM = Upper Medium, LM = Lower Medium, S = Small).

		Follower			
		H	UM	LM	S
Leader	H	97	121	121	145
	UM	72	72	97	97
	LM	72	72	72	72
	S	72	72	72	72

nearest minute) of all flights with the same flight number (implying the same flight carrier, origin airport and destination airport) over the 360-day period from August 8th, 2018 to August 2nd, 2019. Hence, for each flight in $\mathcal{F}$, we have a set of historical landing times collected over a 360-day period and are able to estimate means, variances etc. of the flight's punctuality with respect to its scheduled arrival time. The majority of flights in $\mathcal{F}$ (221 out of 323) operated on at least 300 days during this 360-day period, and the total number of records we have (where the term 'record' here refers to the recorded landing time of a flight in $\mathcal{F}$ during the 360-day historical period) is 98,814, equating to about 306 per flight on average.

Ideally, we would like to use these historical data to estimate probability distributions for the pre-tactical and tactical delays affecting the unconstrained landing times A_i for flights $i \in \mathcal{F}$. However, the records in our data are *actual* landing times (denoted L_i in our model), which may be affected by queueing delays and other congestion effects, as well as poor weather. The records in our dataset do not provide us with a means of calculating the extent to which landing times are influenced by airport congestion and other similar effects, so we must rely on an approximate method to fit distributions for the unconstrained landing times in our model. Our method involves filtering out some of the historical data by identifying days on which queueing delays appeared to be particularly significant; please see Appendix D for further details.

After filtering out some data as described above, we are left with a reduced dataset consisting of 248 days' worth of data, with 199.5 records per flight on average. For each flight $i \in \mathcal{F}$, we use these data to calculate the sample mean and sample variance of the actual landing time. We then consider how much of this variance should be explained by pre-tactical uncertainty in our model. Recall that the mean and variance of A_i are given by equations (8) and (9) respectively. Our approach is to equate the expressions for $\mathbb{E}[A_i]$ and $\mathrm{Var}(A_i)$ to the sample mean and sample variance for flight i in our dataset, denoted $\bar{x}_i$ and s_i^2 respectively, in order to derive suitable values for our model parameters. As mentioned in Section 2, we rely upon gamma distributions for modeling the pre-tactical delays. Specifically, we assume that

$$\Delta_i^{\text{pre}} = Y_i - (a_i - h_i), \tag{18}$$

where Y_i is gamma-distributed with the density function

$$f_{Y_i}(t) = \frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)} t^{\alpha_i - 1} \exp(-\beta_i t) \quad (t > 0),$$

which has mean $\alpha_i \beta_i^{-1}$ and variance $\alpha_i \beta_i^{-2}$ $(\alpha_i, \beta_i > 0)$. Then, by setting $\bar{x}_i$ and s_i^2 equal to the expressions in (8) and (9) respectively, we obtain the following expressions for α_i and β_i:

$$\alpha_i = \frac{(\bar{x}_i - h_i)^2}{s_i^2 - \sigma_i^2(\bar{x}_i - h_i)}, \quad \beta_i = \frac{\bar{x}_i - h_i}{s_i^2 - \sigma_i^2(\bar{x}_i - h_i)}. \tag{19}$$

Our dataset provides values of $\bar{x}_i$ and s_i^2 for each $i \in \mathcal{F}$. In calculating these sample statistics, we have found that it is necessary to exclude 'outliers', as these tend to dominate the calculation of the sample variances s_i^2 and cause the distributions of Δ_i^{pre} in our model to be too platykurtic in shape. We define an outlier as an arrival that does not occur within 120 minutes of its scheduled arrival time. Using this definition, about 1.06% of records in our dataset are outliers.

We set the parameter h_i to be 15 minutes earlier than the scheduled departure time d_i in our numerical experiments, on the basis that this is a reasonable estimate for the average taxi-out time before departure (Badrinath et al. (2020)). We consider different possible cases for σ_i in our experiments, with larger values implying a greater proportion of tactical uncertainty (as opposed to pre-tactical). We note that the expressions in (19) are valid only if $\sigma_i^2 < s_i^2(\bar{x}_i - h_i)^{-1}$. If we wish to consider a larger σ_i^2 value, we can simply set Δ_i^{pre} equal to its data-calibrated expected value $\bar{x}_i - a_i$ rather than sampling it using (18). This represents the case where all of the variation in A_i is accounted for at the tactical level.

Figure 5 shows the results of fitting distributions for Δ_i^{pre} and $(\Delta_i^{\text{pre}} + \Delta_i^{\text{tac}})$ to the empirical data by calculating values for α_i and β_i using the method described above. We have illustrated the method using a particular flight in $\mathcal{F}$, referred to as 'Flight X' to preserve anonymity. The grey histogram shows the distribution (over all 248 days in our reduced dataset) of the delay (or 'lateness') in minutes. The red, green and blue solid curves show the fitted distributions of the pre-tactical delay given σ_i values of 1, 0.75 and 0.5 respectively. Similarly, the red, green and blue dashed curves show the fitted distributions of the overall delay (including pre-tactical and tactical) for the same σ_i values. Recall that if σ_i is reduced, then a larger proportion of the empirical variance is accounted for at the pre-tactical level in our model. This explains why the solid curves become 'flatter' (indicating more variance in the pre-tactical delay) as σ_i becomes smaller.

It can be seen that the 3 dashed curves in Figure 5 are almost indistinguishable from each other. This shows that the value of σ_i has almost no effect on the unconditional distribution of A_i. Indeed, the role of σ_i in our model is to determine the relative proportions of pre-tactical and tactical uncertainty affecting flight i's arrival time; it does not affect the total amount of uncertainty. However, the value of σ_i does have a very significant effect on the nature of the decision problem formulated in Section 2, as larger values imply that decisions must be made under higher levels of uncertainty. This is demonstrated by our numerical results in Section 5.

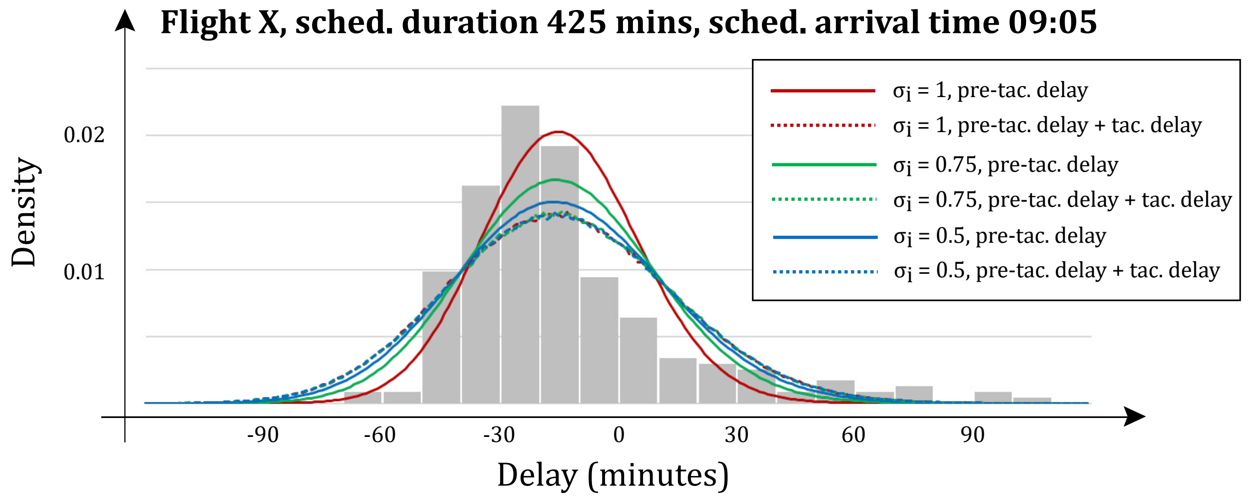

Figure 5 **The results of fitting distributions for Δ_i^{pre} and $(\Delta_i^{\text{pre}} + \Delta_i^{\text{tac}})$ using the empirical data for a particular flight in $i \in \mathcal{F}$, with the results shown for three different values of σ_i.**

5. Numerical experiments

Our computational study is based on the schedule of operations at Heathrow Airport during the eight-hour period from 6:00AM to 2:00PM on August 1st, 2019. Our set of flights $\mathcal{F}$ consists of all 323 flights scheduled during this eight-hour period. We compare the performances of the various solution algorithms described in Section 3 by randomly generating a series of 5000 test scenarios. To ensure fairness, all of our experiments are performed on a single desktop computer with an Intel(R) Core(TM) i7-9700 CPU and 16GB of RAM. Programs are implemented using Python, with the PyPy Just-in-Time compiler (`https://pypy.org`) used to enhance computation speed.

Each of the 5000 test scenarios consists of pre-tactical delays Δ_i^{pre} for all $i \in \mathcal{F}$, complete random trajectories $\{X_i(t)\}_{t\in\mathcal{T}}$ for all $i \in \mathcal{F}$, weather forecast trajectories $\{T_0(t), T_1(t)\}_{t\in\mathcal{T}}$ and a set of possible separation times $\{M_{ij}\}$ for each $j \in \mathcal{F}$ (taking into account all possibilities for the weight class of the predecessor flight i). In each scenario, the pre-tactical delays Δ_i^{pre} are generated immediately by sampling from gamma distributions with parameter values configured according to the historical data as described in Section 4. The flight trajectories, weather forecasts and separation times are also generated immediately, but this information (unlike the set of pre-tactical delays) is initially 'hidden' from the decision-maker and revealed gradually as time progresses. This hidden information comprises the 'true' sample path, which we refer to as Ω^{true}. Please see Appendix E for details of how Ω^{true} is generated.

5.1. How time is handled in our experiments

In each test scenario we wish to evaluate the performances of the SimHeur, DetHeur, FCFS and DStat algorithms, as described in Section 3. Recall that the SimHeur and DetHeur algorithms both operate by continuously updating their beliefs of the optimal landing sequence, with the simulation 'clock' being moved forward in increments that depend on the actual amounts of time spent by these algorithms on their various computational steps. Thus, in order to allow these algorithms to perform the same number of computations that they would perform in a 'real time' setting, we would need to run both of them for (at least) 8 hours. However, in this computational study we wish to test a large number of different random scenarios, and it is not practical to spend 16 hours on each individual scenario. We therefore compress the time scale in our experiments so that when the simulation clock is moved forward in step 3, the time increment added is 60 times the actual amount of CPU time elapsed since the previous increment was made. This effectively means that one second of computation time is equated to one minute of operational time, and the SimHeur and DetHeur are only allowed to perform 1/60 of the number of steps that they would be able to perform in reality. We conjecture that SimHeur is likely to be affected more by this time compression than DetHeur due to its higher computational demands, and hence the improvements in performance of SimHeur relative to DetHeur that we find in our experiments are likely to be 'lower bounds' of the improvements that would be achievable in reality.

Note: For clarity, when time units (e.g. minutes, hours) are referred to in the remainder of this section, these should be understood as units of 'operational time' rather than 'CPU time'; for example, we still refer to the time interval 6:00AM-2:00PM as being 8 hours long, although our time compression implies that it is simulated in only 8 minutes of CPU time.

We also note that this time compression does not have any effect on the performance of the FCFS policy, as the landing times L_i for each $i \in \mathcal{F}$ under FCFS can be calculated in a deterministic way as soon as the sample path Ω^{true} has been generated. In practice, this means that there is no need to use a simulation clock at all when calculating the FCFS performance; instead, it can be determined immediately and costs a negligible amount of CPU time. Similarly, the DStat algorithm does not require a simulation clock and is not affected by the time compression, since its recommended sequence is obtained using only the information available at the beginning of $\mathcal{T}$ and its actual performance can then be calculated deterministically using Ω^{true}.

We count time in minutes and start counting at 5:00AM, so that $t = 60$ indicates 6:00AM and $t = 540$ indicates 2:00PM. One reason for starting at 5:00AM is so that the SimHeur and DetHeur algorithms can have some initial time to evaluate possible sequences and decide on their estimated 'optimal' sequences before the time window of interest (6:00AM-2:00PM) actually begins. In addition, some early-arriving flights might arrive in the pool between 5:00AM and 6:00AM, in which case we would like to have the option of releasing them earlier than 6:00AM. As discussed in Section 2, the departure time d_i for a particular flight $i \in \mathcal{F}$ is allowed to occur before time $t = 0$. In the context of our model, this implies that the time h_i at which flight i's trajectory $X_i(t)$ begins varying according to Brownian motion may precede $t = 0$. If this is the case, then (as discussed in Section 2.4) we must generate an initial ETA $X_i(t)$ by sampling from a Normal distribution with mean $a_i + \Delta_i^{\text{pre}}$ and variance $\sigma_i^2|h_i|$. Although all flights in $\mathcal{F}$ are scheduled to land before 2:00PM, we need to allow some extra time beyond this in order to simulate late arrivals, so we define $\mathcal{T} = [0, T]$ where $T = 780$. This allows for the possibility of flights landing up to 4 hours late. (In practice, we do not need to continue simulating until time T if all flights have already landed.) Ideally, each trajectory $X_i(t)$ should vary continuously throughout $t \in \mathcal{T}$, but our method of pre-generating the random information (as discussed in Section 3) implies that we can only store a finite number of $X_i(t)$ values for each $i \in \mathcal{F}$. Therefore, for the purposes of these experiments, we generate and store values of each $X_i(t)$ (and also $T_0(t)$, $T_1(t)$) only for $t = 0$, 0.01, 0.02, ..., T, so that the aircraft ETAs and weather forecasts change every 0.01 minutes.

5.2. Adjusting the values of physical parameters

In the implementation of this study, we can distinguish between 'physical' parameters that affect the dynamics and costs of the problem itself and 'algorithmic' parameters that are only relevant to the workings of the SimHeur and DetHeur algorithms. The former class includes the tactical-stage variance parameters σ_i, the Erlang parameter k (affecting separation time variances), the initial weather forecast parameters t_0 and t_1, the weather variance parameter ν, the times h_i at which flights enter the 'tactical uncertainty' stage, the scaling factor ϕ affecting required time separations under bad weather, the threshold τ that determines when sequencing decisions are made, the relative cost parameters g_i for $i \in \mathcal{F}$, the 'tolerance' parameters $\gamma^{[S]}$ and $\gamma^{[W]}$ included in the objective function (13) and the objective function weights $\theta^{[S]}$ and $\theta^{[W]}$. The latter class includes parameters such as $S_{\min}$, l, n_{rel}, n_{repop}, m_{mut} that should be chosen carefully in order for the SimHeur and DetHeur algorithms to perform well, but do not change the nature of the physical problem. In order to avoid having too many variable factors, we have chosen to vary only the physical parameters in this computational study, and keep the values of the algorithmic parameters fixed throughout all of our experiments. The values of the algorithmic parameters have been chosen according to a preliminary study in which the SimHeur and DetHeur algorithms were applied to some small 'test' problems; see Appendix F for further details.

For the purposes of this study, we assume that the variance parameter σ_i affecting the tactical uncertainty is the same for all flights and set $\sigma_i = \sigma$ for all $i \in \mathcal{F}$, but we consider different possible cases for σ. Specifically, we conduct 1000 experiments for each of the 5 cases $\sigma = 0.1$, $\sigma = 0.3$, $\sigma = 0.5$, $\sigma = 0.7$ and $\sigma = 0.9$. For each value of σ, we then set the values of α_i and β_i as specified in (19) in order to ensure that the values of $(\Delta_i^{\text{pre}} + \Delta_i^{\text{tac}})$ are consistent with the historical data. Thus, each flight has its own unique distribution for its unconstrained landing time A_i, and we consider different possible cases for the proportion of $\text{Var}(A_i)$ accounted for at the tactical stage as opposed to the pre-tactical stage. For further details of how we generate the physical and algorithmic parameters in our experiments, please refer to Appendix F.

5.3. Performance comparisons with respect to weighted objective function

In this subsection we present overall summative findings, obtained by collating the results from all 5000 scenarios in our study. Please see `https://doi.org/10.5281/zenodo.7339074` for a detailed summary of results from all 5000 experiments, including the values of the physical parameters and the performances achieved by the various algorithms in each individual scenario.

Table 2 shows a summary of the improvements achieved by the SimHeur algorithm against the other algorithms with respect to our weighted objective function (13). Columns 3, 4 and 5 show 95% confidence intervals for the mean percentage improvement (with respect to the value of (13)) achieved by SimHeur against DetHeur, FCFS and DStat respectively. For clarity, the percentage improvement in a particular scenario is calculated as $100 \times (\Phi_{\text{H}} - \Phi_{\text{SH}})/\Phi_{\text{H}}$, where Φ_{SH} is the objective function value under SimHeur and Φ_{H} is the corresponding value under another policy $H \in \{\text{DetHeur, FCFS, DStat}\}$. Column 6 (resp. 7, 8, 9) shows the percentage of all experiments in which SimHeur (resp. DetHeur, FCFS, DStat) achieved the smallest objective function value. The first row shows the results from all 5000 experiments, and the next 5 rows show the results for each of the different cases we considered for the value of σ.

Table 2 **Performance comparisons between SimHeur (SH), DetHeur (DH), First-come-first-served (FC) and DStat (DS) with respect to the value of the weighted objective function (13).**

		Pct. Improvement			**Pct. of Experiments**			
σ value	Count	SH vs. DH	SH vs. FC	SH vs. DS	SH best	DH best	FC best	DS best
Any value	**5000**	**10.83 ± 0.41**	**43.17 ± 0.35**	**64.99 ± 0.73**	**74.52**	**24.52**	**0.20**	**0.76**
0.1	1000	2.14 ± 0.56	51.20 ± 0.62	19.70 ± 0.78	56.30	39.90	0.00	3.80
0.3	1000	7.56 ± 0.72	48.33 ± 0.63	59.54 ± 0.76	75.30	24.70	0.00	0.00
0.5	1000	12.43 ± 0.88	44.49 ± 0.64	76.22 ± 0.53	79.50	20.50	0.00	0.00
0.7	1000	15.22 ± 0.96	38.95 ± 0.70	83.19 ± 0.41	80.10	19.80	0.10	0.00
0.9	1000	16.82 ± 1.01	32.85 ± 0.75	86.27 ± 0.33	81.40	17.70	0.90	0.00

One would expect the relative improvement of SimHeur versus DetHeur to increase as the amount of stochasticity in our model increases. Indeed, Table 2 shows a clear trend for SimHeur to improve its advantage over DetHeur as σ is increased. On the other hand, SimHeur's advantage over FCFS diminishes when σ is increased. This seems to indicate that the FCFS policy becomes stronger when there is more tactical uncertainty, which can be explained by the fact that flights are more likely to arrive in the pool significantly later than expected in such circumstances, and must then be released as soon as possible in order to avoid large penalties. The DStat algorithm performs relatively well in the $\sigma = 0.1$ case and it can even outperform the other policies in some scenarios, which is a sign that the heuristic procedure described in Appendix C works well for solving the relevant static, deterministic optimization problem. However, its performance deteriorates sharply as σ increases and it becomes worse than FCFS in such cases. Indeed, larger σ values can cause DStat to incur very large holding costs due to the increased likelihood of flights arriving in the pool either very late (and thus causing other flights to wait before being released) or very early (and thus having to wait themselves).

5.4. Performance comparisons with respect to delays and other measures

Here we investigate how SimHeur compares to DetHeur, FCFS and DStat with respect to a range of other performance measures in order to obtain further insights into their different modes of decision-making and the effects in terms of schedule delays and operational delays. Table 3 shows a list of performance measures that can be calculated from the set of results for an individual scenario (by taking an average over flights $i \in \mathcal{F}$, or calculating a relevant percentile, for example). Each of these performance measures has been calculated for each of the four policies in each scenario, and the averages are then taken over all scenarios and shown in columns 2-5.

The first 7 rows of Table 3 are related to schedule delays, defined as the differences between actual landing times L_i and pre-scheduled times a_i. In terms of average schedule delays, SimHeur achieves improvements of about 19%, 39% and 76% versus DetHeur, FCFS and DStat respectively. Significant improvements can also be seen in the percentiles of the schedule delay. The next 7 rows are related to operational delays, defined as $L_i - (A_i - \rho_i)$ for $i \in \mathcal{F}$. As explained in Section 2.4, this can be interpreted for $i \in \mathcal{F}$ as the sum of the holding time in the pool (ρ_i) and the waiting time in the queue $(L_i - A_i)$, i.e. the total amount of airborne delay. The savings in average operational delays achieved by SimHeur versus DetHeur, FCFS and DStat are about 9%, 20% and 56% respectively. It is important to note that these results are averaged over all 5000 scenarios. Further inspection shows that the savings achieved by SimHeur are strongly influenced by parameters such as σ, with larger values of σ tending to yield greater savings.

The last 3 rows of Table 3 show some miscellaneous additional statistics of interest. The 15th row shows that the SimHeur, DetHeur and DStat policies tend to 'group' flights of the same weight class together to a greater extent than FCFS. It is well-known in aircraft sequencing problems that arranging the landing sequence into 'strings', with each string consisting of consecutive flights of the same weight class, is the most efficient way to minimize average separation times (or, equivalently, maximize average throughput rates). Obviously, the FCFS policy will ignore this principle, but it

Table 3 Performance comparisons between SimHeur, DetHeur, First-come-first-served and DStat with respect to various scenario-based performance measures (results are averaged over all 5000 scenarios)

Performance measure	SimHeur	DetHeur	FCFS	DStat
Avg. schedule delay $L_i - a_i$ (for all $i \in \mathcal{F}$)	6.19 min	7.64 min	10.10 min	25.81 min
75th percentile of schedule delay	17.85 min	19.29 min	24.90 min	37.36 min
90th percentile of schedule delay	29.67 min	30.68 min	40.15 min	47.29 min
95th percentile of schedule delay	39.26 min	40.14 min	50.98 min	55.49 min
Pct. of flights $i \in \mathcal{F}$ with $L_i > a_i + \gamma$	51.88%	55.21%	53.70%	75.41%
Avg. of $L_i - (a_i + \gamma)$ for flights i with $L_i > a_i + \gamma$	14.67 min	15.06 min	20.51 min	26.93 min
Avg. schedule delay cost $C_i^{[S]}$ (for all $i \in \mathcal{F}$)	179.61	200.10	328.39	775.90
Avg. operational delay $L_i - (A_i - \rho_i)$ (for all $i \in \mathcal{F}$)	15.28 min	16.73 min	19.19 min	34.90 min
75th percentile of operational delay	21.10 min	23.11 min	25.76 min	44.26 min
90th percentile of operational delay	27.39 min	28.92 min	30.26 min	52.46 min
95th percentile of operational delay	31.34 min	32.58 min	32.66 min	57.71 min
Avg. holding time in pool ρ_i (for all $i \in \mathcal{F}$)	5.54 min	4.55 min	0 min	13.41 min
Avg. waiting time in queue $L_i - A_i$ (for all $i \in \mathcal{F}$)	9.73 min	12.18 min	19.19 min	21.49 min
Avg. operational delay cost $C_i^{[W]}$ (for all $i \in \mathcal{F}$)	270.49	317.66	370.92	1389.14
Pct. of flights landing after a flight of the same type	75.43%	77.06%	56.44%	79.87%
Pct. of flights released from pool during bad weather	10.16%	10.20%	11.32%	10.83%
Avg. deviation from FCFS position (per flight in $\mathcal{F}$)	2.84	3.00	0	6.04

is unsurprising that the other policies appear to follow it. The DetHeur and DStat policies tend to arrange flights based on weight classes slightly more than SimHeur, which may be explained by their tendency to underestimate the costs of potential runway sequences (due to ignoring the effects of uncertainty) and hence overestimate the amount of available time to arrange flights into suitable ‘strings’ before they need to be released from the pool.

The 16th row shows that the FCFS policy is slightly more likely than the other policies to release flights from the pool during bad weather, which may be explained by the fact that the other policies might sometimes attempt to delay a flight’s release until weather conditions have improved. The 17th row shows the average deviation per flight from the FCFS sequence, which is calculated for a particular flight as the difference between its position in the actual landing sequence and the position that would occur under a FCFS policy (for example, if a particular flight is the 10th to arrive in the pool but the 13th to land, then it has a deviation of 3). The results show that DetHeur deviates slightly more from the FCFS sequence than SimHeur on average, which follows from the fact that it has a slightly greater tendency to attempt to group flights into strings of the same weight class (as discussed above). Meanwhile, the DStat policy’s average deviation is much higher, which may be explained by the fact that its sequencing decisions are made without knowledge of the actual times that planes arrive in the pool.

5.5. Is SimHeur always better than DetHeur?

Our summative results in Table 2 have shown that although SimHeur performs better than the other policies over the full set of 5000 scenarios, its relative advantage over the other policies varies considerably according to the value of σ. We have also found that other physical parameters, such as k and γ, can affect the relative strength of SimHeur, although they are not as influential as σ. More interestingly, we have found that changing the weights $\theta^{[S]}$ and $\theta^{[W]}$ used in the objective function (13) can influence the results greatly, and if $\theta^{[S]}$ is much smaller than $\theta^{[W]}$ (implying that operational delays are given more weight than schedule delays in the objective), then SimHeur may perform worse than DetHeur. To illustrate this, Table 4 shows a comparison between the performances of SimHeur and DetHeur in each of the 5 different cases we considered for $\theta^{[S]}$ and $\theta^{[W]}$. In each of the 5 cases, we have reported 95% confidence intervals for the percentage improvement given by SimHeur vs. DetHeur with respect to the objective function value (in column 3), and also for the two separate components of our objective function (in columns 4 and 5).

Table 4 Percentage improvements given by SimHeur vs. DetHeur with respect to the objective function value and its separate components, for various combinations of the objective function weights $\theta^{[S]}$ and $\theta^{[W]}$

		Pct. Improvement given by SimHeur vs. DetHeur		
Obj. function weights	Count	Weighted objective function $\sum_i g_i \left[\theta^{[S]}C_i^{[S]} + \theta^{[W]}C_i^{[W]}\right]$	Schedule delay component $\sum_i g_i C_i^{[S]}$	Operational delay component $\sum_i g_i C_i^{[W]}$
$\theta^{[S]} = 0.1$, $\theta^{[W]} = 0.9$	1015	-2.59 ± 0.62	-3.52 ± 0.39	-2.48 ± 0.66
$\theta^{[S]} = 0.3$, $\theta^{[W]} = 0.7$	995	7.96 ± 0.80	-1.54 ± 0.61	11.22 ± 0.94
$\theta^{[S]} = 0.5$, $\theta^{[W]} = 0.5$	995	15.71 ± 0.85	5.70 ± 0.72	21.53 ± 1.06
$\theta^{[S]} = 0.7$, $\theta^{[W]} = 0.3$	983	17.63 ± 0.84	14.19 ± 0.79	21.38 ± 1.10
$\theta^{[S]} = 0.9$, $\theta^{[W]} = 0.1$	1012	15.73 ± 0.77	19.59 ± 0.76	3.16 ± 1.12

The results in Table 4 show that SimHeur tends to perform worse than DetHeur with respect to both components of the objective function in the case $\{\theta^{[S]} = 0.1, \theta^{[W]} = 0.9\}$, although it is generally stronger than DetHeur in the other cases. To make sense of this, it is useful to understand how the nature of the problem changes when operational delays are given a much higher weight than schedule delays. In these cases, it becomes less important for flights to land near their scheduled times and the problem instead becomes focused on the need to arrange the sequence of weight classes in order to minimize the average time separation between consecutive landings. In these circumstances, it is rarely optimal to keep planes waiting in the pool in order to allow an earlier-scheduled plane to take the next position in the landing sequence (which might happen when schedule delays are given a higher weight) and instead the optimal decisions tend to involve releasing planes from the pool quickly, with the decision of which plane to release mainly based on weight classes rather than individual on-time requirements. To confirm this, we have found that when $\theta^{[S]} = 0.1$ and $\theta^{[W]} = 0.9$, the average holding time in the pool is 3.73 minutes under SimHeur and 1.11 minutes under DetHeur. Also, the average deviation from the FCFS sequence (per flight) is 1.09 under SimHeur and 1.21 under DetHeur. All of these statistics are smaller than the corresponding values shown in Table 3 (which are averaged over all combinations for the objective function weights), so this confirms the notion that smaller weights for the schedule delays imply shorter pool-holding times under both SimHeur and DetHeur.

Usually, an important driver for SimHeur's advantage over DetHeur is the fact that it can delay a plane's release from the pool if the probability of that plane going straight into service without having to wait in the queue is negligibly small (see Section 3.1). This mechanism allows it to gain more information before deciding which plane to release next. In the case $\{\theta^{[S]} = 0.1, \theta^{[W]} = 0.9\}$, planes must be released with greater urgency, so SimHeur's advantage in this respect is diminished. Furthermore, DetHeur has an advantage over SimHeur in that it evaluates the cost of a particular sequence much more rapidly (albeit less accurately) by using expected value trajectories rather than sampling trajectories randomly. Thus, when planes must be released from the pool quickly, DetHeur's greater evaluation speed enables it to explore the solution space faster than SimHeur and arrive at a suitable decision within a shorter timeframe.

In general, the question of whether SimHeur should be preferred to DetHeur may depend on several different performance criteria as well as practical considerations. Although our experiments have shown that SimHeur consistently outperforms DetHeur with respect to various performance measures over a range of different parameter values, we have also identified cases (as above) where DetHeur obtains an advantage due to its ability to evaluate solution costs and explore the solution space more rapidly than SimHeur. Furthermore, in all of our experiments, we have assumed that SimHeur is able to simulate the costs of different sequences using a fully correct model of the continuous-time stochastic processes underpinning the probabilistic behavior of the system. In reality, it may be difficult to model the probabilistic behavior accurately, and this could present a practical challenge. Nevertheless, the potential to achieve significant reductions in delays (and the prospect of being able to achieve even greater improvements with investment of greater CPU effort) provides strong motivation for pursuing a simheuristic approach.

We have also carried out some additional experiments to investigate what happens when there are multiple periods of bad weather during $\mathcal{T}$, instead of only a single period, and have found that

SimHeur's advantage over DetHeur tends to diminish in these situations, although it still performs well in general. The results of these experiments (with accompanying discussion) can be found in Appendix G.

6. Conclusions

This paper has introduced an original mathematical model for stochastic runway scheduling, in which aircraft ETAs and weather conditions evolve dynamically according to continuous-time stochastic processes, while runway 'service times' depend on sequence-dependent Erlang distributions. The aim is to consider a high-dimensional and information-rich environment in which air traffic controllers are able to update their plans frequently in response to the latest unfolding events. It is natural to consider a simheuristic approach to such a problem, since other conventional optimization approaches (e.g. two-stage stochastic programming) are less well-equipped to deal with the continuous nature of the information updates and decision epochs in our model.

Our numerical experiments, based on a schedule of more than 300 arrivals at Heathrow Airport and configured using a large set of historical on-time performance data, have shown that the proposed simheuristic algorithm (SimHeur) is capable of outperforming an alternative method based on deterministic forecasts (DetHeur) under a wide range of parameter values, and also improves substantially upon other more naive heuristic rules such as 'first-come-first-served'. Notably, even when the amount of stochasticity in our model is relatively low (e.g. with the choices $\sigma = \nu = 0.1$), we have found that the improvements given by SimHeur versus DetHeur are significant, and these improvements would likely be greater without the severe time compression used in our experiments (i.e. one minute of CPU time to represent one hour of real time). It should also be noted that the advantage of SimHeur over DetHeur tends to become greater when the on-time requirements of individual flights are given more weight in the objective function.

Certainly, our computational study could be expanded in order to consider other schedules, alternative heuristic search methods and changes to the objectives and model parameters, and this should be a direction of further research. We also plan to consider models with both arrivals and departures in future work, and to consider decision-making problems featuring further complexities such as assignments of flights to different runways.

Acknowledgments

This work has been supported by the Engineering and Physical Sciences Research Council (EPSRC) through Programme Grant EP/M020258/1 "Mathematical models and algorithms for allocating scarce airport resources (OR-MASTER)". We would like to thank Flightradar24 (https://www.flightradar24.com/) for granting permission for their on-time performance data to be used in our study. We are also grateful to the editors and reviewers for their comments and suggestions, which have helped to improve this paper.

References

Badrinath, S., Balakrishnan, H., Joback, E., and Reynolds, T. (2020). Impact of Off-Block Time Uncertainty on the Control of Airport Surface Operations. *Transportation Science*, 54(4):920–943.

Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., Peterson, E., Sherry, L., Trani, A., and Zou, B. (2010). *Total delay impact study: A comprehensive assessment of the costs and impacts of flight delay in the United States.* Technical report, Federal Aviation Administration, Washington, DC.

Beasley, J., Krishnamoorthy, M., Sharaiha, Y., and Abramson, D. (2000). Scheduling Aircraft Landings - The Static Case. *Transportation Science*, 34(2):180–197.

Beasley, J., Krishnamoorthy, M., Sharaiha, Y., and Abramson, D. (2004). Displacement Problem and Dynamically Scheduling Aircraft Landings. *Journal of the Operational Research Society*, 55(1):54–64.

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

Bennell, J., Mesgarpour, M., and Potts, C. (2017). Dynamic scheduling of aircraft landings. *European Journal of Operational Research*, 258:315–327.

Birge, J. and Louveaux, F. (2011). *Introduction to Stochastic Programming, 2nd edition.* Springer.

Cavusoglu, S. and Macario, R. (2021). Minimum delay or maximum efficiency? Rising productivity of available capacity at airports: Review of current practice and future needs. *Journal of Air Transport Management*, 90:101947.

de Neufville, R. and Odoni, A. (2013). *Airport Systems: Planning, Design and Management, 2nd edition.* McGraw-Hill.

Errico, A. and Di Vito, V. (2019). *Aircraft operating technique for efficient sequencing arrival enabling environmental benefits through CDO in TMA.* Proc. AIAA Scitech 2019 Forum, San Diego, CA.

Eurocontrol (2022). All-causes delays to air transport in europe annual 2021. `https://www.eurocontrol.int/publication/all-causes-delay-and-cancellations-air-transport-europe-2021`. Accessed on May 1, 2022.

Eurocontrol (2023). European Aviation Overview: July 4 2023. `https://www.eurocontrol.int/sites/default/files/2023-07/eurocontrol-european-aviation-overview-20230704.pdf`. Accessed on July 24, 2023.

Folks, J. and Chhikara, R. (1978). The Inverse Gaussian Distribution and its Statistical Application - A Review. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40(3):263–275.

Gilbo, E. (1993). Airport capacity: Representation, estimation, optimization. *IEEE Transactions on Control Systems Technology*, 1(3):144–154.

Hansen, M., Nikoleris, T., Lovell, D., Vlachou, K., and Odoni, A. (2009). *Use of Queueing Models to Estimate Delay Savings from 4D Trajectory Precision.* Proc. 8th USA/Europe Air Traffic Management R&D Seminar, Napa, CA.

Heidt, A., Helmke, H., Kapolke, M., Liers, F., and Martin, A. (2016). Robust runway scheduling under uncertain conditions. *Journal of Air Transport Management*, 56:28–37.

Jacquillat, A. and Odoni, A. (2015). An Integrated Scheduling and Operations Approach to Airport Congestion Mitigation. *Operations Research*, 63(6):1390–1410.

Jacquillat, A. and Odoni, A. (2018). A roadmap toward airport demand and capacity management. *Transportation Research Part A*, 114:168–185.

Jacquillat, A., Odoni, A., and Webster, M. (2017). Dynamic Control of Runway Configurations and of Arrival and Service Departure Rates at JFK Airport Under Stochastic Queue Conditions. *Transportation Science*, 51(1):155–176.

Juan, A., Faulin, J., Grasman, S., Rabe, M., and Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72.

Khassiba, A., Bastin, F., Cafieri, S., Gendron, B., and Mongeau, M. (2020). Two-Stage Stochastic Mixed-Integer Programming with Chance Constraints for Extended Aircraft Arrival Management. *Transportation Science*, 54(4):897–919.

Koopman, B. (1972). Air-Terminal Queues under Time-Dependent Conditions. *Operations Research*, 20(6):1089–1114.

Liu, M., Liang, B., Zheng, F., Chu, C., and Chu, F. (2018). *A Two-stage Stochastic Programming Approach for Aircraft Landing Problem.* Proc. 2018 International Conference on Service Systems and Service Management (ICSSSM), Hangzhou, China.

Liu, M., Liang, B., Zhu, M., and Chu, C. (2020). Stochastic Runway Scheduling Problem With Partial Distribution Information of Random Parameters. *IEEE Access*, 8:68460–68473.

Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Montlaur, A. and Delgado, L. (2017). Flight and passenger delay assignment optimization strategies. *Transportation Research Part C*, 81:99–117.

Moser, I. and Hendtlass, T. (2007). *Solving dynamic single-runway aircraft landing problems with extremal optimization.* Proc. IEEE symposium on computational intelligence in scheduling, Honolulu, HI.

Murca, M. and Muller, C. (2015). Control-based optimization approach for aircraft scheduling in a terminal area with alternative arrival routes. *Transportation Research Part E*, 73:96–113.

NATS Ltd (2023a). Demand Capacity Balancer. `https://www.nats.aero/services-products/products/n/demand-capacity-balancer-dcb/`. Accessed on July 24, 2023.

NATS Ltd (2023b). Strategic ACM. `https://www.nats.aero/services-products/products/n/strategic-acm/`. Accessed on July 24, 2023.

Nelson, B. (2013). *Foundations and methods of stochastic simulation: a first course.* Springer Science & Business Media.

Newell, G. (1979). Airport Capacity and Delays. *Transportation Science*, 13(3):201–241.

Odoni, A., Morisset, T., Drotleff, W., and Zock, A. (2011). *Benchmarking Airport Airside Performance: FRA vs. EWR.* Proc. 9th USA/Europe Air Traffic Management R&D Seminar, Berlin, Germany.

Psaraftis, H. (1978). *A Dynamic Programming Approach to the Aircraft Sequencing Problem.* Technical report R78-4, MIT Flight Transportation Laboratory.

Pyrgiotis, N. and Odoni, A. (2016). On the Impact of Scheduling Limits: A Case Study at Newark Liberty International Airport. *Transportation Science*, 50(1):150–165.

Sama, M., D'Ariano, A., and Pacciarelli, D. (2013). Rolling Horizon Approach for Aircraft Scheduling in the Terminal Control Area of Busy Airports. *Procedia - Social and Behavioral Sciences*, 80:531–552.

SESAR (2023a). Arrival Sequencing Benefits from a Common Service Approach. URL: https://www.sesarju.eu/sesar-solutions/e-aman-common-service. Accessed August 08 2023.

SESAR (2023b). Extended Arrival Management (AMAN) horizon. URL: https://www.sesarju.eu/sesar-solutions/extended-arrival-management-aman-horizon. Accessed August 08 2023.

Shone, R., Glazebrook, K., and Zografos, K. (2019). Resource allocation in congested queueing systems with time-varying demand: An application to airport operations. *European Journal of Operational Research*, 276(2):566–581.

Shone, R., Glazebrook, K., and Zografos, K. (2021). Applications of stochastic modeling in air traffic management: Methods, challenges and opportunities for solving air traffic problems under uncertainty. *European Journal of Operational Research*, 292(1):1–26.

Solak, S., Solveling, G., Clarke, J.-P., and Johnson, E. (2018). Stochastic Runway Scheduling. *Transportation Science*, 52(4):917–940.

Solveling, G., Solak, S., Clarke, J.-P., and Johnson, E. (2011). Runway operations optimization in the presence of uncertainties. *Journal of Guidance, Control and Dynamics*, 34(5):1373–1382.

Stamatopoulos, M., Zografos, K., and Odoni, A. (2004). A decision support system for airport strategic planning. *Transportation Research Part C*, 12:91–117.

Tielrooij, M., Borst, C., van Paassen, M., and Mulder, M. (2015). *Predicting Arrival Time Uncertainty from Actual Flight Information.* Proc. 11th USA/Europe Air Traffic Management R&D Seminar, Lisbon, Portugal.

UK Aeronautical Information Services (2019). Aeronautical Information Circular P 092/2017. `https://www.skybrary.aero/bookshelf/books/1166.pdf`. Accessed on July 31, 2019.

Zografos, K., Madas, M., and Androutsopoulos, K. (2017). Increasing airport capacity utilisation through optimum slot scheduling: review of current developments and identification of future needs. *Journal of Scheduling*, 20(1):3–24.

Appendix A: Details of the SimHeur algorithm

In this appendix we provide detailed descriptions of the steps of the SimHeur algorithm. Firstly, let the flights in $\mathcal{F}$ be indexed in ascending order by their $a_i + \Delta_i^{\text{pre}}$ values, which are assumed known at the beginning of $\mathcal{T}$. Hence, our initial belief is that flight 1 will be the first to arrive in the terminal area and flight $F := |\mathcal{F}|$ will be the last.

Step 1: Initialization

We initialize a set (or 'population') $\mathcal{S}_0$ of solutions (sequences), where each sequence $s \in \mathcal{S}_0$ is a tuple of length $l \in \mathbb{N}$ and specifies the next l flights that will land on the runway, in order, if this sequence is followed. It will be convenient to let $\mathcal{S}_t$ denote the population at time $t \in \mathcal{T}$. The number of sequences in $\mathcal{S}_0$ is denoted by $S \in \mathbb{N}$ and it will also be the case that $|\mathcal{S}_t| \leq S$ for all $t \in \mathcal{T}$. Let $S_{\min} \leq S$ be an integer that represents the minimum allowable population size. This means that if $|\mathcal{S}_t| < S_{\min}$ at some time point t then we must 'repopulate' $\mathcal{S}_t$ by generating new sequences until its size is restored to S. We also initialize an iteration counter, $n = 0$, and an additional counter $m = 0$. The parameters S, $S_{\min}$ and l remain constant until the final stages of SimHeur's running time, at which point it becomes necessary to reduce their values in order to ensure that it is still feasible to find S distinct l-tuples consisting of only the flights in $\mathcal{F}$ that haven't already been released. Indeed, after all flights in $\mathcal{F}$ have been released it is necessary to set $S = S_{\min} = l = 0$, at which point steps 2A-2C and 4A-4C in Figure 3 become redundant, but the algorithm continues to run until all flights in the queue have completed service.

In our numerical experiments in Section 5, we generate the initial sequences $s \in \mathcal{S}_0$ by making random changes to a 'first-come-first-served' sequence, as follows:

(a) Let $\mathcal{S}_0$ consist of only one sequence, $(1, 2, ..., l)$.

(b) Make a change to the sequence $(1, 2, ..., l)$ by applying a heuristic move operator, denoted H. The steps of this heuristic move operator are as follows:

  (i) Let $L := l(l+1)/2$. We randomly select an integer $j \in \{1, 2, ..., l\}$ in the following way: the probability of selecting 1 is l/L, the probability of selecting 2 is $(l-1)/L$, and in general, the probability of selecting p is $(l+1-p)/L$ for $1 \leq p \leq l$.

  (ii) The flight in position j of s is to be shifted by a certain number of positions either forwards or backwards. First, decide on the direction of movement using a simple 'coin flip', so that the 'forwards' and 'backwards' directions are selected with probability 0.5 each.

  (iii) Let P be sampled uniformly at random from the set $\{1, 2, 3\}$. The flight in position j of s is removed from s and then re-inserted at position $\max\{j - P,\ 1\}$ (if the 'forwards' direction was selected in substep (ii)) or position $\min\{j+P,\, l\}$ (if the 'backwards' direction was selected).

If the new sequence is not already in $\mathcal{S}_0$, then add it. Otherwise, repeat this substep.

(c) If $|\mathcal{S}_0| = S$, terminate. Otherwise, return to substep (b).

Step 2A: Simulation and Evaluation

In this step we update performance estimates of all sequences in our current population $\mathcal{S}_t$. For each flight i in a particular sequence $s \in \mathcal{S}_t$ we must estimate its contribution to the objective function (13) given that s is followed. We do this by randomly sampling a sequence of events (referred to as a *sample path* or *sample trajectory*), denoted by Ω_t. The sample path Ω_t includes realizations, denoted by $\tilde{Q}_i$, of pool arrival times for all flights $i \in \mathcal{F}$ yet to arrive in the pool (i.e. $t < Q_i$), and also realizations of times needed for the remaining part of the journey for flights that are either still in the pool (i.e. $Q_i < t < R_i$) or still enroute to the runway (i.e. $R_i < t < A_i$). It also includes realizations $\tilde{M}_{ij}$ of the separation times for all consecutive pairs (i,j) of aircraft in the queue, and (if relevant) realizations $\tilde{U}_0$ and $\tilde{U}_1$ for the starting and ending times of any future period of bad weather. We assume that all of these values are sampled from the correct distributions described in Section 2. For example, following the details in Section 2, the time remaining until flight i arrives in the pool $(Q_i - t)$ has density function

$$f_{Q_i - t}(u \mid X_i(t)) = \frac{X_i(t) - \tau - t}{\sigma_i \sqrt{2\pi u^3}} \exp\left(-\frac{(X_i(t) - \tau - t - u)^2}{2\sigma_i^2 u}\right),$$

which can be integrated numerically in order to allow a quantile to be sampled according to a uniform $[0,1)$ distribution (the 'inverse transform sampling' method). A similar method can also be used for sampling the remaining travel times for flights that have already arrived in the pool or been released from the pool; for example if flight i is still in the pool, its remaining travel time is sampled from an Inverse Gaussian distribution with mean τ, variance $\sigma_i^2 \tau$. If a service is in progress, then (as described in Section 2.4) the elapsed service time is known and we sample the remaining service time from a conditional gamma distribution.

After Ω_t has been generated, the future release times, landing times etc. of the flights in a particular sequence $s \in \mathcal{S}_t$ can be worked out in a deterministic way based on the generated timings of events, and this yields an overall cost estimate for sequence s. For clarity, we assume here (only for the purposes of estimating costs under different possible sequences) that if a particular sequence $s \in \mathcal{S}_t$ is followed starting from time t, then the following procedure is used:

(a) If the first flight in s is already in the pool, then it is released immediately. Otherwise, we wait until it arrives in the pool, without releasing any other flights in the meantime, and then release it immediately.

(b) Substep (a) is repeated for the next sequential flight in s, and this process is repeated until all flights in s have been released.

It should be noted that the cost estimate for s obtained by the above procedure is based only on the l flights that are included in sequence s; it does not take into account the pool holding times and landing times of any subsequent flights, even though these would be affected by the sequencing decisions for flights in s. In this respect, our cost estimates are only based on 'looking ahead' by a limited amount of time into the future, and it is important to avoid making l too small in order to avoid being too myopic. On the other hand, larger values of l are associated with too much computational expense and compromise the performance of SimHeur. In practice, the values of l that we use are sufficiently small to ensure that the sequences $s \in \mathcal{S}_t$ consist only of flights that are already in the 'tactical uncertainty' stage (i.e. $t > h_i$ for flights $i \in s$) and, hence, we do not consider sequencing options for flights that are yet to realize their pre-tactical uncertainty. This explains why there is no loss of generality in assuming that Δ_i^{pre} is realized for all $i \in \mathcal{F}$ at the beginning of $\mathcal{T}$ and scaling the time units accordingly.

During the running of SimHeur we revisit step 2A many times and acquire cost estimates for each population member $s \in \mathcal{S}_t$ at many different time points (and using many different sample paths). Let $(t_1, t_2, ...)$ denote the sequence of time epochs at which these estimates are obtained, with $(\Omega_{t_1}, \Omega_{t_2}, ..)$ being the corresponding sequence of sample paths generated. Let $J_s^{(n)}$ denote the nth cost estimate for sequence s, given by sample path Ω_{t_n}. We note here that the iteration counter n is reset to zero in some later steps of the algorithm when new sequences are added (see steps 4A-4C) and our notation assumes that any sequence included in the population at epoch t_n is also included at epochs t_j for $j < n$, i.e. $\mathcal{S}_{t_n} \subseteq \mathcal{S}_{t_{n-1}} \subseteq ... \subseteq \mathcal{S}_{t_1}$. At time t_n, we update two overall performance indicators of sequence $s \in \mathcal{S}_{t_n}$, denoted $V_s^{(n)}$ and $W_s^{(n)}$ as shown in equations (14)-(15) (these are both initialized at zero for $n = 0$).

There is one further performance measure that we update in this step. Consider the flights that are already in the queue at time t_n and suppose flight $i \in \mathcal{F}$ is the last flight in the queue, i.e. the most recent flight to have been released from the pool. Also, let $\tilde{L}_i^{(n)}$ denote the actual landing time for flight i under the sample path Ω_{t_n}. For each sequence $s \in \mathcal{S}_t$, let j_s be the first sequential flight in s and define the binary variable $\xi_s^{(n)}$ as follows:

$$\xi_s^{(n)} := \begin{cases} 1, & \text{if } \tilde{A}_{j_s}^{(n)} > \tilde{L}_i^{(n)} + \tilde{M}_{i,j_s}^{(n)}, \\ 0, & \text{otherwise,} \end{cases}$$

where $\tilde{A}_{j_s}^{(n)}$ and $\tilde{M}_{i,j_s}^{(n)}$ are (respectively) the unconstrained landing time for flight j_s and the landing time separation between i and j_s under sample path Ω_{t_n}. Hence, if $\xi_s^{(n)} = 1$, this indicates that flight j_s does not arrive early enough to be able to land at the earliest possible moment after flight i's landing. One might say that there is some 'idle runway time' caused by the late arrival of flight j_s. We then define $\Upsilon_s^{(n)}$ as a (possibly weighted) average of $\xi_s^{(n)}$ over all sample paths, as follows:

$$\Upsilon_s^{(n)} := \begin{cases} 0, & \text{if } n = 0, \\ (1-\psi_n)\Upsilon_s^{(n-1)} + \psi_n \xi_s^{(n)}, & \text{otherwise.} \end{cases}$$

If $\Upsilon_s^{(n)}$ happens to be very small, then this suggests that there is little benefit in releasing flight j_s from the pool immediately, as it is likely to be forced to wait in the queue and will have to wait until time $L_i + M_{i,j_s}$ before it is able to land. It may be advantageous to delay its release so that we can acquire more information (from system state updates) before making a final decision about which flight to release next from the pool. We elaborate on this further in step 2C.

Step 2B: Ranking and Selection

After n cost evaluations have been performed we can rank the sequences in $\mathcal{S}_{t_n}$ according to their $V_s^{(n)}$ values and remove any sequences that fail to satisfy the criterion (16). We note that this step is not actually performed at each iteration $n \in \mathbb{N}$ in our algorithm; instead, as shown in Figure 3, it is only performed when n is a multiple of r, for some pre-determined $r \in \mathbb{N}$. The reason for this is that the ranking and selection process involves pairwise comparisons between all sequences in our current population, and this can be computationally expensive, so there is little value in performing this step at every iteration given that the differences in $V_s^{(n)}$ and $W_s^{(n)}$ values on consecutive iterations are likely to be small.

We also note that, since $Z_{s,s'}$ (defined in (17)) is positive, the sequence with the smallest sample mean is guaranteed to be retained in the population.

Step 2C: Release of Flights from Pool

Let s^* denote the sequence in our current population $\mathcal{S}_t$ with the smallest value of $V_s^{(n)}$ after n cost evaluations have been performed. If $n \geq n_{\text{rel}}$, where $n_{\text{rel}} \in \mathbb{N}$ is a pre-determined threshold, then we check to see whether $\Upsilon_{s^*}^{(n)}$ exceeds another pre-determined value $\lambda \in [0,1)$. If the additional condition $\Upsilon_{s^*}^{(n)} > \lambda$ holds, then it is decided that the flights at the front of sequence s^* should be released as soon as possible if they are already in the pool.

Specifically, suppose the conditions $n \geq n_{\text{rel}}$ and $\Upsilon_{s^*}^{(n)} > \lambda$ hold and let

$$u := \max\{j \in \mathbb{N} : Q_i \leq t \text{ for all flights } i \text{ in the first } j \text{ positions of } s^*\}.$$

In other words, u is the number of positions in sequence s^* that we are able to count, starting from the beginning, without getting to an aircraft that isn't in the pool yet. If $u \geq 1$, then we should release these u aircraft as soon as possible, so that they join the queue in the same order that they appear in s^*. As shown in Figure 3, these u aircraft are only 'marked' for release at this stage. They are actually released in step 3, following the next system state update, in order to ensure that the state information (including the weather state, for example) at their time of release is accurate. On the other hand, if $u = 0$, then no aircraft should be released.

We note that, as with several other parameters in our algorithm, setting the value of n_{rel} involves a 'trade-off'. Larger values enable us to be more confident that the sequence s^* is genuinely the best sequence available due to the greater number of cost evaluations performed, but by requiring n to be large before any flights are released, we might delay their release for too long and incur greater costs as a result. Furthermore, as explained in step 2A, the condition $\Upsilon_{s^*}^{(n)} > \lambda$ is designed to ensure that we can derive some benefit from delaying the release of a flight from the pool in situations where the flight is likely to be delayed in the queue anyway (and therefore an early release would not imply an earlier landing time). The benefit of delaying the release is that we are able to acquire more information (through system state updates) before deciding which flight should be committed to the queue next. However, our numerical experiments indicate that λ should be set to a very small value in order to give the best results, and indeed $\lambda = 0$ is often the best choice. The reason for this is that if any unnecessary 'idle runway time' occurs (i.e. a flight arrives at the runway later than its earliest feasible landing time based on time separations), this can cause delays to many subsequent flights, implying a very significant increase in the value of the objective function (13). Therefore, if there is even the slightest possibility of idle runway time occurring, we may wish to release the next flight as soon as possible. Setting $\lambda = 0$ effectively implies that we only need to find one random sample path (among possibly thousands) with unnecessary idle runway time in order for the condition $\Upsilon_{s^*}^{(n)} > \lambda$ to be met.

Step 3: System State Update

All of the steps in our algorithm require some computational effort. In this step we move the simulation 'clock' forward according to the amount of time elapsed since the previous system state update (or since initialization, if this step is being encountered for the first time) and update the latest system state information. Specifically, if δt is the amount of time elapsed since the previous update, then the current time t should be incremented by an amount proportional to δt. It is then necessary to update the ETAs $X_i(t)$ for all flights $i \in \mathcal{F}$ such that $t < A_i$. In addition, we need to check whether any service phase completions have occurred during this time increment and (if necessary) also update the weather forecast and the current weather state.

In our computer implementation, we have found that the most efficient approach is to pre-generate all random events and their timings, so that we have a pre-generated 'actual' sample path

Ω^{true} (hidden from the decision-maker). Then, in order to update the system state at a new time point, we only need to 'look up' the relevant information in Ω^{true} rather than sampling from any distributions again. Further details about how this is done can be found in Appendix E.

This step also involves releasing any flights that have been marked for release (see step 2C), so that new flights are added to the queue if necessary. Also, as noted in step 1, we may need to reduce the values of S, $S_{\min}$ and l in this step if there are only a few flights remaining that haven't already been released from the pool.

Step 4A: Repopulation (Type 1)

As shown in Figure 3, this step follows step 3 in the case where at least one flight has just been released from the pool. In this case, the newly-released flights are no longer eligible to be included in the sequences in our population (as we only consider sequencing decisions for flights that haven't been released yet), so we perform a 'reset' by removing all sequences from our current population, erasing the information $\{V_s^{(n)}, W_s^{(n)}, \Upsilon_s^{(n)}\}$ for all removed sequences and setting $n = 0$. We then create a new population $\mathcal{S}_t$ by performing the following substeps:

(a) Form a new sequence b consisting of the flights in positions $u+1, u+2, ..., |s^*|$ of the sequence s^* that was chosen as the best sequence in the previous population. (Note that u represents the number of flights that have just been released, as defined in step 2C.)

(b) Consider all flights i that have not yet been added to the queue and are not already included in b. Among these flights, select the one with the earliest ETA $X_i(t)$ and append it to the end of b.

(c) Repeat substep (b) as many times as necessary until the number of flights in b is l.

(d) Generate an extra $S - 1$ sequences, where each new sequence is formed by applying the heuristic move operator H described in step 1 to the sequence b formed above, in order to obtain a new population of S sequences.

After the new population $\mathcal{S}_t$ has been created, the algorithm returns to step 2A (simulation and evaluation).

Step 4B: Filter Population

Following the latest system state update (step 3), it may be beneficial to add new sequences to the population $\mathcal{S}_t$ and evaluate these (in step 2A) according to the latest state information. In order to do this, we need to 'make room' for the new sequences by removing some of the weaker sequences from $\mathcal{S}_t$. This step is only performed if $n \geq n_{\text{repop}}$, where $n_{\text{repop}} \in \mathbb{N}$ is a pre-determined threshold, because we must have performed a sufficient number of iterations to be able to reliably judge which sequences in $\mathcal{S}_t$ are the weakest.

In this step we simply rank the sequences in $\mathcal{S}_t$ according to their sample means $V_s^{(n)}$ (comparable to the 'fitness' estimates used in metaheuristic algorithms) and remove the $(S - S_{\min})$ sequences with the highest values, so that the new population size is $S_{\min}$. If $S_{\min} = 1$ then we only retain the 'best' sequence in the current population, but there are some potential advantages in choosing a larger $S_{\min}$ value: (i) the sample means $V_s^{(n)}$ are associated with sampling error and the ranking order of the sequences may change as we repeat step 2A more times; (ii) even if there was no sampling error, the system state is continuously evolving and new information might imply that the ranking order should be changed. Note that sequences can be removed in this step even if they satisfy the 'ranking and selection' criterion (16) in step 2B.

After this step, we then add new sequences to the population in step 4C.

Step 4C: Repopulation (Type 2)

As shown in Figure 3, this step can be reached in two different ways. If $n \geq n_{\text{repop}}$, then we reduce the population size to $S_{\min}$ as described in step 4B before arriving at this step. On the other hand, we might also reach this step if the population size has been reduced to $S_{\min}$ (or smaller) following the 'ranking and selection' process in step 2B. In either case, we can assume that the current population size $|\mathcal{S}_t|$ is not greater than $S_{\min}$.

The purpose of this step is to add new sequences to the population until its size is restored to S. Recall that m is a counter, initialized with a value of zero in step 1. We assume that $m_{\text{mut}} \in \mathbb{N}$ is a pre-determined threshold which determines the point at which we should 'mutate' the best sequence in our current population. The substeps are described below.

(a) Let s^* denote the sequence in the current population $\mathcal{S}_t$ with the smallest sample mean $V_s^{(n)}$.
(b) If s^* was already included in the population when we last entered step 4C, then increase m by 1. Otherwise, set $m = 0$.
(c) If $m \geq m_{\text{mut}}$, create a new sequence b by applying a random mutation to s^* and then set $m = 0$. Otherwise, set $b = s^*$.
(d) Make a change to the sequence b by applying a heuristic move operator, denoted H. If the new sequence is not already in $\mathcal{S}_t$, then add it. Otherwise, repeat this step.
(e) If $|\mathcal{S}_t| = S$, terminate. Otherwise, return to step (d).

The heuristic move operator H is the same one used in steps 1 and 4A (see earlier). The 'random mutation' referred to in substep (c) above works as follows:

(i) Let $M(t)$ denote the number of flights in $\mathcal{F}$ that have not yet been added to the landing queue at time t; that is, $M(t)$ consists of flights that are either still in the pool or yet to arrive in the pool.

(ii) Let $P(t) := \min\{4,\, Q(t)\}$, where $Q(t) := \min\{l,\, M(t)\}$. Here, $P(t)$ is interpreted as the length of a particular subsequence within s that we want to 'shuffle' in order to obtain a new sequence.

(iii) Define $R(t) := Q(t) - P(t) + 1$ as the number of possible starting positions of the string that we are going to shuffle.

(iv) Let $L := R(t)(R(t)+1)/2$. We randomly select an integer $j \in \{1, 2, ..., R(t)\}$ in the following way: the probability of selecting 1 is $R(t)/L$, the probability of selecting 2 is $(R(t)-1)/L$, and in general, the probability of selecting p is $(R(t)+1-p)/L$ for $1 \le p \le R(t)$.

(v) Consider the subsequence formed by taking the flights in positions $j,\, j+1, ..., j+P(t)-1$ of s. Remove all flights in this subsequence from s, then 'shuffle' the subsequence, i.e. choose a random permutation of it. Finally, re-insert the shuffled subsequence in the same position within s that it occupied before. We thus obtain a new sequence, interpreted as a 'mutation' of s.

We note that the purpose of the mutation is to force the algorithm to explore a different part of the solution space, and this is consistent with the VNS methodology. The condition $m \ge m_{\text{mut}}$ indicates that the algorithm has been through the repopulation process m times without successfully finding a new sequence that performs better than s^*. This suggests that a local optimum has been found, and hence we should migrate to another region of the solution space.

We also reset n to zero in this step and erase the information $\{V_s^{(n)}, W_s^{(n)}, \Upsilon_s^{(n)}\}$ for sequences that have been retained from our previous population. Following this step, the algorithm returns to step 2A (simulation and evaluation).

We have now described all steps in the SimHeur algorithm. As shown in Figure 3, the algorithm terminates when all flights in $\mathcal{F}$ have completed service, at which point we obtain a value for the objective function (13).

Appendix B: Details of the DetHeur algorithm

The steps of the DetHeur algorithm are similar to those described in Appendix A for SimHeur, except for the following changes:

• In step 2A, we generate the sample path $\mathcal{S}_t$ by setting $\tilde{Q}_i = X_i(t) - \tau$ for all flights i that haven't arrived in the pool yet (i.e. $t < Q_i$). After i has been released (i.e. for $t \ge R_i$) we use

$\tilde{A}_i = X_i(t)$ as a deterministic prediction of its runway arrival time. We also set $\tilde{M}_{ij} = e_{ij}$ for all consecutive pairs of flights (i,j) in a particular sequence and set $\tilde{U}_0 = T_0(t)$ and $\tilde{U}_1 = T_1(t)$ if there is a future period of bad weather expected. The sample mean $V_s^{(n)}$ is replaced by the single cost evaluation $J_s^{(n)}$, so that any previous cost evaluations for sequence s are discarded. We do not require $W_s^{(n)}$ or $\Upsilon_s^{(n)}$.

- Step 2B (ranking and selection) is omitted.
- In step 2C, the condition $n \geq n_{\text{rel}}$ is no longer required (equivalently, we might say that $n_{\text{rel}} = 1$). The condition $\Upsilon_s^{(n)} > \lambda$ is also no longer applicable.
- The condition $n \geq n_{\text{repop}}$ is no longer required in order to enter step 4B. Equivalently, we might say that $n_{\text{repop}} = 1$.

We note that the removal of the condition $\Upsilon_s^{(n)} > \lambda$ implies that DetHeur takes a somewhat conservative approach by releasing the first flight in s^* as soon as it arrives in the pool, rather than delaying its release in order to acquire more information. Indeed, since DetHeur does not generate random sample paths, it cannot estimate the probability of idle runway time in the same way as SimHeur. One might argue, however, that it should release the first flight (say flight j) from s^* if and only if the runway arrival time $\tilde{A}_j = t + \tau$ under the expected value trajectory at time t satisfies $\tilde{A}_j \geq \tilde{L}_i + \tilde{M}_{ij}$, where i is the latest flight to be released; in other words, the flight should not be released if we expect it to be delayed in the queue. This seems a reasonable suggestion, but it is easy to show using experiments that DetHeur performs extremely poorly under such a rule. As noted earlier, any unnecessary idle runway time tends to increase the objective function value very significantly due to the 'knock-on' effect of one delayed landing causing another. If we release flight j at the point where $\tilde{A}_j = \tilde{L}_i + \tilde{M}_{ij}$ then there is roughly a 50% chance that unnecessary idle runway time will occur, and this must be avoided.

Appendix C: Obtaining the DStat policy

The heuristic method for obtaining the DStat policy referred to in Section 3.3 is as follows:

1. We begin with a sequence s_0 of length $|\mathcal{F}|$ in which the flights are ordered according to their ETAs following the realization of pre-tactical uncertainty; that is, if flight $i \in \mathcal{F}$ appears before flight $j \in \mathcal{F}$ then this implies $a_i + \Delta_i^{\text{pre}} \leq a_j + \Delta_j^{\text{pre}}$.
2. Set $s^{\text{best}} := s_0$ and initialize a counter $c = 0$.
3. Set the pool arrival time to $Q_i = a_i + \Delta_i^{\text{pre}} - \tau$ for all $i \in \mathcal{F}$, set $U_0 = t_0$, $U_1 = t_1$ and assume that all separation times M_{ij} conform to their expected values $E_{ij}(R_j)$ (defined in (10)). For the release times R_j, we assume that each flight i in s^{best} is released from the pool at the earliest

possible moment, with the restriction that flights must be released in the same order that they appear in s^{best}. For example, if the 2nd flight in s^{best} happens to arrive in the pool earlier than the 1st flight, then the 2nd flight should be released immediately after the 1st flight arrives in (and is released from) the pool. We also define $A_j = R_j + \tau$ as the unconstrained arrival time of flight $j \in \mathcal{F}$, with the actual landing time given by $L_j = \max\{A_j, L_i + M_{ij}\}$, where i is the predecessor of j in the queue. We then calculate the value of the objective function (13) if the sequence s^{best} is followed and let this be denoted by C^{best}.

4. Select a subsequence of 6 flights appearing consecutively in s^{best} by sampling uniformly at random from the $|\mathcal{F}|-5$ possible alternatives (corresponding to starting positions $1, 2, ..., |\mathcal{F}|-5$ within s^{best}).
5. The subsequence selected in step 4 is removed from s^{best} and the flights in this subsequence are then 'shuffled'; i.e. a random permutation of the subsequence is chosen. The shuffled subsequence is then re-inserted into s^{best} in the same position that it occupied before. This yields a new sequence that we refer to as s^{curr}.
6. Calculate the value of the objective function (13) under the new sequence s^{curr} by setting the values of Q_i, U_0, U_1, M_{ij} to their expected values and calculating R_i, A_i as described in step 3. Let C^{curr} denote the objective function value under s^{curr}.
7. If $C^{\text{curr}} < C^{\text{best}}$, then set $C^{\text{best}} := C^{\text{curr}}$ and $s^{\text{best}} := s^{\text{curr}}$, and set $c = 0$. Otherwise, increase the counter c by 1.
8. If $c = 10,000$, then terminate the procedure and accept s^{best} as the DStat policy. Otherwise, return to step 4.

Note that the final value of C^{best} associated with the sequence s^{best} is not a measure of DStat's actual performance, as C^{best} is based on the assumption of all random variables conforming to their expected values. To evaluate DStat's actual performance, we must use the 'real' information contained in Ω^{true}, as explained in Section 3.3.

Appendix D: Filtering the historical dataset

This appendix describes our method of filtering the historical dataset in order to improve approximations for the distributions of unconstrained arrival times. Firstly, for each of the 360 days in our historical period we examine the sequence of actual landings that took place during the 6AM-2PM interval and separate the flights that landed into two sets. Set S_1 (resp. S_2) consists of flights that landed immediately after a flight that landed earlier (resp. later) than its scheduled landing time. We then compare the proportions of flights landing earlier than their scheduled times in sets S_1

and S_2. If the difference in these two proportions is statistically significant at the 5% level, then we discard all data from the whole of that particular day when fitting the distributions for the A_i. The rationale for this approach is as follows: if the queueing delays on a particular day are significant, then delays are likely to propagate, in the sense that the late arrival of one flight causes the late arrival of another. Therefore if we restrict attention to days on which the timeliness of one landing tends to be independent of the timeliness of its immediate predecessor, we can be more confident that the recorded landing times of flights are similar to their unconstrained landing times.

After carrying out the procedure described above, we identified 112 days (out of 360) on which the differences between proportions of late-arriving flights in sets S_1 and S_2 were significant. We removed all of these 112 days from our dataset and were left with 248 days' worth of data, with 199.5 records per flight on average. The box plot in Figure 6 shows a comparison between the distributions of average delay over all flights in $\mathcal{F}$ for the 248 retained days and the 112 removed days. (Negative delays occur when actual landing times are earlier than scheduled times.) All of the days with average delays greater than 10.38 minutes were among the 112 flights removed from our dataset, suggesting that our method is somewhat effective in filtering out the days with abnormally long delays that may be attributable to airport congestion.

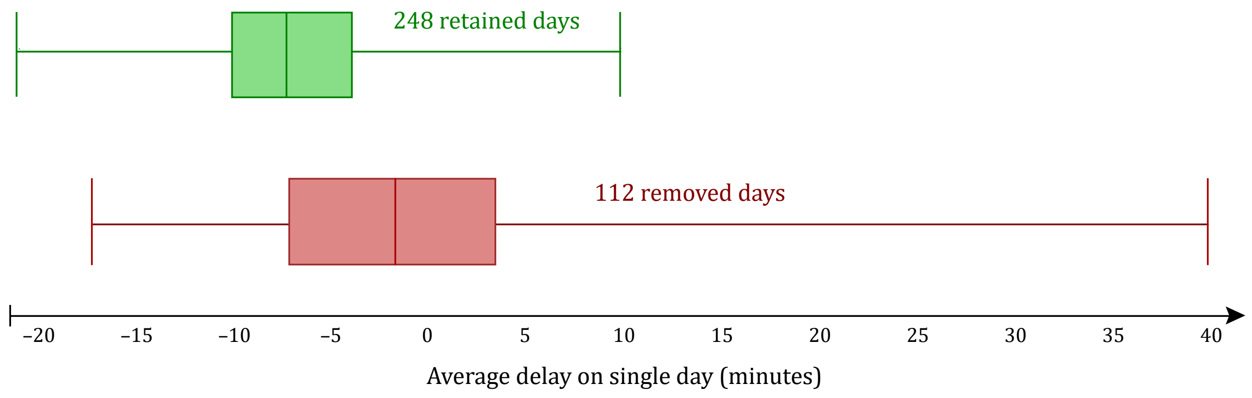

Figure 6 Distributions of average delay on a single day for the 248 retained days and 112 removed days.

Appendix E: The method of pre-generating random events

The pre-generated sample path Ω^{true} includes complete trajectories $\{X_i(t)\}_{t\in\mathcal{T}}$ for $i\in\mathcal{F}$, weather forecast trajectories $\{T_0(t)\}_{t\in\mathcal{T}}$ and $\{T_1(t)\}_{t\in\mathcal{T}}$, and a set of possible separation times $\{M_{ij}\}$ for each $j\in\mathcal{F}$ (taking into account all possibilities for the weight class of the predecessor flight i). The steps for generating the trajectories $\{X_i(t)\}_{t\in\mathcal{T}}$ are described below.

1. For each $i\in\mathcal{F}$ we set the initial ETA, $X_i(0)$, as follows:

$$X_i(0)=\begin{cases} a_i+\Delta_i^{\text{pre}}+\mathrm{N}(0,\,\sigma_i^2|h_i|), & \text{if } h_i<0,\\ a_i+\Delta_i^{\text{pre}}, & \text{if } h_i\geq 0,\end{cases}$$

where $\mathrm{N}(a,b)$ denotes a Normal random variable with mean a and variance b.

2. We step forward in hundredths of a minute and, for each $t = 0.01,\ 0.02,\ ..., T$, generate the value $X_i(t)$ using a standard method for simulating Brownian motion:

$$X_i(t) := X_i(t-0.01) + \mathrm{N}(0,\ 0.01 \times \sigma_i^2).$$

We also define $X_i(u) := X_i(t-0.01)$ for $u \in [t-0.01, t)$.

The above method ensures that $X_i(t)$ is defined for all $t \in \mathcal{T}$ (it is a piecewise constant function of time). Although a time discretization is used, the SimHeur and DetHeur algorithms may carry out system state updates at any time t in the continuous interval $\mathcal{T}$ (with t depending on the exact amount of CPU time spent on computations). In practice, this means that the values $X_i(0)$, $X_i(0.01)$, $X_i(0.02)$, etc. are stored inside an array and at time t, the algorithm looks up the value $X_i(t')$ where $t' := \max\{u \in \{0,\ 0.01,\ 0.02, ..., T\} : u \leq t\}$; for example, if $t = 2.764$ then we look up the value $X_i(2.76)$.

The steps for generating $\{T_0(t)\}_{t\in\mathcal{T}}$ and $\{T_1(t)\}_{t\in\mathcal{T}}$ are very similar to the above, except we set initial values $T_0(0) = t_0$ and $T_1(0) = t_1$ and use ν as the variance parameter instead of σ_i.

In order to simulate separation times M_{ij}, we pre-generate a set of values $\{p_j^{\mathrm{sep}}\}_{j\in\mathcal{F}}$, with each p_j^{sep} being randomly sampled from a Uniform$[0,1]$ distribution. Then, once the weight class of the preceding flight i becomes known during the simulation, we calculate M_{ij} by sampling the $(p_j^{\mathrm{sep}})^{\mathrm{th}}$ quantile of the Erlang distribution for M_{ij}.

Appendix F: Generating values of the physical and algorithmic parameters

As explained in Section 5, we set $\sigma_i = \sigma$ for all $i \in \mathcal{F}$ and conducted 1000 experiments for each value of σ in the set $\{0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}$. In each test scenario we sample the values of the remaining physical parameters as detailed below. (Note: in the following list, the word ‘sampled’ implies ‘sampled uniformly at random’.)

- The value of k is sampled from the set $\{16,\ 25,\ 44,\ 100,\ 400\}$. This implies that for each pair of consecutively-landing flights (i,j), the separation time M_{ij} has a coefficient of variation sampled from the set $\{0.05,\ 0.1,\ 0.15,\ 0.2,\ 0.25\}$.
- With probability 0.75, the initial forecast $[t_0,\ t_1]$ for the period of bad weather is sampled from the set $\{[285,\ 315],\ [270,\ 330],\ [240, 360]\}$. With the remaining probability 0.25, there is no period of bad weather.
- For simplicity, the weather variance parameter ν is set equal to σ in all experiments, so that the uncertainty in the weather forecast is always similar to the uncertainty affecting unconstrained aircraft arrival times.

- The parameter ϕ affecting separation times under bad weather is set to 10/9, so that bad weather causes a 10% reduction in achievable throughput rates.
- We set $h_i = d_i - 15$ for all $i \in \mathcal{F}$, so that each flight enters its 'tactical uncertainty' phase 15 minutes prior to its scheduled departure time.
- For simplicity, we set $\tau = 30$ in all experiments.
- The relative cost parameter g_i depends on flight i's weight class in the following way: if i is in the 'H' class, g_i is sampled from the continuous interval [0.8, 1]. If i is in the 'UM' or 'LM' class, g_i is sampled from [0.6, 0.8], and if i is in the 'S' class, g_i is sampled from [0.4, 0.6].
- The tolerance parameter $\gamma^{[S]}$ is sampled from the set {0, 15}, with 15 being an 'industry standard' value, as mentioned in Section 2.4. On the other hand, we simply set $\gamma^{[W]} = 0$, so that any amount of air holding delay is penalized.
- The objective function weight for schedule delays $\theta^{[S]}$ is sampled from {0.1, 0.3, 0.5, 0.7, 0.9}. We then set the weight for operational delays as $\theta^{[W]} = 1 - \theta^{[S]}$.

As explained in Section 5, the algorithmic parameters used by the SimHeur and DetHeur algorithms are kept at fixed values in our experiments. We determined these values by conducting a preliminary computational study to investigate sensitivity of the solutions with respect to these parameters. Tests were carried out using a small 'test' problem with a time interval $\mathcal{T}$ of length 2 hours and a set $\mathcal{F}$ consisting of 60 flights with randomly-generated pre-scheduled arrival times and weight classes. Based on the results of this study, we selected the following parameter values:

- (*) Default population size: $S = 20$
- (*) Default sequence length: $l = 15$
- (*) Minimum population size: $S_{\text{min}} = 10$
- (*) Threshold for performing mutations: $m_{\text{mut}} = 25$
- Threshold for performing ranking and selection: $r = 50$
- Significance level used in the ranking and selection criterion: $\eta = 0.05$
- Threshold for releasing aircraft from pool: $n_{\text{rel}} = 50$
- Threshold for filtering population: $n_{\text{repop}} = 500$
- Maximum 'idle runway probability' for delaying pool release: $\lambda = 0$
- Step sizes for sequence cost estimates: $\psi_n = 1/n$

The parameters marked (*) are used by both the SimHeur and DetHeur algorithms. The others are used only by the SimHeur algorithm.

Appendix G: Increasing the number of bad weather periods

In this appendix we present the results of additional experiments in which the number of bad weather periods during $\mathcal{T}$ is increased from 1 to $n \in \{2,3\}$. The method of modeling bad weather periods described in Section 2.3 can be generalized in quite an intuitively simple way so that, instead of a single bad weather period $\mathcal{U}$ predicted to begin at time $T_0(t)$ and end at time $T_1(t)$ (where t is the time point at which the forecast is made), there are multiple bad weather periods $\mathcal{U}^{(i)}$ for $i = 1,...,n$ and each period $\mathcal{U}^{(i)}$ has a predicted start time $T_0^{(i)}(t)$ and end time $T_1^{(i)}(t)$. We initialize the intervals $[T_0^{(i)}(t), T_1^{(i)}(t)]$ so that they do not overlap with each other at $t = 0$, but since we allow $T_0^{(i)}(t)$ and $T_1^{(i)}(t)$ (for each i) to vary according to Brownian motion (BM) as described in Section 2.3, they may overlap at some point $t > 0$. If two periods $\mathcal{U}^{(i)}$ and $\mathcal{U}^{(j)}$ overlap with each other then they effectively become a single, elongated period of bad weather, but we continue to model $T_0^{(i)}(t)$, $T_1^{(i)}(t)$, $T_0^{(j)}(t)$ and $T_1^{(j)}(t)$ as independent BM processes, so that they might later stop overlapping and become distinct periods again. Throughout these experiments we use the same value ν as the variance parameter for all of the BM processes.

We conducted 1000 experiments with $n = 2$ as the number of bad weather periods, followed by a further 1000 experiments with $n = 3$. The experimental setup was exactly as described in Section 5 in all other respects, including the methods used to generate values of the physical parameters. For the experiments with $n = 2$ we set the initial forecasts for the bad weather periods to be from 8AM-9AM and from 11AM-12PM, so that they are both predicted to be one hour long. For the experiments with $n = 3$ we set the initial forecasts to be from 7:40AM-8:20AM, from 9:40AM-10:20AM and from 11:40AM-12:20PM, so that they are each predicted to be 40 minutes long. Thus, in all experiments we have a total of two hours of predicted bad weather, and the bad weather periods are distributed evenly during the 8-hour interval from 6AM to 2PM.

Table 5 shows the results of the experiments with $n = 2$, and Table 6 shows the results for $n = 3$. In both tables we report 95% confidence intervals for the percentage improvement (in terms of the weighted objective function) achieved by SimHeur vs. DetHeur, FCFS and DStat, with comparisons shown over all scenarios and also for the subsets of scenarios corresponding to different values of σ. Thus, the results are shown in the same format as Table 2.

As with the earlier set of experiments (reported in Section 5), we find that SimHeur's advantage vs. DetHeur tends to increase as σ increases. The FCFS policy becomes stronger (relative to SimHeur) as σ increases, while DStat's performance deteriorates rapidly with σ. However, by comparing the results in Tables 2, 5 and 6, we may infer that SimHeur's advantage over DetHeur tends to diminish as the number of bad weather periods increases. Furthermore, in the $\sigma = 0.1$ case, Table 6 indicates that SimHeur may be inferior to DetHeur when $n = 3$.

Table 5 Performance comparisons between SimHeur (SH), DetHeur (DH), First-come-first-served (FC) and DStat (DS) with respect to the value of the weighted objective function (13) for the experiments with $n = 2$.

		Pct. Improvement			**Pct. of Experiments**			
σ value	Count	SH vs. DH	SH vs. FC	SH vs. DS	SH best	DH best	FC best	DS best
Any value	**1000**	$\mathbf{8.50 \pm 0.96}$	$\mathbf{46.92 \pm 0.70}$	$\mathbf{51.82 \pm 1.76}$	**66.80**	**30.90**	**0.40**	**1.90**
0.1	200	0.23 ± 1.74	53.78 ± 1.20	11.72 ± 1.79	46.50	47.50	0.00	6.00
0.3	200	5.28 ± 1.82	51.62 ± 1.16	47.67 ± 2.29	65.00	34.50	0.00	0.50
0.5	200	11.46 ± 2.22	49.15 ± 1.20	59.57 ± 3.89	72.50	24.50	0.00	3.00
0.7	200	11.86 ± 2.22	42.93 ± 1.34	68.62 ± 2.14	74.50	25.50	0.00	0.00
0.9	200	13.70 ± 2.22	37.13 ± 1.66	71.56 ± 2.10	75.50	22.50	2.00	0.00

Table 6 Performance comparisons between SimHeur (SH), DetHeur (DH), First-come-first-served (FC) and DStat (DS) with respect to the value of the weighted objective function (13) for the experiments with $n = 3$.

		Pct. Improvement			**Pct. of Experiments**			
σ value	Count	SH vs. DH	SH vs. FC	SH vs. DS	SH best	DH best	FC best	DS best
Any value	**1000**	$\mathbf{7.65 \pm 1.01}$	$\mathbf{47.54 \pm 0.68}$	$\mathbf{53.03 \pm 1.65}$	**66.20**	**31.70**	**0.10**	**2.00**
0.1	200	-1.75 ± 1.82	54.43 ± 1.20	12.38 ± 1.87	42.50	48.50	0.00	9.00
0.3	200	4.98 ± 2.11	52.44 ± 1.19	49.06 ± 2.24	63.50	35.50	0.00	1.00
0.5	200	9.58 ± 2.28	49.43 ± 1.17	63.27 ± 2.20	69.50	30.50	0.00	0.00
0.7	200	11.67 ± 2.25	43.90 ± 1.29	68.70 ± 2.02	76.00	24.00	0.00	0.00
0.9	200	13.76 ± 2.18	37.49 ± 1.50	71.73 ± 2.25	79.50	20.00	0.50	0.00

Further inspection of the results reveals that when there are multiple periods of bad weather, the nature of the problem changes in such a way that operational delays become more critical. One consequence of this is that it becomes more important for planes to be 'grouped together' in strings of based on their weight classes, in order to minimize average separation times. Indeed, as discussed in Section 2.3, the effect of bad weather is that the required separation times are increased by a scalar factor ϕ, so time separations become more critical in the objective function. Even if the current weather conditions are fine, it makes sense that if a future bad weather period is expected, then runway throughput rates should be maximized in order to mitigate the adverse effects of the future weather. In Section 5.5 we have outlined the reasons why SimHeur's advantage over DetHeur tends to diminish as the relative importance of operational delays (as opposed to schedule delays) becomes higher. In order for SimHeur to perform better under such circumstances, we conjecture that it would need to become more efficient in converging to strong-performing solutions without needing to generate so many random trajectories.