A multi-fidelity modelling approach for airline disruption management using simulation

Journal:	*Journal of the Operational Research Society*
Manuscript ID	TJOR-2020-OP-0610.R1
Manuscript Type:	Original Paper
Date Submitted by the Author:	24-Apr-2021
Complete List of Authors:	Rhodes-Leader, Luke; Lancaster University, STOR-i CDT Nelson, Barry; Lancaster University, Management Science Onggo, Bhakti; University of Southampton, Worthington, David; Lancaster University, Management Science
Keywords:	Simulation, Integer Programming, Multi-objective, Transport, Optimization
Abstract:	Disruption is a serious and common problem for the airline industry. High utilisation of aircraft and airport resources mean that disruptive events can have large knock-on effects for the rest of the schedule. The airline must rearrange their schedule to reduce the impact. The focus in this paper is on the Aircraft Recovery Problem. The complexity and uncertainty involved in the industry makes this a difficult problem to solve. Many deterministic modelling approaches have been proposed, but these struggle to handle the inherent variability in the problem. This paper proposes a multi-fidelity modelling framework, enabling uncertain elements of the environment to be included within the decision making process. We combine a deterministic integer program to find initial solutions and a novel simulation optimisation procedure to improve these solutions. This allows the solutions to be evaluated whilst accounting for the uncertainty of the problem. The empirical evaluation suggests that the combination consistently finds good rescheduling options.
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
ARPrevision.tex	

A multi-fidelity modelling approach for airline disruption management using simulation

L. A. Rhodes-Leader[a], B. L. Nelson[a], B. S. Onggo[b] and D. J. Worthington[a]

[a]STOR-i CDT, Lancaster University, UK; [b]Southampton Business School, Southampton University, UK

ARTICLE HISTORY

Compiled April 23, 2021

ABSTRACT

Disruption is a serious and common problem for the airline industry. High utilisation of aircraft and airport resources mean that disruptive events can have large knock-on effects for the rest of the schedule. The airline must rearrange their schedule to reduce the impact. The focus in this paper is on the Aircraft Recovery Problem. The complexity and uncertainty involved in the industry makes this a difficult problem to solve. Many deterministic modelling approaches have been proposed, but these struggle to handle the inherent variability in the problem. This paper proposes a multi-fidelity modelling framework, enabling uncertain elements of the environment to be included within the decision making process. We combine a deterministic integer program to find initial solutions and a novel simulation optimisation procedure to improve these solutions. This allows the solutions to be evaluated whilst accounting for the uncertainty of the problem. The empirical evaluation suggests that the combination consistently finds good rescheduling options.

KEYWORDS

Simulation; Optimisation; Integer Programming; Multi-objective; Transport

1. Introduction

Disruption to airline schedules creates major issues within the aviation industry. During planning stages, airlines use optimisation algorithms to produce schedules maximising profit with high utilisation of their resources. Developments in this field are enabling schedules with higher utilisation of resources whilst including some slack to recover from minor disruptions. However,

CONTACT L. A. Rhodes-Leader. Email: l.rhodes-leader@lancaster.ac.uk

due to complexity and high levels of uncertainty in the industry, it is rare that flight programmes operate as initially planned. The Federal Aviation Administration estimated that the total cost of delays in the U.S.A. were $33 billion in 2019 (Federal Aviation Administration, 2020). In the same year, only 77.6% of flights in Europe arrived within 15 minutes of the schedule, with 4.3% of flights experiencing departure delays exceeding 1 hour (EUROCONTROL, 2020). Disruptive events have various roots which Bratu and Barnhart (2006) classify as either shortages in airline resources (such as aircraft requiring unscheduled maintenance) or shortages in airport resources (such as reductions in aircraft movements due to weather conditions). In addition, an airline's policy of waiting for connecting passengers can also cause delays. The impacts of such events can propagate through the system causing further delays and cancellations, particularly when the airline has high aircraft utilisation and airport resources operate at or near full capacity.

When a disruption occurs, the Operations Control Centre (OCC) of the airline must propose alterations to its schedule to Air Traffic Control (ATC) for their approval, bearing in mind the repercussions on finances, reputation and passengers. ATC has the right to reject proposals, in which case OCC needs to be able to suggest "good but different" solutions. The OCC wants to identify good actions and evaluate them from the perspective of the airline and the passengers. Kohl, Larsen, Larsen, Ross, and Tiourine (2007) describe the practice of the OCC during disrupted operations, stating that the problem is often split into three: aircraft, crew and passengers. This paper focusses on the Aircraft Recovery Problem (ARP). The available options for the ARP include delaying or cancelling flights, or exchanging aircraft.

The financial costs of a disruption can be significant. However, there are other concerns for the airline too. The airline will not want to incur high levels of delay, as this can lead to discontented passengers and affect reputation. Neither will it want to make too many changes to its schedule. Rerouting aircraft can interfere with crew and future fleet maintenance schedules and gate assignments (Thengvall, Bard, & Yu, 2000). As the ARP solution must be used in conjunction with the crew recovery solution, minimising schedule alterations helps the integration. Whilst both considerations do have financial consequences, these can be difficult to quantify, making this a multi-objective problem.

The complexity of the industry can make it difficult to determine the consequences of schedule alterations. Operational research techniques can be of use here, helping to search for good solutions (Kohl et al., 2007). Some of the complexity can be represented using deterministic models, a number of which have been proposed (Clausen, Larsen, Larsen, & Rezanova, 2010). However, these models cannot fully account for the various uncertain elements of the environment, such as runway queueing times, which can affect the performance and viability of solutions. This raises the question of how the inherent variability of airline operations can be included within the search for the best recovery action, which is the focus of this paper.

A paradigm for modelling the uncertainty is stochastic simulation. In a simulation model, both complexity and uncertainty are explicitly represented. However, simulation does not provide a natural strategy for searching through the possible revised schedules. This search must cope with combinatorial constraints, a difficulty for simulation optimisation. One potential approach is to consider multi-fidelity modelling, combining multiple modelling paradigms of varying accuracy, each used to gain an insight into the problem.

This paper presents a multi-fidelity modelling framework for the ARP, which is designed to address key features of the real problem. Overall it recognises and takes advantage of the multi-objective nature of the problem to generate a range of solutions. Each of these solutions is found by combining the abilities of:

(1) Integer Programming (IP) to efficiently find good solutions to a highly constrained deterministic problem, allowing the discrete reassignment aircraft allocation problem to be solved within appropriate time scales;
(2) simulation to evaluate the performance of complex stochastic systems; and
(3) a simulation optimisation algorithm designed to improve the solutions from the IP.

In our context, we take the 'low-fidelity' model to mean the IP, as it ignores uncertainty and works in discrete time. The simulation will be referred to as a 'high fidelity model', as it aims represent much more of the real system than the IP does.

The primary contribution of this paper is this framework, allowing stochastic information from the simulation to be used within the optimisation. To demonstrate the feasibility and potential benefits of the multi-fidelity modelling framework, the proposed approach is applied to three realistic (but not real) test cases which are designed to reflect the main features of this problem, including realistic flight schedules and disruptions, as well as the multi-criteria nature of the decision making process. Data used in these test cases are given in the Supplementary Material, including sources where available in case other researchers wish to apply other modelling frameworks. Early proof-of-concept work was presented in Rhodes-Leader, Onggo, Worthington, and Nelson (2018).

The remainder of this paper is structured as follows. Section 2 discusses related work in airline disruption management and multi-fidelity modelling. The formal problem statement and method overview are in Section 3. Section 4 describes the low-fidelity IP model and how it is used to find “good but different” solutions to the deterministic problem. Section 5 describes the high-fidelity simulation model, and how simulation optimisation is used to improve upon the IP generated solutions. This includes how we extended the STRONG algorithm for simulation optimisation to deal with the constraints of the ARP. Computational experiments and results are discussed in Section 6, followed by conclusions in Section 7.

2. Related work

Airline disruption management has received much attention within the literature, with many deterministic models and solution methods proposed. Clausen et al. (2010) review the area. Rosenberger, Johnson, and Nemhauser (2003) use a set-packing IP formulation along with a problem reduction heuristic for the ARP. Recently, advances in solvers have allowed more sophisticated exact solution methods, such as column and row generation. These have been applied in both the ARP (Liang et al., 2018) and the combined aircraft and crew problem (Maher, 2016). Heuristic approaches include Steepest Ascent Local Search (Løve, Sørensen, Larsen, & Clausen, 2005), Genetic Algorithms (Jeng, 2012) and Large Neighbourhood Searches (Sinclair, Cordeau, & Laporte, 2014).

One class of models reported in the ARP literature is time-space networks. These consider schedules as networks where flights are arcs between nodes representing airports at a given time. Examples of the use of these models include Thengvall et al. (2000) for the ARP, Bratu and Barnhart (2006) (aircraft and passenger) and Zhang, Henry Lau, and Yu (2015) (aircraft and crew).

As with many real problems, the ARP has a number of performance measures of interest including costs, passenger delays and resuming normal operations quickly (Clausen et al., 2010). Thengvall et al. (2000) note the trade-off between maximum revenue and minimal schedule alterations, though not explicitly including these in the model. Hu, Liao, Zhang, and Song (2017) propose a full multi-objective formulation, trying to identify Pareto optimal solutions, with objectives considering disruption cost, number of aircraft rerouted and maximum delay. The authors use a local neighbourhood search heuristic combined with an ϵ-constraint method.

Rosenberger et al. (2003) conclude that a robust solution accounting for uncertainty may perform better than deterministically optimal solutions. Zhu, Zhu, and Gao (2015) incorporate some uncertainty, assuming the duration of unplanned maintenance is unknown. They propose a two-stage recourse stochastic program, the first allocating aircraft to rotations, then retiming flights. The scenarios are the ready times of aircraft requiring unplanned maintenance. However, including scenarios with other uncertain elements would greatly increase the problem size, making it computationally intractable when decisions must be made quickly.

Rosenberger et al. (2000) develop a stochastic discrete event simulation model of airline operations known as SimAir to simulate the schedule over long periods (weeks or months). This is extended by Rosenberger et al. (2002) and Lee et al. (2003). The purpose of SimAir lies in the schedule planning stage, providing a method to evaluate recovery policies on a long term schedule and monitor its robustness. There is also scope to use simulation at the operational level. A simulation to perform "what if" analysis could be valuable to the OCC (Kohl et al., 2007). Abdelghany, Abdelghany, and Ekollu (2008) propose a network simulation to predict which flights will be disrupted by considering future aircraft and crew connections. Only these flights are included in an IP model for the aircraft and crew recovery problem. Hutchison and Hill (2001) propose a Simultaneous Perturbation Stochastic Approximation simulation optimisation process to change the delays on flights of Ground Delay Programmes under poor weather conditions. This did not involve reallocating aircraft.

Stochastic turn times and flight durations within the ARP are considered by Arias, Mujica Mota, Guimarans, and Boosten (2013), combining a deterministic constraint programming approach with a Monte Carlo simulation to try to minimise delay. The simulation evaluates the constraint program solution under several scenarios by adding normal noise to the turn times and flight durations. If the solution is not robust across these scenarios, it is rejected and the process begins again. Guimarans, Arias, and Mujica Mota (2015) expand this approach by combining the constraint program with a Large Neighbourhood Search to propose new solutions and use the simulation at each proposed solution to evaluate the acceptance criteria. However, the simulation remains rudimentary, and using a higher fidelity model would harm the performance of the algorithm if the simulation is used at every iteration. Of the current literature, our proposal most closely resembles the approaches of Arias et al. (2013) and Guimarans et al. (2015), with the aim to incorporate a higher-fidelity simulation model. Wang et al. (2019) develop a discrete event dynamic system simulation model for the ARP. Rather than using optimisation, which flights to delay or cancel is based on a set of rule-based methods. Reallocating aircraft is not considered. The simulation is used to identify the best proposed solution.

Multi-fidelity modelling could improve the efficiency of simulation optimisation by reducing computation times. The simpler, or 'low-fidelity', models can quickly identify potentially good solutions to be evaluated using the highest fidelity model for an improved understanding of their performance. These are prevalent in simheuristics (Juan, Faulin, Grasman, Rabe, & Figueira, 2015), which often use deterministic models and heuristic searches to identify solutions to test in the simulation. Examples include combining a linear program with a simulation for production planning in a manufacturing context (Bang & Kim, 2010), a deterministic model with simulation for the inventory routing model with stochastic demands (Onggo, Panadero, Corlu, & Juan, 2019) and infinite-server queueing models and simulation for staffing A&E departments (Izady & Worthington, 2012). Xu et al. (2016) argue that low-fidelity models may have inconsistent inaccuracies, changing the optimal solution. This motivates utilising the simulation information within the search process itself. For example, one could model the high-fidelity performance as the low-fidelity performance plus an error function, which could be a quadratic response surface (Osorio & Chong, 2015) or a Gaussian Process (Huang, Allen, Notz, & Miller, 2006; Inanlouganji, Pedrielli, Fainekos, & Pokutta, 2018). This allows both models to be used within the search. Our algorithm is more similar to that of Jian and Henderson (2015) who use simulation optimisation to improve upon the solutions from low-fidelity fluid and Markov chain models for bike allocation within a bike-sharing system.

3. Problem statement and method overview

Suppose an incident has occurred that disrupts an airline's intended schedule. The airline wishes to take action to minimise the disruption's impact, returning to normal operations within a certain time period known as the *recovery window*. The recovery window could extend to the end of the operating day (for short-haul carriers), as in Løve et al. (2005) and Zhang et al. (2015). Maher (2016) and Hu et al. (2017) use shorter recovery windows, whilst Sinclair et al. (2014) works with periods of multiple days. Note that when the recovery window does not coincide with a break in operations, such as the day's end, one must consider the desired state of the fleet at the end of the recovery window. Maher (2016) point out that longer recovery windows can make the problems more difficult to solve.

Let A be the set of aircraft in play for resolving the disruption, wherever they are located. Here we assume all aircraft are of the same type. Let F be the set of flights originally covered by A within the recovery window, with $n = |F|$ being the number of flights. Each flight, $f \in F$, will require an aircraft, $a \in A$, and a planned delay time, d^f, or a cancellation which must be submitted to Air Traffic Control (ATC) before the departure time. Let $\mathbf{x} = \{x_a^f \in \{0,1\} : f \in F, a \in A\}$ denote the aircraft allocation (where $x_a^f \in \{0,1\}$ and $x_a^f = 1$ if and only if aircraft a is assigned to flight f) and $\mathbf{d} = (d^f : f \in F)$ be the vector of planned delays. The OCC has the multi-objective aim of rescheduling to minimise alterations to the original schedule, delays and costs by setting the decision variables **x** and **d**. The costs arise directly from delays, passenger compensation and cancellation charges. Furthermore, this plan should be achievable, avoiding over-promising which could lead to flight f being delayed by more than d^f, damaging the airline's reputation and resulting in operational costs as a new plan with further schedule adjustments must be submitted to ATC. These additional costs are accounted for by a penalty if the actual delay exceeds d^f.

This paper proposes a multi-fidelity modelling approach to the ARP with two stages. Stage one uses a low-fidelity deterministic time-space network IP with discrete time. This IP identifies an aircraft allocation to flights and gives an initial value for the planned delay of each flight, considering all objectives. Solving the IP in a multi-objective manner produces a set of rescheduling options with different priorities for these objectives. This allows the combinatorial aircraft allocation aspect of the problem to be solved in a deterministic manner, separating it from the simulation optimisation.

The second stage uses each solution from the IP as a starting point for a local search for improvement in expected cost using simulation optimisation. This search enables more detailed information from the simulation to direct the optimisation process. We fix the aircraft allocation for each solution, treating only the planned delays as decision variables, removing the combinatorial constraints. The delays are treated as continuous variables, allowing the use of continuous simulation optimisation methods and gradient information.

4. Low-fidelity model

In this section we describe our low-fidelity model.

4.1. Time-space network model

The low-fidelity model is based on a time-space network IP (e.g., Thengvall et al., 2000). The flight schedule covering the recovery window is represented by a sequence of flight arcs between airports and ground arcs connecting a landing to a subsequent take-off. The departure and arrival of each flight is a transition node, $\nu \in V_T$, representing an airport at a particular time. Under disruption, the network is augmented with additional flight arcs for each flight starting from a later time. Let f_δ be the flight arc representing the flight f delayed by δ minutes. The potential delays are discrete time steps of size m up to a maximum M, i.e. $\delta \in \{0, m, 2m, ..., M\}$. This introduces new nodes and ground arcs into the network. Let L denote the set of all flight arcs, while the set of flight arcs associated with flight f is L^f. G is the set of all ground arcs in the network.

In addition to the transition nodes, each aircraft $a \in A$ has an input node, $i(a) \in V_I$, representing its location at the beginning of the recovery window. This applies to all aircraft in the fleet, regardless of the airport at which they are currently located. The input node is connected to the first node in V_T at which a is available for flight by a ground arc. No other arcs leave $i(a)$. All aircraft will end the recovery window at a return node, $r \in V_R$. Some aircraft may have a specified return node, denoted $r(a)$. For example, an aircraft may be required at an airport for its scheduled maintenance. In this case, a constraint is imposed to ensure the new schedule respects the maintenance plan. Let A_R denote the set of aircraft required to be at specific return nodes. The complete set of all nodes is $V = V_T \cup V_I \cup V_R$.

Figure 1 shows an example network for two aircraft, A1 and A2. The original schedule is represented by the original flight arcs and the ground arcs. At the end of the day, A1 is required at BHX for maintenance, so the return node for BHX is labelled $r(\text{A1})$ and $A_R = \{\text{A1}\}$. Suppose A2 will not be ready to fly until 7:00, hence $i(A2)$ is connected to node (BHX,7) rather than (BHX,6). If M=60 minutes and m=30 minutes, two additional flight arcs are added for each flight representing a delay of 30 minutes (dashed arcs) and 60 minutes (dotted arcs). The recovery window ends at 18:00, so no additional arcs are added that arrive beyond this time.

[Figure 1 about here.]

To ensure continuous aircraft movement through the network, flow constraints are imposed. Let L^{ν}_{in} and G^{ν}_{in} be the sets of flight arcs and ground arcs incident on node $\nu \in V$, and L^{ν}_{out} and G^{ν}_{out} be the sets of flight arcs and ground arcs exiting $\nu \in V$.

Between flights, aircraft must be prepared for the next flight. The industry term for this is the 'turn time'. The schedule must leave a minimum turn time between flights, $t_{\min}$. For node ν representing an airport at time t_ν, let $V^{\nu}_{t_{\min}}$ be the set of all nodes, $\nu' \in V_T$, representing the same airport at time $t_{\nu'}$ less than $t_{\min}$ later than time t_ν, i.e. $t_{\nu'} \in [t_\nu, t_\nu + t_{\min})$. If an aircraft ends a flight at node ν, it cannot start a new flight from any node in $V^{\nu}_{t_{\min}}$. The choice of $t_{\min}$ affects the recovery solution. Since turn time is actually variable, larger values of $t_{\min}$ correspond to higher quantiles of the turn time distribution, creating more robust solutions. However, if $t_{\min}$ is too large, it may delay flights that are not disrupted (if there is little buffer between flights).

A recovery plan must also ensure that the schedule beyond the recovery window is feasible with little or no disruption. For this to occur, the airline must have sufficient aircraft at each airport to operate the schedule after the recovery window. This is known as aircraft balance. Let r_A be the number of aircraft required by return node r. This constraint is used so that normal operations, i.e. no delays or cancellations, are resumed by the end of the recovery window. However, the aircraft allocation may differ from the original schedule.

An airline may only be allowed to use an airport runway during particular time slots due to the airport's runway capacity constraints. A "slot" s is a time period at a particular airport, W_s, with a restriction on the number of aircraft movements. The set S lists all slots. Let K_s be the maximum number of aircraft movements allowed by the airline in slot s at the relevant airport W_s and L_s be all flight arcs impacting slot s. This mechanism also deals with curfew times at airports, for example, preventing aircraft taking off during the night.

4.2. Integer program formulation

The model formulation is adapted from the aircraft recovery model in Zhang et al. (2015), but also draws on the models of Thengvall et al. (2000) and Jeng (2012), modified to allocate specific aircraft to flights, rather than fleet assignment.

We assume the cost of delaying flight f by δ is given by $c^{f_\delta} = c_d\delta + P^f(\delta)$, where c_d is the cost of delay per minute and P^f is the appropriate passenger compensation cost function, with the cost per passenger often being a step function of the delay. For example, for short-haul flights arriving or departing Europe, Civil Aviation Authority (2015) state that for a delay of 2 hours, the airline must cover food and drink of each passenger, whilst a delay of over 3 hours leads to a compensation of €250 per passenger. However, more general delay costs could be accommodated within this framework. Let C^f be the cost of cancelling flight f. Let $o_a^f \in \{0, 1\}$ indicate whether aircraft a was assigned to flight f before the disruption occurred.

The decision variables for the IP are all binary variables. Let $x_a^{f_\delta}$ indicate that aircraft a is assigned to flight arc f_δ, y^f indicate that flight f is cancelled, and z_a^γ indicate that aircraft a uses ground arc γ. The complete formulation is as follows.

$$\min \quad \sum_{a\in A}\sum_{f_\delta\in L} c^{f_\delta} x_a^{f_\delta} + \sum_{f\in F} C^f y^f \tag{1a}$$

$$\min \quad \sum_{a\in A}\sum_{f_\delta\in L} (1 - o_a^f) x_a^{f_\delta} \tag{1b}$$

$$\min \quad \sum_{a\in A}\sum_{f_\delta\in L} \delta x_a^{f_\delta} \tag{1c}$$

subject to

$$\sum_{a\in A}\sum_{f_\delta\in L^f} x_a^{f_\delta} + y^f = 1 \qquad \forall f \in F \tag{1d}$$

$$\sum_{a\in A}\sum_{f_\delta\in L_s} x_a^{f_\delta} \le K_s \qquad \forall s \in S \tag{1e}$$

$$\sum_{f_\delta\in L_{\text{in}}^{\nu}} x_a^{f_\delta} + \sum_{f_\delta\in L_{\text{out}}^{\nu}} x_a^{f_\delta} + \sum_{\nu'\in V_{t_{\min}}^{\nu}}\ \sum_{f_\delta\in L_{\text{out}}^{\nu'}} x_a^{f_\delta} \le 1 \qquad \forall a \in A, \forall \nu \in V_T \tag{1f}$$

$$\sum_{f_\delta\in L_{\text{in}}^{\nu}} x_a^{f_\delta} + \sum_{\gamma\in G_{\text{in}}^{\nu}} z_a^{\gamma} - \sum_{f_\delta\in L_{\text{out}}^{\nu}} x_a^{f_\delta} - \sum_{\gamma\in G_{\text{out}}^{\nu}} z_a^{\gamma} = 0 \qquad \forall a \in A, \forall \nu \in V_T \tag{1g}$$

$$\sum_{\gamma\in G_{\text{out}}^{i}} z_a^{\gamma} = 1_{\{i=i(a)\}} \qquad \forall a \in A, \forall i \in V_I \tag{1h}$$

$$\sum_{f_\delta\in L_{\text{in}}^{r(a)}} x_a^{f_\delta} + \sum_{\gamma\in G_{\text{in}}^{r(a)}} z_a^{\gamma} = 1 \qquad \forall a \in A_R \tag{1i}$$

$$\sum_{a\in A}\sum_{f_\delta\in L_{\text{in}}^{r}} x_a^{f_\delta} + \sum_{a\in A}\sum_{\gamma\in G_{\text{in}}^{r}} z_a^{\gamma} \ge r_A \qquad \forall r \in V_R \tag{1j}$$

$$x_a^{f_\delta}, z_a^{\gamma}, y^f \in \{0,1\} \qquad \forall a \in A, f \in F, f_\delta \in L, \gamma \in G \tag{1k}$$

Objective (1a) relates to the cost of the recovery action, objective (1b) to the number of changes made to aircraft allocation and objective (1c) to the total planned delay. Constraint (1d) ensures that each flight is either flown once or cancelled. Constraint (1e) is the slot constraint. Constraint (1f) prevents a turn time of less than the minimum allowable turn time; if aircraft a uses a flight arc incident on node ν, it cannot then use a flight arc exiting ν or any other node ν' within $t_{\min}$ of ν. Constraint (1g) is a flow constraint for each $\nu \in V_T$, whilst constraints (1h) and (1i) are analogous for the input and return nodes (where applicable), respectively. Constraint (1j) ensures aircraft balance at the end of the recovery window.

4.3. Solving the integer program

To generate multiple Pareto optimal solutions, the IP is solved using the ϵ-constraint method (Haimes, Lasdon, & Wismer, 1971). The objectives (1b) and (1c) are added to the problem as constraints:

$$\sum_{a\in A}\sum_{f_\delta\in L}(1-o_a^f)x_a^{f_\delta} \in [l_E, u_E]$$

$$\sum_{a\in A}\sum_{f_\delta\in L}\delta x_a^{f_\delta} \in [l_D, u_D].$$

The program is solved several times with different constraint limits, l_D, u_D, l_E and u_E, minimising objective (1a). These parameters are modified efficiently using the method proposed by Laumanns, Thiele, and Zitzler (2006) to explore various regions of the Pareto frontier. For further information about this method, see Section 1 of the accompanying supplementary material. A time limit is imposed for this application. Each iteration, defined by a set of constraint limits, is solved to optimality, except the final iteration which may run out of time.

The result is a set of solutions, $\mathcal{X}$, each with an aircraft allocation (including which flights are cancelled), $\mathbf{x}$, and a delay value for all flights in the programme, $\mathbf{d}$. The time limit means Pareto optimal solutions cannot be guaranteed. These solutions become starting points for the simulation optimisation, seeking local improvement.

5. High-fidelity model

The input recovery schedule for the simulation is given by an aircraft allocation $\mathbf{x}$ and a set of planned delays, $\mathbf{d}$. The elements of $\mathbf{x}$ and $\mathbf{d}$ link to the IP variables as follows:

$$x_a^f = \sum_{f_\delta \in L^f} x_a^{f_\delta},$$
$$d^f = \sum_{a \in A} \sum_{f_\delta \in L^f} \delta x_a^{f_\delta},$$

whilst a cancellation of flight f can be inferred when $\sum_{a \in A} x_a^f = 0$.

Let D^f be the actual delay of flight f. This is a random variable with a distribution dependent on $(\mathbf{x}, \mathbf{d})$. The objective function is the disruption's expected cost:

$$g(\mathbf{x}, \mathbf{d}) = \mathbb{E}\left\{ \sum_{f \in F} \left(c_d D^f + c_p (D^f - d^f)^+ + P^f(D^f) \right) \right\} + C(\mathbf{x}). \tag{2}$$

Here, c_p is the penalty per minute for the actual delay D^f exceeding the planned delay d^f, incurred in addition to the delay cost $c_d D^f$. $P^f(D^f)$ represents the compensation associated with passenger delays of flight f and $C(\mathbf{x})$ is the cost of cancellations in the allocation $\mathbf{x}$. This objective function corresponds to (1a) in the IP. However, the framework could be used to handle any cost function used by an airline.

5.1. Simulation model

The simulation model is built within AnyLogic 8.2.3 (The AnyLogic Company, 2017). It simulates a homogeneous sub-fleet of a short-haul airline operating the recovery action $(\mathbf{x}, \mathbf{d})$ over a set of airports during the defined recovery window. Each aircraft follows its assigned schedule, subject to stochastic flight durations, turn times, queueing times and maintenance. The general framework is similar to SimAir, as discussed in Lee et al. (2003).

Where possible, distribution parameters used Maximum Likelihood Estimators based on the available data (Flightradar24 AB, 2017), obtained using the `pyflightdata` Python package (Allamraju, 2017). The data set contains information on 2.8 million flights, starting or ending at a set of 163 European airports. The information consists of origin, destination, aircraft type and identification, airline, departure and arrival times, and approximately half of the entries also have scheduled departure and arrival times. Where the data source does not track this information, assumptions on distributions and parameters were made. For a more in depth reporting of the simulation model used here, see Sections 2 and 3 of the supplementary material.

Whilst the simulation does not model all the details of airline operations, it does contain the key features necessary to evaluate our multi-fidelity approach. An airline has access to their own historical data for flight durations and turn times at all airports of interest, and a detailed knowledge of aircraft time-to-failure distributions and the times to repair aircraft, based on their facilities across the network. In addition, they will know the operating procedures at the individual airports at which they operate, accurate weather forecasts, and use of general data sources such as Flightradar24 AB (2017) to estimate how other airline's operations may impact on runway operations. All of this information will allow an airline to improve upon the simulation model used here, making it more specialised and relevant to their circumstances.

5.2. Simulation optimisation

Solutions found using the low-fidelity model, $(\mathbf{x}, \mathbf{d}_0) \in \mathcal{X}$, can be evaluated with more fidelity using the simulation model. Furthermore, to account for uncertainty not present in the low-fidelity model, an improvement is sought. Fixing $\mathbf{x}$ for each solution removes the combinatorial aspects, leaving a continuous optimisation problem over the planned delays, $\mathbf{d}$. To reduce the problem further, only the n^+ flights to which the IP allocates non-zero delays, $f \in F^+ = \{f \in$

$F : d_0^f > 0\}$, are varied in the optimisation. The simulation optimisation problem is therefore

$$\begin{aligned} &\min_{\mathbf{d}} && g(\mathbf{x}, \mathbf{d}) && && (3a)\\ &\text{subject to} && d^f \in [0, u^f] && \forall f \in F^+ && (3b)\\ & && d^f = 0 && \forall f \notin F^+ && (3c) \end{aligned}$$

where g is defined in equation (2), and the constraint (3b) states that a flight cannot leave early and must not exceed a maximum delay u^f. Thus, the feasible region, $\mathcal{D}$, is a hyper-box in $\mathbb{R}^{n^+}$.

We created an extended version of the STRONG simulation optimisation algorithm proposed by Chang, Hong, and Wan (2013) to enable it to cope with constraints. The main components of the extension are described below.

STRONG combines trust-region optimisation (see Conn, Gould, & Toint, 2000) with Response Surface Modelling (see Kleijnen, 2014). At the j^{th} iteration, a linear or quadratic meta-model, $\hat{q}_j(\mathbf{d})$, is built to approximate the objective function around the current solution, $\mathbf{d}_j$. A new solution, $\mathbf{d}_j^*$, is proposed using a sub-problem of (approximately) minimising $\hat{q}_j(\mathbf{d})$ over an area around $\mathbf{d}_j$, $\mathcal{B}_j$, known as the trust region and determined by the trust-region radius, Δ_j. The proposed step to $\mathbf{d}_j^*$ is accepted or rejected based on two tests. The first compares the observed decrease in g with the predicted decrease in $\hat{q}_j$, whilst the second is a hypothesis test to see if $g(\mathbf{x}, \mathbf{d}_j^*)$ is significantly lower than $g(\mathbf{x}, \mathbf{d}_j)$ in the statistical sense. The trust region shrinks or expands depending on whether the meta-model produces a good solution. The primary difficulty for simulation optimisation is that $g(\mathbf{x}, \mathbf{d})$ and $\hat{q}_j(\mathbf{d})$ can only be estimated by taking multiple replications of the simulation at the solution $(\mathbf{x}, \mathbf{d})$.

As STRONG is for unconstrained optimisation, some significant adaptations are required as the optimum may lie on the boundary of $\mathcal{D}$. We briefly describe these in the remainder of this section. For further details on STRONG and the implementation of these adaptations, see Chang et al. (2013) and Section 4 of the Supplementary Materials, respectively.

5.2.1. Building the meta-model

When near a boundary, the classical factorial designs proposed by Chang et al. (2013) for fitting $\hat{q}_j(\mathbf{d})$ are not feasible as some design points may lie outside $\mathcal{D}$. A good design matrix, $\mathbf{D}_j$, is required to estimate $\hat{q}_j(\mathbf{d})$. One criterion for designs is D-optimality, which seeks to minimise the generalised variance of the design, $\det[(\mathbf{D}_j^T \mathbf{D}_j)^{-1}]$ (see Montgomery, 2009). Finding such a design is a complex optimisation problem. We implement the Coordinate-Exchange Algorithm heuristic proposed by Meyer and Nachtsheim (1995) to build the design in each iteration. This reduces the task to a sequence of one-dimensional problems by iteratively moving the design points around the trust region in a coordinate-wise manner to increase det$[\mathbf{D}_j^T\mathbf{D}_j]$. Whilst not guaranteeing optimality, it produces good designs quickly, an important feature as it is run at each iteration in STRONG.

5.2.2. Proposing a new solution

After the meta-model is estimated, the sub-problem needs to be solved. Constraints on the feasible region mean that a solution proposed by minimising $\hat{q}_j(\mathbf{d})$ in the negative gradient direction within the trust region (as in STRONG) may not be feasible. Thus we adapt a procedure to guarantee a feasible solution. For bound constraints, such as (3b), Conn, Gould, and Toint (1988) suggested using a hyper-box trust region, defined by the ℓ_∞ norm, as this aligns itself with the boundaries of $\mathcal{D}$. Thus, our sub-problem becomes:

$$\min_{\mathbf{s}} \quad \hat{q}_j(\mathbf{d}_j + \mathbf{s}) \tag{4a}$$

$$\text{subject to} \quad \mathbf{d}_j + \mathbf{s} \in \mathcal{D} \tag{4b}$$

$$||\mathbf{s}||_\infty \leq \Delta_j. \tag{4c}$$

The objective is to find a step $\mathbf{s}$ that minimises the meta-model, $\hat{q}_j(\mathbf{d})$, whilst retaining feasibility (constraint (4b)) and remaining within the trust region (constraint (4c)). Rather than solving this constrained problem exactly, Conn et al. (1988) propose finding the Generalised Cauchy Step, $\mathbf{s}_j^*$, the step that minimises $\hat{q}_j(\mathbf{d})$ along the negative gradient direction projected onto the feasible trust region. This path is known as the projected gradient path, and consists of a series of straight lines. This simplified problem can be solved exactly. For the linear model, the step is to a corner of the hyper-box $\mathcal{D}\cap\mathcal{B}_j$. For the quadratic models, we use Algorithm 17.3.1 of Conn et al. (2000, p. 791), which finds local minima in each line segment of the project gradient path, and selects the smallest. The proposed solution, $\mathbf{d}_j^* = \mathbf{d}_j + \mathbf{s}_j^*$, is then simulated to estimate $g(\mathbf{x}, \mathbf{d}_j^*)$.

5.2.3. Final stages

The algorithm ends when a replication budget is exhausted or a tolerance on optimality has been met. Due to discretisation of the IP, some unnecessary delays may have been inserted. Therefore, the algorithm tests the final solution $(\mathbf{x}, \mathbf{d}^*)$ against $(\mathbf{x}, \mathbf{0})$, to check that these delays are necessary. A one-sided Welch's t-test (Welch, 1938) is used to compare:

$$H_{\text{null}}: \; g(\mathbf{x}, \mathbf{0}) \geq g(\mathbf{x}, \mathbf{d}^*); \qquad H_{\text{alt}}: \; g(\mathbf{x}, \mathbf{0}) < g(\mathbf{x}, \mathbf{d}^*).$$

If H_{null} is rejected, the final solution returned is $(\mathbf{x}, \mathbf{0})$.

The whole procedure is performed for each $(\mathbf{x}, \mathbf{d}_0) \in \mathcal{X}$, producing a set of improved solutions that have been thoroughly evaluated using the simulation. The airline can then consider each solution alongside other practicalities, before a solution is proposed to ATC.

6. Empirical Evaluation

In practice, the IP would be solved as in Section 4.3 to produce a set of solutions and the simulation optimisation procedure of Section 5.2 would then be used once on each solution. However, when performing a computational evaluation of simulation optimisation algorithms on real-world problems, the reporting must consider two levels of uncertainty. The first is the uncertainty in the performance of an individual solution. As the exact performance is unknown, performance must be estimated using many replications of the simulation. As the rescheduling is a one-time-only decision, understanding the full distribution of outcomes is important, rather than just the expected value. Thus, the solution returned by the simulation optimisation algorithm is simulated many times for a thorough understanding of its performance. Using Common Random Numbers (CRN) improves the comparison by ensuring, as much as possible, that the solutions face the same situations.

The second layer is in the algorithm itself. The algorithm described in Section 5 relies on the random output of simulation replications. Therefore, the solution returned is dependent on the starting seed of the random number stream; a single algorithm run cannot characterise the algorithm's performance. Thus, for the purposes of evaluating this method, for each problem, we perform macro-replications of the whole algorithm (excluding the IP, which is deterministic). Each macro-replication produces a possibly unique solution, which is simulated using 1000 CRN replications.

Another difficulty is that the optimal expected cost is unknown. Thus, one cannot simply measure the optimality gap. Instead, we either evaluate the simulation optimisation performance in terms of the improvement over the IP solutions or the gap to the best solution found across the macro-replications.

6.1. *Problem descriptions*

Three realistic problems were created based on short-haul carrier schedules extracted from the data source (Flightradar24 AB, 2017). Problem 1 involves a fleet of 8 aircraft, one of which has a technical fault requiring a few hours to repair. Problem 2 has a similar setting, but with a fleet of 48 aircraft. In both cases, the proposed method is used to reschedule the remainder of the day's flights.

In Problem 3 poor weather at a hub airport causes high congestion at the runways, affecting 11 immediate flights and their subsequent schedules. The fleet size is 102 aircraft. Here, the airline cannot make changes to flights from the affected airport, which are excluded from the IP, but can rearrange future flights. The original schedule under the disruption is simulated to estimate when the affected aircraft become available after completing their weather disrupted flights. These times are used for the respective input node times within the IP.

The cost per minute of delay is set to $c_d = €50$ with a penalty cost of $c_p = €20$ per minute. Further details of the problems, algorithm parameter settings and results can be found in the supplementary material.

6.2. *Integer program results*

For each problem, the IP was solved using the Gurobi Optimizer 7.0.2 (Gurobi Optimization, LLC., 2017) on a Cluster of 4 nodes each with 2 x Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.6GHz and 512Gb RAM, running Ubuntu Linux. For Problem 1, 8 processors were used, and 20 processors for Problems 2 and 3. The results are shown in Table 1.

[Table 1 about here.]

Each of the problems show a clear trade-off between cost and the number of aircraft exchanges. This is predictable, since allowing more aircraft to be exchanged gives more flexibility to make schedule changes, and thus lower cost solutions become available. In each problem, the algorithm prevents cancellations and largely avoids long delays. Therefore, compensation costs are generally avoided, as these only apply to cancelled flights and delays exceeding 2 hours. Thus, cost and delay are highly correlated.

Many of the solutions are achieved within two minutes. This speed would be appropriate for the application. However, for the larger problems, some of the solution times are too long, especially as the maximum allowable delay for Problem 3 is only $M = 60$ minutes. Whilst the time available to solve a problem depends on the nature of the disruption, the time limits used in the literature vary from 3 minutes (e.g. Løve et al., 2005) to 20 minutes for a multi-objective approach (Hu et al., 2017). As the IP is only the first stage of our proposed framework, this suggests more specialised problem reduction and solution approaches are required.

One important observation is the time to find the first Pareto optimal solution in the larger problems: it took 8 and 12 iterations of the multi-objective algorithm for Problems 2 and 3, respectively. This suggests that the pure ϵ-constraint method could be be improved upon, such as using a hybrid multi-objective to penalise aircraft exchanges that are not Pareto optimal.

Additional experimentation (not included in Table 1) indicated a further trade-off in the choice of m. A smaller value of m allows a greater choice of flight arcs; thus a more flexible solution space. This can improve solutions significantly as new aircraft exchanges become available. However, the IP grows quickly as m decreases (the number of variables is $O(1/m)$). Moreover, Thengvall et al. (2000) found it increased the number of non-integer solutions to the linear relaxation, increasing the solution time. This is particularly clear in Problem 3. Using m=10 minutes instead gives an optimal cost of €15,000, but the first iteration takes 864.70s to solve. The improvement in solution quality suggests that developing problem reduction techniques that enable a smaller step size would be advantageous for real implementation.

6.3. Simulation optimisation results

Each of the solutions in Table 1 provide a starting point for the simulation optimisation algorithm, with a budget of 2000 simulation replications, and 50 replications for the comparison with $\mathbf{d} = \mathbf{0}$. (An iteration in progress when the budget runs out is allowed to complete.) The maximum number of samples at each design point to build the meta-model $\hat{q}_j(\mathbf{d})$ is 50. Up to 100 replications are used for the acceptance tests. The full set of algorithm parameters used are listed in Section 5 of the supplementary material.

Each experiment is repeated multiple times (100 for Problem 1, and 50 for Problems 2 and 3). This produces either 100 or 50 (possibly unique) solutions for each starting point. To evaluate these solutions, each is simulated 1000 times using CRN to estimate its cost distribution.

For comparison, each starting point and a “No Action” response are also simulated using the same CRN replications. This solution involves no changes to the schedule; each disrupted aircraft operates its original schedule once it is available to do so, generally leading to large unplanned delays. Whilst unrealistic, the “No Action” response of making no changes to the schedule gives a benchmark for potential improvement.

Figures 2, 3 and 4 show the Empirical Cumulative Distribution Functions (ECDFs) of the simulated cost of the IP solution, the “No Action” solution and the simulation optimisation solutions (combined into one distribution). Combining the simulation optimisation solutions into one ECDF helps to account for the variance due to the starting seed, though it can hide poor performing solutions.

[Figure 2 about here.]

The ideal outcome from these plots would be the simulation optimisation ECDF never going below the IP solution ECDF. In general, whilst the mean decreases in all scenarios, the variance increases as the simulation optimisation capitalises on the scheduled buffer to reduce delays where possible. The worst case is seen in Problem 1 (Figure 2) where the upper tail of the distributions get longer. The majority of the macro-replication solutions see an increase in the 95th percentile. However, the upper quantiles are similar to the IP solutions, giving a sense of robustness.

For Problem 2, see Figure 3, the improvement is smaller, as much of the gains have been found using the IP. For each of the starting solutions the 95th percentile is reduced in at least 78% of macro-replications.

[Figure 3 about here.]

In the third problem, Figure 4, the gain made by the IP is fairly small. These poor initial results could be due to conservatively estimated ready times of the aircraft directly affected by the poor weather. However, the change in aircraft allocation allows the simulation optimisation to reduce the costs. The combination leads to a much improved result. In all macro-replications, the simulation reduces both the mean and the 95th percentile cost.

[Figure 4 about here.]

In all cases the IP and simulation optimisation solutions are more robust than the “No Action” option.

The final analysis considers the consistency of the simulation optimisation. As the aim is to minimise the expected cost, we consider the gap between the best mean cost from the macro-replications. The estimated “relative optimality gap” of the solution of the k^{th} macro-replication of Plan p, $(\mathbf{x}_p, \mathbf{d}_{kp})$, is

$$\rho_{\text{gap}}(\mathbf{x}_p, \mathbf{d}_{kp}) = \frac{\hat{g}(\mathbf{x}_p, \mathbf{d}_{kp}) - \min_k\{\hat{g}(\mathbf{x}_p, \mathbf{d}_{kp})\}}{\min_k\{\hat{g}(\mathbf{x}_p, \mathbf{d}_{kp})\}} \times 100\%.$$

Note that as the true optimal is unknown, this is an estimate of the optimality gap relative to the best solution discovered. Figure 5 shows the ECDFs of $\rho_{\text{gap}}(\mathbf{x}_p, \mathbf{d}_{kp})$ for each problem (all starting points combined into one plot). It shows that the algorithm is least consistent for Problem 1, where it is only within 5% of the best solution in 60% of macro-replications. The consistency is much higher for Problems 2 and 3, where in 90% of macro-replications it achieves within 5% of the best solution found.

[Figure 5 about here.]

7. Conclusions and further work

This paper has proposed a multi-fidelity modelling approach for the ARP. It combines integer programming with simulation to effectively search through the solution space. Using the simulation as part of the optimisation allows some correction for simplifications in the IP. The nature of the real problem is that OCC needs to be able to integrate the ARP solution with other parts of the recovery plan and provide it to ATC, who can reject a proposed plan. This increases the value of producing a range of good but different solutions within a limited time. The results show that this combination is consistently able to find good solutions to the ARP, with the simulation optimisation providing improvements over the initial solutions. We believe that the combination of IP to find Pareto-optimal aircraft allocations to a deterministic version of the problem, followed by a local-search based simulation optimisation to fine tune the delays is well-suited to the ARP context.

As part of the work it was necessary to adapt the simulation optimisation algorithm STRONG to cope with bound constraints, and as such could be applicable beyond the ARP studied here.

Whilst we have demonstrated our framework across three realistic problems, further investigation of the framework across a larger variety of problem instances would improve our understanding of the performance and potentially identify if there are particular problem structures or sizes that cause a difficulty for this framework.

To move to a real implementation, the major challenge is a reduction in the computation time. The pure ϵ-constraint method for the IP could be compared with a hybrid multi-objective approach, investigating which finds the Pareto frontier sooner. Furthermore, more problem-specific solution methods could improve the solution times for the IP. The simulation optimisation procedure could be naturally parallelised by simulating each design point on a different processor. This would decrease the computation time significantly, as this is the most expensive part of the process. One may also wish to improve what the simulation explicitly models, such as passenger connections and non-linear costs. This would improve the performance estimation and increase the value of the method.

Acknowledgement

We would like to thank Richard Standing, Nigel Jackson, Stewart Preston and Mike Chester at Rolls-Royce (R^2 Data Labs) for the original research idea and contextual information.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the EPSRC under Grant EP/L015692/1 STOR-i Centre for Doctoral Training; NSF under DMS-1854562; and Rolls-Royce Limited.

References

Abdelghany, K. F., Abdelghany, A. F., & Ekollu, G. (2008). An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, *185*(2), 825–848.

Allamraju, H. (2017). *pyflightdata.* Accessed March 11th, 2017. `https://github.com/supercoderz/pyflightdata`.

Arias, P., Mujica Mota, M., Guimarans, D., & Boosten, G. (2013). A methodology combining optimization and simulation for real applications of the stochastic aircraft recovery problem. In *Proceedings of the 8th EUROSIM Congress on Modelling and Simulation, EUROSIM 2013* (pp. 265–270). Washington, DC, USA: IEEE Computer Society.

Bang, J.-Y., & Kim, Y.-D. (2010). Hierarchical production planning for semiconductor wafer fabrication based on linear programming and discrete-event simulation. *IEEE Transactions on Automation Science and Engineering*, *7*(2), 326–336.

Bratu, S., & Barnhart, C. (2006). Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, *9*(3), 279–298.

Chang, K.-H., Hong, L. J., & Wan, H. (2013). Stochastic trust-region response-surface method (STRONG) - a new response-surface framework for simulation optimization. *INFORMS Journal on Computing*, *25*(2), 230–243.

Civil Aviation Authority. (2015). *Your Rights When You Fly.* Accessed March 20th 2020. `https://www.caa.co.uk/Passengers/Resolving-travel-problems/Delays-cancellations/Your-rights/Your-rights-when-you-fly/`.

Clausen, J., Larsen, A., Larsen, J., & Rezanova, N. J. (2010). Disruption management in the airline industry - concepts, models and methods. *Computers and Operations Research*, *37*(5), 809–821.

Conn, A. R., Gould, N. I. M., & Toint, P. L. (1988). Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, *25*(2), 433–460.

Conn, A. R., Gould, N. I. M., & Toint, P. L. (2000). *Trust region methods.* Philadelphia: Society for Industrial and Applied Mathematics.

EUROCONTROL. (2020). *CODA digest - annual report for 2019: All-causes delay and cancellations to air transport in europe.*

Federal Aviation Administration. (2020). *Cost of delay estimates: 2019.* `https://www.faa.gov/data_research/aviation_data_statistics/media/cost_delay_estimates.pdf`.

Flightradar24 AB. (2017). *Flightradar24.* Accessed March 7th, 2017. `https://www.flightradar24.com`.

Guimarans, D., Arias, P., & Mujica Mota, M. (2015). Large neighbourhood search and simulation for disruption management in the airline industry. In M. Mujica Mota et al. (Ed.), *Applied simulation and optimization: In logistics, industrial and aeronautical practice* (pp. 169–201). Cham: Springer International Publishing.

Gurobi Optimization, LLC. (2017). *Gurobi Optimizer Reference Manual.* Accessed July 23rd, 2018. `http://www.gurobi.com`.

Haimes, Y., Lasdon, L. S., & Wismer, D. A. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, *1*(3), 296–297.

Hu, Y., Liao, H., Zhang, S., & Song, Y. (2017). Multiple objective solution approaches for aircraft rerouting under the disruption of multi-aircraft. *Expert Systems With Applications*, *83*, 283–299.

Huang, D., Allen, T. T., Notz, W. I., & Miller, R. A. (2006). Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, *32*(5), 369–382.

Hutchison, D. W., & Hill, S. D. (2001). Simulation optimization of airline delay with constraints. In B. A. Peters et al. (Ed.), *Proceedings of the 2001 Winter Simulation Conference* (pp. 1017–1022). Piscataway, New Jersey: IEEE.

Inanlouganji, A., Pedrielli, G., Fainekos, G., & Pokutta, S. (2018). Continuous simulation optimization with model mismatch using gaussian process regression. In M. Rabe et al. (Ed.), *Proceedings of the*

2018 Winter Simulation Conference (pp. 2131–2142). Piscataway, New Jersey: IEEE.

Izady, N., & Worthington, D. (2012). Setting staffing requirements for time dependent queueing networks: The case of accident and emergency departments. *European Journal of Operational Research*, *219*(3), 531–540.

Jeng, C.-R. (2012). Real-time decision support for airline schedule disruption management. *African Journal of Business Management*, *6*(27), 8071–8079.

Jian, N., & Henderson, S. G. (2015). An introduction to simulation optimization. In L. Yilmaz et al. (Ed.), *Proceedings of the 2015 Winter Simulation Conference* (pp. 1780–1794). Piscataway, New Jersey: IEEE.

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

Kleijnen, J. P. C. (2014). Response surface methodology. In M. C. Fu (Ed.), *Handbook of simulation optimization* (Vol. 216, pp. 81–104). New York: Springer. (Retrieved from `https://ebookcentral.proquest.com` (visited 24/12/2019))

Kohl, N., Larsen, A., Larsen, J., Ross, A., & Tiourine, S. (2007). Airline disruption management - perspectives, experiences and outlook. *Journal of Air Transport Management*, *13*(3), 149–162.

Laumanns, M., Thiele, L., & Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, *169*(3), 932–942.

Lee, L. H., Huang, H. C., Lee, C., Chew, E. P., Wikrom, J., Yong, Y. Y., ... Banks, J. (2003). Discrete Event Simulation Model for Airline Operations: SIMAIR. In S. Chick et al. (Ed.), *Proceedings of the 2003 Winter Simulation Conference* (pp. 1656–1662). Piscataway, New Jersey: IEEE.

Liang, Z., Xiao, F., Qian, X., Zhou, L., Jin, X., Lu, X., & Karichery, S. (2018). A column generation-based heuristic for aircraft recovery problem with airport capacity constraints and maintenance flexibility. *Transportation Research Part B: Methodological*, *113*, 70–90.

Løve, M., Sørensen, K. R., Larsen, J., & Clausen, J. (2005). Using heuristics to solve the dedicated aircraft recovery problem. *Central European Journal of Operations Research*, *13*(2), 189–207.

Maher, S. J. (2016). Solving the integrated airline recovery problem using column-and-row generation. *Transportation Science*, *50*(1), 216–239.

Meyer, R. K., & Nachtsheim, C. J. (1995). The coordinate-exchange algorithm for constructing exact optimal experimental designs. *Technometrics*, *37*(1), 60–69.

Montgomery, D. C. (2009). *Design and analysis of experiments* (7th ed.). John Wiley and Sons (Asia) Pte Ltd.

Onggo, B. S., Panadero, J., Corlu, C. G., & Juan, A. A. (2019). Agri-food supply chains with stochastic demands: A multi-period inventory routing problem with perishable products. *Simulation Modelling*

Practice and Theory, *97*, 101970.

Osorio, C., & Chong, L. (2015). A computationally efficient simulation-based optimization algorithm for large-scale urban transportation problems. *Transportation Science*, *49*(3), 623–636.

Rhodes-Leader, L. A., Onggo, B. S., Worthington, D. J., & Nelson, B. L. (2018). Multi-fidelity simulation optimisation for airline disruption management. In M. Rabe et al. (Ed.), *Proceedings of the 2018 Winter Simulation Conference* (p. 2179-2190). Piscataway, New Jersey: IEEE.

Rosenberger, J. M., Johnson, E. L., & Nemhauser, G. L. (2003). Rerouting aircraft for airline recovery. *Transportation Science*, *37*(4), 408–421.

Rosenberger, J. M., Schaefer, A. J., Goldsman, D., Johnson, E. L., Kleywegt, A. J., & Nemhauser, G. L. (2000). SimAir: A stochastic model of airline operations. In J. A. Joines et al. (Ed.), *Proceedings of the 2000 Winter Simulation Conference* (pp. 1118–1122). Piscataway, New Jersey: IEEE.

Rosenberger, J. M., Schaefer, A. J., Goldsman, D., Johnson, E. L., Kleywegt, A. J., & Nemhauser, G. L. (2002). A stochastic model of airline operations. *Transportation Science*, *36*(4), 357–377.

Sinclair, K., Cordeau, J.-F., & Laporte, G. (2014). Improvements to a large neighborhood search heuristic for an integrated aircraft and passenger recovery problem. *European Journal of Operational Research*, *233*(1), 234–245.

The AnyLogic Company. (2017). *Anylogic 8.2.3.* Accessed July 23rd, 2018. `http://www.anylogic.com`.

Thengvall, B. G., Bard, J. F., & Yu, G. (2000). Balancing user preferences for aircraft schedule recovery during irregular operations. *IIE Transactions*, *32*(3), 181–193.

Wang, D., Wu, Y., Hu, J.-Q., Liu, M., Yu, P., Zhang, C., & Wu, Y. (2019). Flight schedule recovery: A simulation-based approach. *Asia-Pacific Journal of Operational Research*, *36*(6), 1–19.

Welch, B. L. (1938). The significance of the difference between two means when the population variances are unequal. *Biometrika*, *29*(3), 350–362.

Xu, J., Zhang, S., Huang, E., Chen, C.-h., Lee, L. H., & Celik, N. (2016). MO^{2}TOS: Multi-fidelity optimization with ordinal transformation and optimal sampling. *Asia-Pacific Journal of Operational Research*, *33*(3), 1650017:1–26.

Zhang, D., Henry Lau, H. Y. K., & Yu, C. (2015). A two stage heuristic algorithm for the integrated aircraft and crew schedule recovery problems. *Computers and Industrial Engineering*, *87*, 436–453.

Zhu, B., Zhu, J.-f., & Gao, Q. (2015). A stochastic programming approach on aircraft recovery problem. *Mathematical Problems in Engineering*, *2015*, 1–9.

Table 1. IP solutions for Problems 1, 2 and 3. Delays are measured in minutes, and the solution time is measured in seconds.

Problem	Max. Delay M	Time Step m	Solution	Cost (€1000)	Aircraft Exchanges	Total Delay	Max. Delay in Solution	Cancelled Flights	Solution Time
1	180	5	1	15.00	30	300	95	0	4.23
			2	15.00	15	300	95	0	15.67
			3	15.75	12	315	95	0	12.61
			4	16.50	8	330	95	0	25.66
			5	25.25	4	495	125	0	18.71
2	600	15	8	6.00	19	120	60	0	441.51*
			9	6.25	17	135	60	0	65.06
			10	9.75	15	195	60	0	90.28
			11	9.75	14	195	90	0	62.00
			12	11.06	13	195	120	0	76.96
			13	12.75	12	255	90	0	104.75
3	60	20	12	27.00	17	540	60	0	2131.90*
			13	29.00	13	580	60	0	67.41
			14	31.00	9	620	60	0	72.30
			15	33.00	8	660	60	0	61.63
			16	35.00	4	700	60	0	447.72

* Cumulative time of multiple iterations of the multi-objective algorithm producing cost optimal solutions but with more aircraft exchanges than the Pareto optimal solutions.

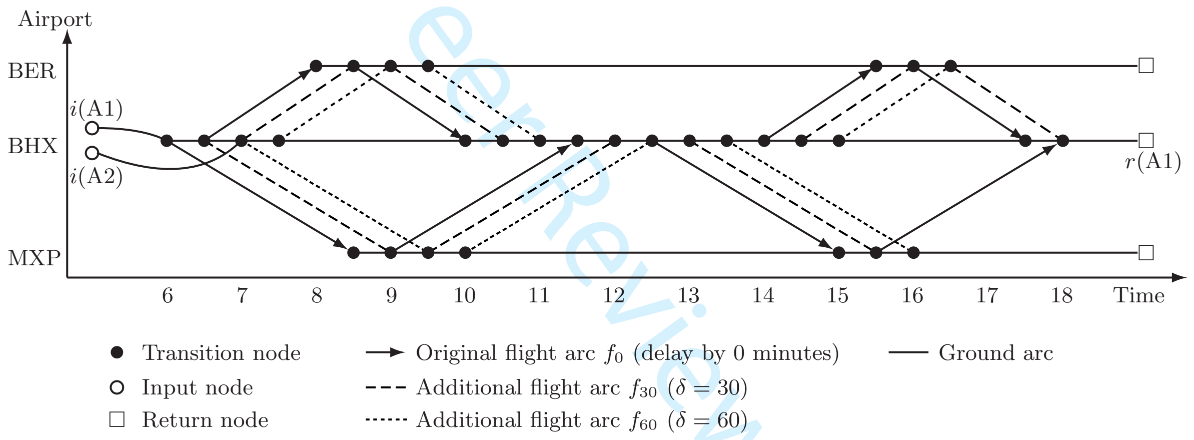
Airport
BER
BHX
MXP
i(A1)
i(A2)
r(A1)
6
7
8
9
10
11
12
13
14
15
16
17
18
Time
● Transition node
○ Input node
□ Return node
Original flight arc f_0 (delay by 0 minutes)
Additional flight arc f_{30} ($\delta = 30$)
Additional flight arc f_{60} ($\delta = 60$)
Ground arc

Figure 1. An example of the time-space network used by the IP under a disruption, $m = 30$ minutes and $M = 1$ hour.

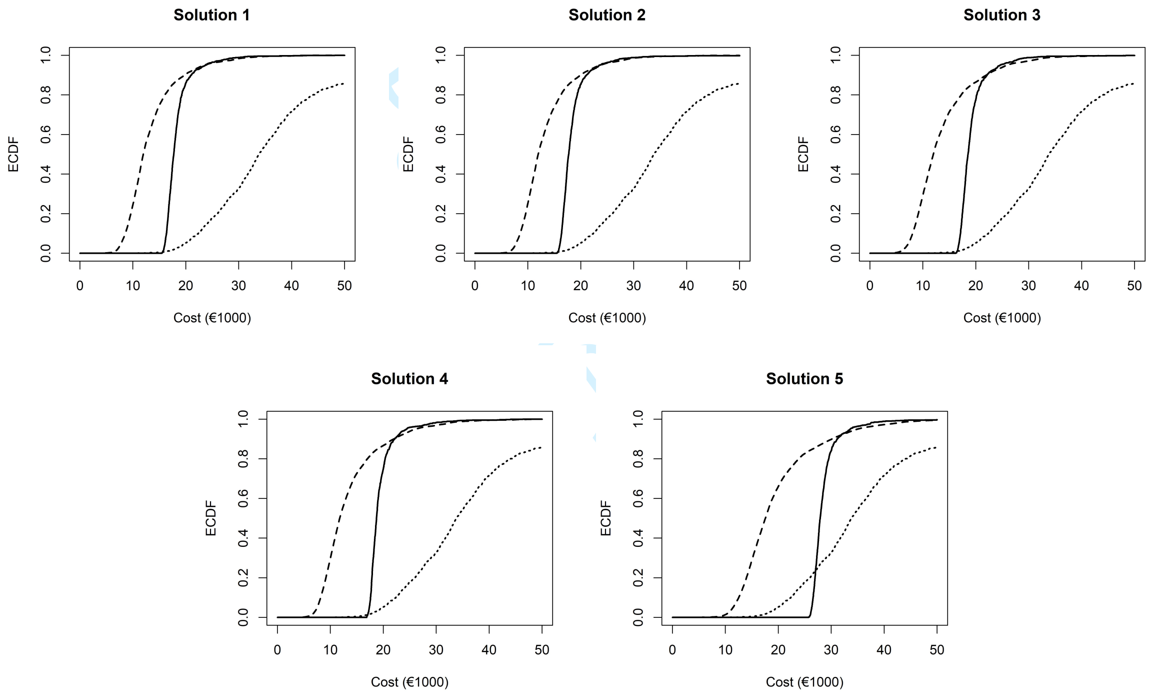

Figure 2. ECDFs for each starting solution in Problem 1. Solid line is the IP solution, dashed line is the combined simulation optimisation solutions, dotted line is 'No Action'.

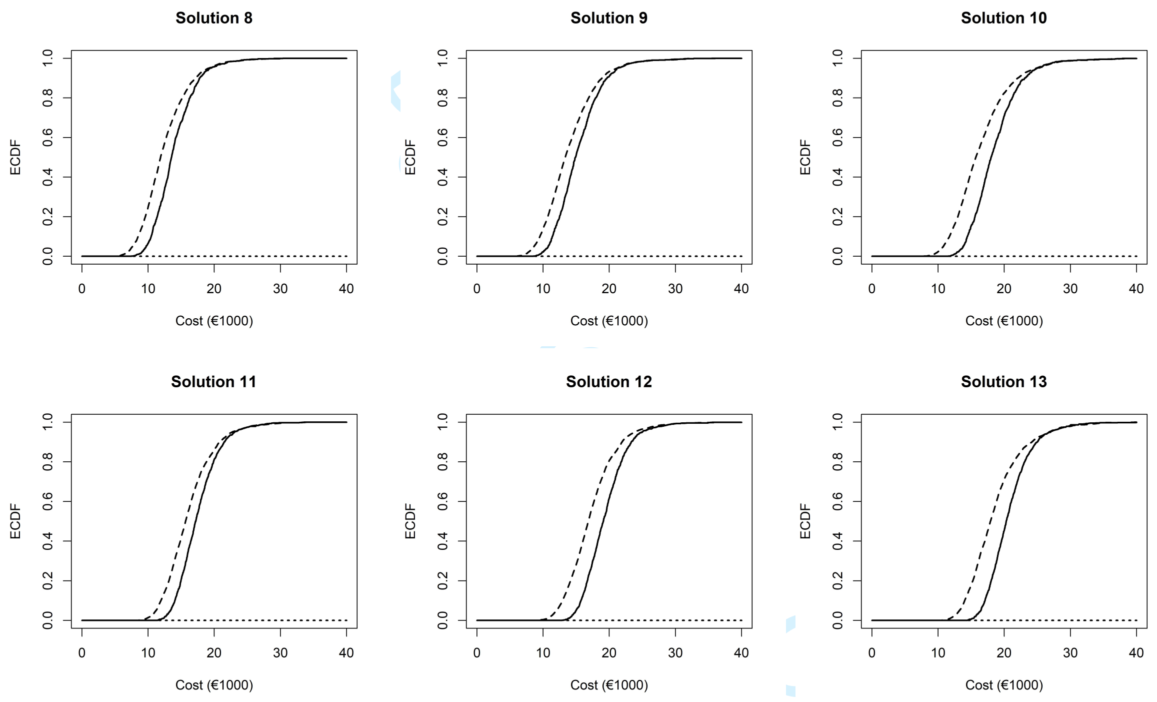

Figure 3. ECDFs for each starting solution in Problem 2. Solid line is the IP solution, dashed line is the combined simulation optimisation solutions, dotted line is 'No Action'.

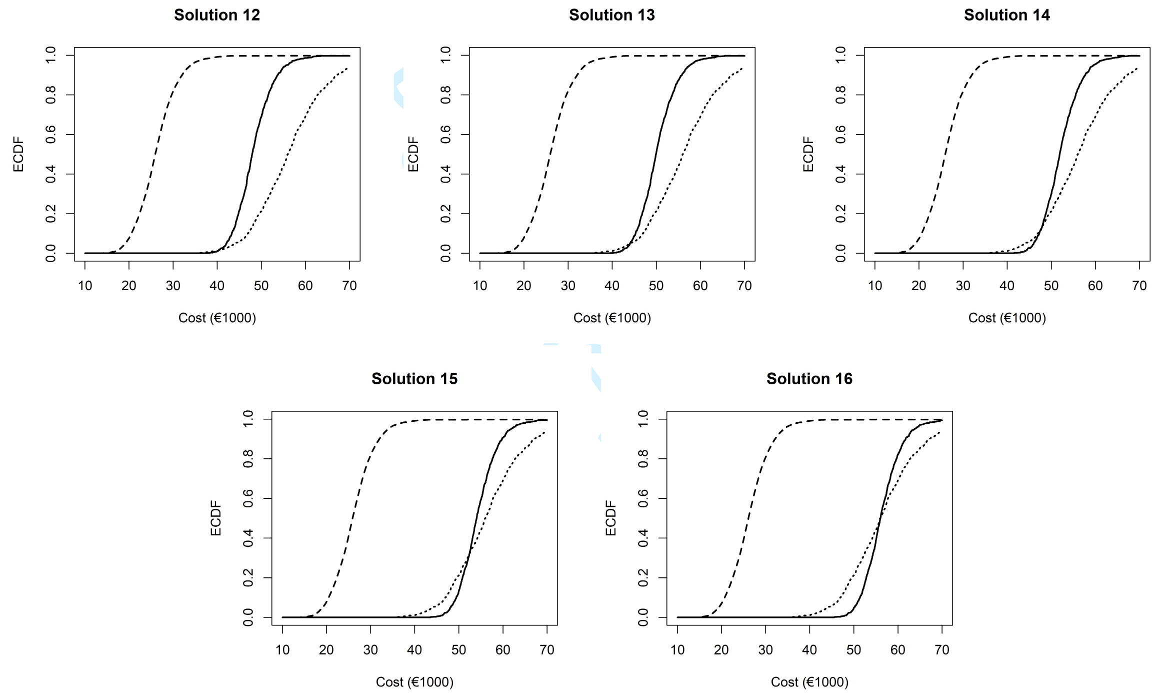

Figure 4. ECDFs for each starting solution in Problem 3. Solid line is the IP solution, dashed line is the combined simulation optimisation solutions, dotted line is 'No Action'.

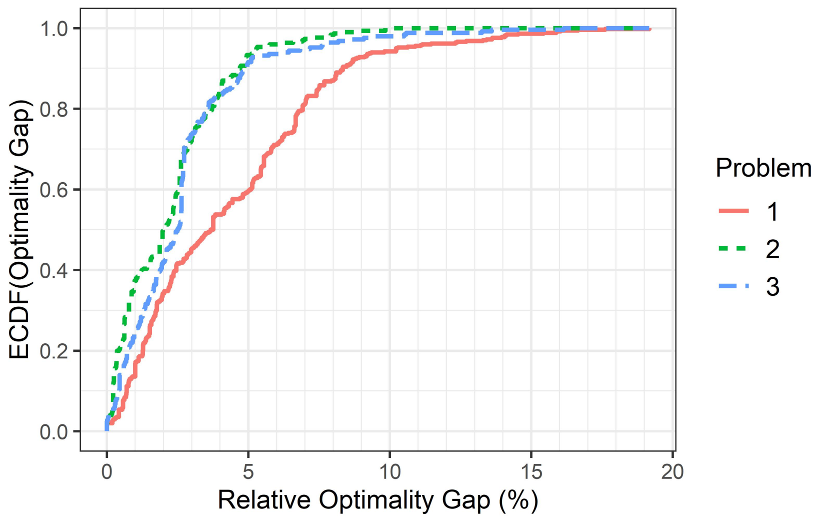

Figure 5. ECDFs of the Relative Optimality Gap for each Problem.

List of Figures

Supplementary material for: A multi-fidelity modelling approach for airline disruption management using simulation

L. A. Rhodes-Leader[a], B. L. Nelson[a], B. S. Onggo[b] and D. J. Worthington[a]
[a]Lancaster University, UK; [b]Southampton University, UK

Contents

1 Choosing the Limits in the ϵ-Constraint Method

When solving the IP formulated in Sections 4.2 and 4.3 of the main text, one must have a way to alter the limits of the constraints created by the delay and aircraft exchanges objectives, denoted l_D, u_D, l_E and u_E:

$$\sum_{a \in A} \sum_{f_\delta \in L} (1 - o_a^f) x_a^{f_\delta} \in [l_E, u_E] \tag{1}$$

$$\sum_{a \in A} \sum_{f_\delta \in L} \delta x_a^{f_\delta} \in [l_D, u_D]. \tag{2}$$

We implement the method proposed by Laumanns et al. (2006), and refer to it as Laumanns' algorithm. Here we give a brief description of this algorithm.

Initially, the IP is solved without (1) and (2) constraining the problem. This is equivalent to defining $l_D = l_E = 0$ (that is, no delays and no exchanges) and $u_D = nM$ (all flights delayed by the maximum amount) and $u_E = n$ (no flights operated using their original aircraft). The result is a cost optimal solution, $(\mathbf{x}_1, \mathbf{d}_1)$. Suppose that this solution has a delay of D_1 and E_1 exchanges. These values are used to partition the objective space into four sectors:

$$[0, D_1) \times [0, E_1), \qquad [D_1, nM] \times [0, E_1), \qquad [0, D_1) \times [E_1, n], \qquad [D_1, nM] \times [E_1, n].$$

Note that as $(\mathbf{x}_1, \mathbf{d}_1)$ is cost optimal, no solution lying the $[D_1, nM] \times [E_1, n]$ sector can dominate $(\mathbf{x}_1, \mathbf{d}_1)$. Therefore, the algorithm will not choose values of (l_D, u_D, l_E, u_E) that overlap with this sector, so $[D_1, nM] \times [E_1, n]$ is considered searched. Laumanns' algorithm now chooses one of the three unsearched sectors (such as $[D_1, nM] \times 0, E_1)$) to define (l_D, u_D, l_E, u_E), applies these limits to (1) and (2) and solves the IP again to find a new solution, $(\mathbf{x}_2, \mathbf{d}_2)$. The new solution's delay, D_2, and number of exchanges, E_2, will be used to partition the objective space into (up to) 9 sectors. Using similar reasoning to the above, the sector $[E_2, u_E) \times [D_2, u_D)$ is considered searched. The process of searching a sector for a solution $(\mathbf{x}_i, \mathbf{d}_i)$, using (D_i, E_i) to generate a smaller partitioning of the objective space, and blocking off searched sectors continues until the whole objective space is marked as searched or a time limit is reached.

Laumanns' algorithm is designed to find all Pareto optimal solutions, without a time limit. In our implementation, the choice of the next sector prioritises reducing aircraft exchanges. Only when this is not possible are the delay constraints made tighter. In addition, the original limit of $u_E = n$ may be considered too weak by an airline, so we suggest using smaller bound. In our experiments, this was set to $u_E = 50$.

2 Simulation Model Overview

The simulation model is built within AnyLogic 8.2.3 (The AnyLogic Company, 2017), a Java based simulation software. It simulates a sub-fleet of homogeneous aircraft operating the recovery action $(\mathbf{x}, \mathbf{d})$ over a set of airports. Each aircraft follows its assignment of the schedule, subject to stochastic flight durations, turn times, queueing times and maintenance. Its general framework is largely based on the SimAir simulation as discussed in Lee et al. (2003). However, it does not consider crew members or passengers (except in calculating passenger compensation).

The data set used for a number of the distributions and schedule creation was sourced from Flightradar24 AB (2017) and obtained using the python package (Allamraju, 2017). It contains information about over 12 million flights in the period 1st November 2016 to 1st April 2017. The information consists of origin, destination, aircraft type and identification, airline, departure and arrival times. Not all entries are complete. In approximately half of the flights, the data also contains scheduled departure and arrival times. For the problems discussed in Section 6 of the paper, 2.8 million flights were used, departing or arriving at one of 163 European airports, many of which appear in the schedule.

The duration of a flight from airport A to airport B is modelled using a log-logistic distribution with parameters fitted using the Maximum Likelihood Estimator (MLE) from the observations of the flight from A to B within the available data (Flightradar24 AB, 2017). This is done for each route that appears in the schedule.

The turn times are assumed to follow a left-truncated Normal distribution. Unlike in the IP, the mean, variance and minimum can vary between airports as well as whether the aircraft requires refuelling. These could be chosen using information on typical turn times for an aircraft from technical documents, such as Airbus (2019). For the short-haul flights considered here, it is assumed that refuelling occurs after two flights. These distribution and parameter assumptions are made as our data source does not track this information. The gate departure time of the aircraft is the maximum of the ready time (after the turn time) and the planned departure time (including the planned delay) according to the input schedule $(\mathbf{x}, \mathbf{d})$. When this time occurs, the aircraft joins the take-off queue.

It is assumed that each airport has two runways with independent segregated operations, i.e., one runway for landings and one runway for departures with the distance between runways sufficiently large to assume that minimum aircraft separation is guaranteed between the two queues (ICAO, 2016). The landing and take-off queues for use of the runways are modelled as $G(t)/D(t)/1$ queues. The arrival process of aircraft outside the airline's fleet to these queues is a deviation-from-schedule model based on the published arrivals and departures schedule within the available data. The departure deviation is modelled using a shifted log-logistic distribution, whilst the arrival deviation is assumed to be Normally distributed, both use MLE parameters estimated from the observed data for the airport, for flights where both the scheduled and actual times are recorded. The service time represents the spacing required between aircraft and is assumed deterministic given the weather conditions and time of day.

At the beginning of the simulation, each aircraft is given a time-to-failure based on the time since its last scheduled maintenance. Only faults that would lead to an aircraft being grounded are considered. The time-to-failure is sampled from a Weibull distribution conditioning on the flying hours accumulated since its previous maintenance checks. Once an aircraft's flying time has exceeded this time-to-failure, it enters the maintenance hangar at the next airport where it lands. The time to complete unplanned maintenance is assumed to come from a Gamma distribution and must be completed before the next flight. The parameters can vary from airport to airport depending on the facilities available. For example, the expected time for unplanned maintenance at a hub airport should be shorter than that of an outstation airport where the airline will have fewer resources. Planned maintenance is also part of the schedule and is assumed to take a fixed amount of time.

Initial conditions for each aircraft in the fleet is based on whether that aircraft is at an airport or in flight. In either case, time before arrival or departure is sampled from the distributions, conditioned on the time already taken in the activity.

Table 1: Parameters and variables for the Aircraft entity.

Parameter	Type	Description
airline	boolean	Is the aircraft part of the fleet?
id	string	Aircraft tail number
flightTime	double	Flight time (minutes) since last maintenance
flightStart	double	Time at which last flight began
flightNum	integer	Index of current flight in the arrays
leg	String[2]	OD pair of airport names for current flight
failure	double	Failure time of the aircraft (generated at beginning)
maintenance	integer	1 if the aircraft needs maintenance, 0 otherwise
planned	integer	1 if the maintenance is scheduled, 0 otherwise
minutes	double	Scheduled flight time (minutes) during period
CheckTime	double	Time of next scheduled maintenance
refuel	integer	1 if aircraft requires fuel after flight, 0 otherwise
inFlight	integer	1 if aircraft in flight at initiation, 0 otherwise
start	double	Time when aircraft began the current activity
RNG	java.util.Random	Specific random number stream for this aircraft
origins	ArrayList<String>	List of origins in schedule
destinations	ArrayList<String>	List of destinations in schedule
takeoffs	ArrayList<double>	List of scheduled departure times in schedule
landings	ArrayList<double>	List of scheduled arrival times in schedule
delays	ArrayList<double>	List of planned delays in schedule
passengers	ArrayList<integer>	List of passenger numbers on each flight in schedule

3 Model Details

This section gives more details of the simulation model. Throughout, a variable type followed by [ ] (such as String[ ] or double[2]) will refer to a Java array of that type, using the Java syntax. The length may or may not be specified. Note that the specific parameters used within the problems are discussed in Section 5.1 and are available within the online data files connected to the paper.

3.1 Aircraft Entity

The main entity (called an "agent" in AnyLogic) is the Aircraft. The parameters and variables of the Aircraft are given in Table 1. The "airline" parameter is true for all aircraft in the fleet, and false for all other aircraft. Other parameters may not be initialised for aircraft outside the fleet. The "inFlight" and "start" parameters are used to determine initial conditions: is the aircraft in flight at the start of the period, and how long has it been in flight or on the ground. The "start" parameter allows the initial activity times to come from a conditional distribution. To increase the correlation when using Common Random Numbers (CRN), each aircraft has its own random number generator object. This is used for generating the failure time ("failure" parameter), turn times, repair times and flight durations. The seed is set when the entity is initialised using the next random integer from the default random number generator of the simulation. As well as information about the aircraft, the schedule information is stored in arrays as parameters of the Aircraft class. AnyLogic stores data in SQL databases, which are quite slow to search. Storing the schedule information in arrays as variables of the Aircraft class decreases the computation time.

Many of the parameters in Table 1 are read in from a database upon initialisation. Others are calculated from the schedule information. The schedule ArrayLists of the aircraft are populated from the schedule SQL database, storing only flights that are allocated to that aircraft. To remove the need for extra rules when the aircraft has completed its schedule, a pseudo-flight is added to the list, with departure after the simulation end time. The sequence of flights is checked for feasibility, ensuring that the destination of one flight matches the origin of the next. If a sequence is found to be infeasible, the simulation terminates. This prevents errors occurring if an infeasible schedule is mistakenly used as an input. If an aircraft has no assigned schedule, it is removed from the population of aircraft.

The failure time for each aircraft is computed on initialisation. For the purposes of this study, it is assumed that failures resulting in immediate grounding follow a Weibull distribution with shape parameter α=10 and scale parameter β=30,000. No other faults are modelled in the simulation. As the time-to-failure is conditioned on being larger than the period of time previously flown, 'flightTime', the failure time is sampled from a conditional distribution. Rather than rejection sampling, this can be done using the procedure in Algorithm 1. The right-hand side of Equation (3) is the Weibull(α,β) Cumulative Distribution Function (CDF) evaluated at 'flightTime'. The right-hand side of Equation (4) is the inverse CDF of the same distribution. The proof that this creates the appropriate conditional distribution is straightforward, but not included here.

Algorithm 1 Generating the failure time of an aircraft

```
1: Calculate the probability of failing before 'flightTime':
       $$u_0 = 1 - \exp\left\{-\left(\frac{\text{flightTime}}{\beta}\right)^{\alpha}\right\} \qquad (3)$$
2: Sample a uniform random number: $U \sim U(u_0, 1)$
3: Calculate failure time using the Inverse Probability Integral Transformation:
       $$\text{failure} = \beta(-\log(1-U))^{1/\alpha} \qquad (4)$$
4: return failure
```

3.2 Main

In the simulation model, the aircraft move in two environments. One is the Airport, which is discussed in Section 3.8. The other is the Main 'agent', in which the flights are operated, populations are initialised and output statistics are collected. The parameters are listed in Table 2. The cost parameters are used to calculate the objective function. The purposes of these will be explained in the appropriate context in the following subsections.

3.3 Flights

Once the aircraft has been served by the airport take-off runway, the aircraft leaves the airport environment (an agent in AnyLogic) and enters the simple flow chart in the Main agent. As the SQL search in AnyLogic is slow, a "flight" class is used, with four attributes: origin, destination, shape parameter, α, and scale (distScale) parameter, β. These are stored in an AnyLogic agent population, which is quick to search when finding the appropriate distribution parameters. The flight durations are modelled using a log-logistic distribution, with Maximum Likelihood Estimator

Table 2: Main Parameters and Variables

Parameter	Type	Description
delayCostpm	double	Cost of delay per minute
exceedancePenalty	double	Cost per minute of delay exceeding planned delay
delay2hrCost	double	Compensation per passenger for 2 hour departure delay
delay3hrCost	double	Compensation per passenger for 3 hour arrival delay
cancelCost	double	Compensation per passenger for a cancellation
fixedCancelCost	double	Non passenger-based cost for a cancellation
diversionCost	double	Cost for diverting a flight
PlanNumber	integer	Label of the schedule to import for the simulation
stopTime	double	End of the simulated period
numFlights	integer	Number of flights in the schedule
disruptedAirport	String[ ]	List of airports with weather disruptions
keepPast	double	Length of history still in the queueing system
closedAirports	ArrayList	List of currently closed airports
cost	double	Output variable used to accumulate cost

(MLE) parameters for each route. The data used to fit the parameters (from Flightradar24 AB (2017)) is the time from take-off to landing. Thus, the service times of both runways, SpacingDep(t) and SpacingArr(t), are removed from the total. So the flight duration is

$$\text{duration} = \alpha \left(\frac{U}{1-U} \right)^{1/\beta} - \text{SpacingDep}(t) - \text{SpacingArr}(t), \tag{5}$$

where $U \sim U(0,1)$ and the first term is the inverse CDF of the log-logistic distribution.

If an aircraft is in flight at initialisation, its remaining flight time is sampled from a conditional distribution based on the time already spent in the air. The sampling uses the same process as the conditional failure times in Algorithm 1, replacing the Weibull CDF with the log-logistic CDF, and "flightTime" with the time already used.

At the end of the flight duration, the simulation identifies the destination airport environment and the aircraft is sent to that environment.

3.4 Runways

The simulation assumes that segregated runway operations are used at each airport (one runway for arrivals and one for departures) and that they operate independently. The functions that represent the minimum legal separation between aircraft using runways are held in Main (as they must be accessed when calculating the flight duration). Both functions, SpacingDep(t) and SpacingArr(t), are step functions. For the purposes of this research, SpacingDep(t) and SpacingArr(t) are set to 1.1 and 1.2 minutes between 6am and 11pm, respectively, and 3 minutes overnight.

3.5 Refuelling Event

An event is scheduled overnight to ensure that all aircraft start the next day refuelled. As the aircraft "refuel" parameter is updated on leaving the gate, this event sets "refuel" to 1 for each aircraft in the fleet. This is in addition to the refuelling that occurs after every second flight.

Table 3: Airport Parameters and Variables

Parameter	Type	Description
name	String	IATA name for the airport
fixShape	integer	Shape parameter for repair time distribution
fixScale	double	Scale parameter for repair time distribution
acheck	double	Hours to complete scheduled maintenance
turnSTD	double	"Standard deviation" of turn time distribution
turnShort	double	"Mean" of turn time distribution (no refuelling)
turnLong	double	"Mean" of turn time with refuelling distribution
turnTime	double[2]	Array of turnShort and turnLong
minTurnTime	double[2]	Array of minimum turn times
runways	integer	Number of runways for each queue
currentWeather	double	Additional separation time due to poor weather
forecastTime	ArrayList<double>	Time of weather change
forecastRate	ArrayList<double>	Additional separation at times in forecastTime
arrivals	ArrayList<double>	Arrival times to airport from other aircraft
departures	ArrayList<double>	Departure times from airport of other aircraft

3.6 Output

The primary output of the simulation is cost. This is added to at various events during the simulation. Other output statistics, collected in AnyLogic Data Sets, are: arrival delay of each flight, departure delay of each flight, whether the delay is greater than 15 minutes (thus affecting on-time performance) and whether an aircraft is late for scheduled maintenance. An AnyLogic Statistics object stores summary statistics of all the outputs.

3.7 Initialisation

The Main environment controls the initialisation of the simulation, including the fleet and airport populations which are described in the relevant sections.

The compensation costs from all cancelled flights are also calculated at this point. The cost for each flight is "fixedCancelCost" plus "cancelCost" for each passenger on the flight. These are added to the "cost" variable.

3.8 Airport Operations

The other environments the aircraft entities operate in are the airports. These are generated from a database at initialisation. The environment is a discrete event flow chart. This section details the processes at the airport.

Each airport has a set of parameters. These describe various properties related to the airport operations and the distributions for activity durations. The parameters are listed in Table 3. In principle, they could vary across airports and be read in from an input file. In this research, some are set throughout, including "turnSTD" and "fixScale"; some are read in via a database, such as "acheck" and "turnShort"; and some are initialised depending on other input data, such as the ArrayLists.

3.9 Runway Resources

The Airport environment has two resource entities: "landingRunway" and "takeoffRunway". These resources serve the landing and take-off queues, respectively. The capacity of each resource is 1, as each airport is assumed to use one runway for arrivals and one for departures. These operate independently of each other.

3.10 Path through Airport for our Aircraft

Figure 1 shows the process an aircraft goes through in the airport, and where aircraft from other airlines interact with this aircraft (in the landing and take-off queues). This process occurs after an aircraft has completed the flight. On initialisation, when the "flightNum" parameter is -1, the aircraft bypasses the first section and goes straight to the "Aircraft requires maintenance?" decision block. The following subsections consider parts of this flow diagram in more detail.

3.11 Arrivals to the Queues

The arrival processes to the landing and take-off queues are a deviation-from-a-schedule process. On initialisation, the arrival and departure schedules at that airport are read from the SQL database into the "arrivals" and "departures" ArrayLists of the airport environment. For most airports, only flights that are scheduled after the start time are included. However, for those airports listed in the array "disruptedAirport" in Main, a history of length "keepPast" is included. Any aircraft that entered the system less than "keepPast" minutes ago, is expected to still be in the queueing system. The longer "keepPast" is, the greater the congestion levels. Ideally, the simulation would use real-time information to set the initial conditions, rather than this mechanism.

Once the schedule has been read in, the deviations are calculated. For the arrivals, a normal random variable is added. The parameters are fitted using MLEs based on the differences between the scheduled arrival and actual arrival in the data (where available) from Flightradar24 AB (2017). For the departures, the deviation is modelled using a shifted log-logistic distribution. The shift is set as the smallest observation, whilst the scale and shape parameters maximise the likelihood of the differences between the scheduled and actual departure times in the data. All of these distributions are modelled individually for each airport, i.e. only arrivals or departures from that airport are used to fit the parameters. If the deviation means that the flight arrived or departed before the current time (or more than "keepPast" minutes before the current time for disrupted airports) it is removed from the ArrayList. The lists are then sorted to ease the search for the next arrival.

Once the "arrivals" and "departures" ArrayLists are populated and ordered, the inter-arrival time is the difference between the current time and the first element in the ArrayList (or a small positive number if the difference is negative, as it may be at the start of the simulation). This is used to schedule the next arrival event, and then the element is deleted from the ArrayList. If the list is empty, an infinite inter-arrival time is set, so that no more arrival events are scheduled. All aircraft generated by this mechanism have the parameter "airline" set to false.

3.12 Runway Service Times

The service time represents the spacing required between aircraft and is assumed deterministic given the weather conditions and time of day, taking the form

$$Q(t) = \frac{s(t)}{w(t)}, \tag{6}$$

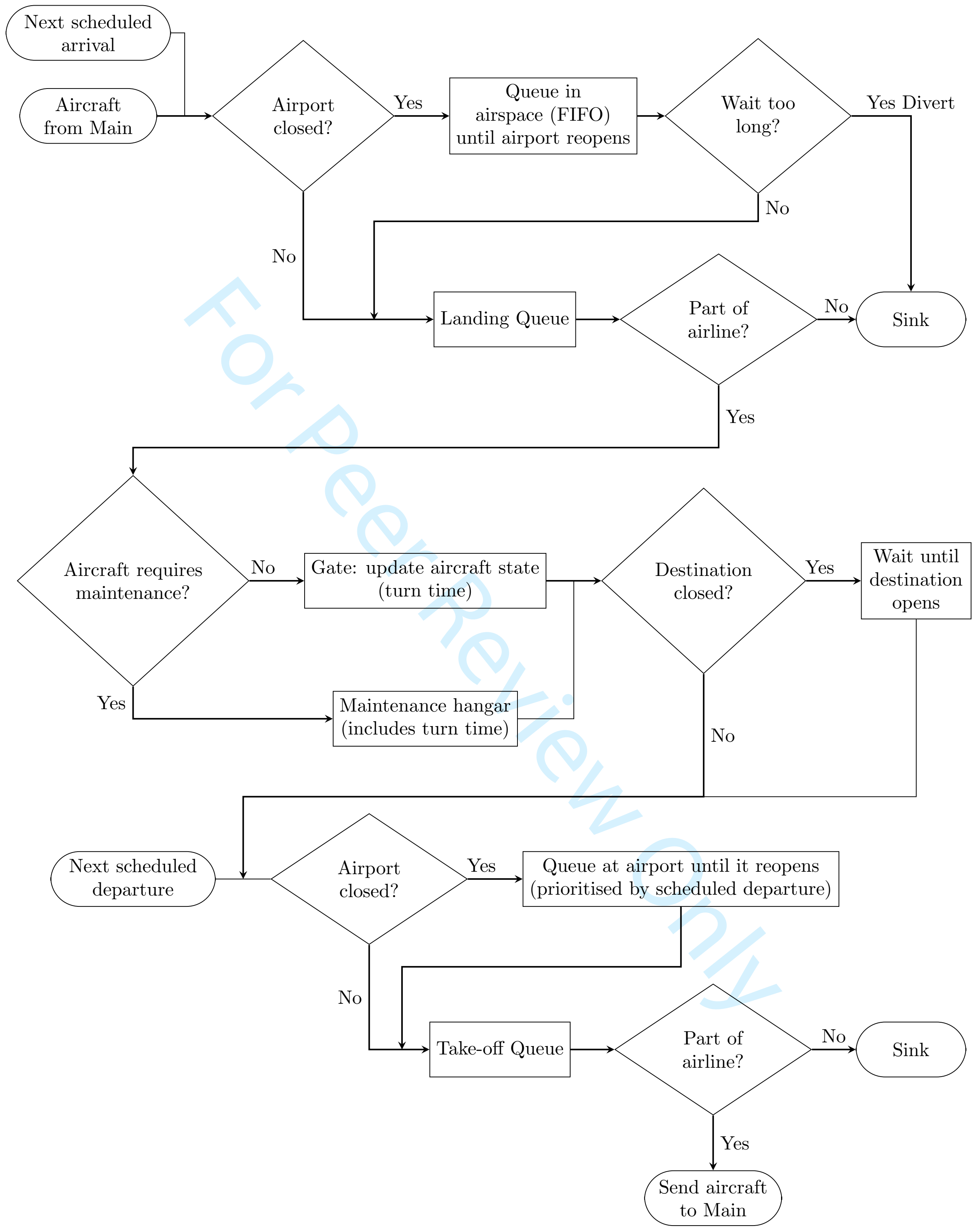

Figure 1: Flow diagram of the path taken by an Aircraft through the Airport environment in the simulation.

where $s(t)$ is the required separation time in normal operating conditions at time t (either SpacingDep(t) or SpacingArr(t)) and $w(t) \in (0, 1]$ is a scaling factor to account for weather conditions at time t. The aircraft separation $s(t)$ is a step function, increasing overnight (11pm to 6am) due to noise pollution constraints. The weather conditions follow a step-function forecast at each airport, with poor weather conditions leading to increased aircraft spacing.

At the beginning of the service period of the take-off queue, the aircraft's "flightStart" parameter is set to the current time. At the end of the landing queue service period, the difference between the current time and "flightStart" is added to the "flightTime" parameter to update the amount of flying since the last maintenance of the aircraft.

After the service time, the aircraft is checked to see if it belongs to the airline, using the "airline" parameter. If not, the entity is sent to a sink.

3.13 Weather Change

The weather conditions, $w(t)$, are represented by the airport variable "currentWeather", which gives the additional separation time required between aircraft using the same runway (1 refers to normal operations). This variable changes in a discrete manner according to a forecast. The forecast is read from a database at the beginning of the simulation. The times of the changes in the forecast and the corresponding values are stored in the "forecastTime" and "forecastRate" ArrayLists, respectively. The change is controlled by an event, "weatherChange", initially scheduled for soon after the start time. The algorithm performed at the event is shown in Algorithm 2.

Algorithm 2 Change in weather event algorithm

```
1: Get current time $t$
2: Find first element of forecastTime greater than $t$, with index $i$
3: Set currentWeather $\leftarrow$ forecastRate[$i-1$]
4: if currentWeather $\leq 0.1$ & landingRunway.capacity $\geq 1$ then
5:     Set landingRunway.capacity and takeoffRunway.capacity to 0
6:     Label airport as closed
7:     Add airport to closedAirports list in Main
8: end if
9: if currentWeather $> 0.1$ & landingRunway.capacity $= 0$ then
10:    Set landingRunway.capacity and takeoffRunway.capacity to normal capacity
11:    Label airport as open
12:    Remove airport from closedAirports list in Main
13:    Search through airport population and free aircraft waiting for this destination to open
14: end if
15: Schedule next weatherChange event at this airport for time forecastTimee[$i$]
```

This whole process assumes that the airline has a method for taking the weather forecast data at an airport and converting this into an additional separation.

As well as changing the service times at the runways (see Equation 6), the "currentWeather" variable also indicates whether the airport is open or closed. For the purposes of this research, we assume that if "currentWeather" goes below 0.1, an airport closes. Much of Algorithm 2 deals with this. Lines 4 to 8 deal with closing an airport that was open. This involves setting the capacity of the resources to 0, and then labelling the airport as closed, blocking the runway queues. This forces arriving aircraft to wait in the airspace. In this case, each aircraft is given a random diversion time (here 30 minutes plus an exponentially distributed random variable). If the wait exceeds this value, the aircraft will leave the queue and be diverted to another airport. Should this happen to an aircraft in the airline, a high cost of “diversionCost” is incurred. Departing aircraft wait in a queue prioritised by the scheduled departure time.

Figure 1 shows that after the turn time, the aircraft checks to see if the destination airport is in the list of closed airports. If it is, it will wait at the origin until it receives a message that the destination has reopened.

Lines 9 to 14 reopens an airport. This undoes the closing procedure, allowing all aircraft to join the runway queues. In addition, a search occurs for all aircraft currently being held on the ground at the origin, allowing the flight to begin.

3.14 Time at an Airport

Once the aircraft leave the landing queue, the “flightTime” parameter is updated and the condition of belonging to the airline is checked. Then the maintenance status of the aircraft is calculated. If the aircraft’s “CheckTime” parameter is before the next flight in the schedule ArrayLists (including the planned delay), the “planned” parameter is set to 1. Then, if either “planned” is 1 or “flightTime” exceeds the “failure” parameter, the aircraft’s “maintenance” parameter is set to 1, otherwise it remains at 0. This is the condition checked at the start of the middle section of the flowchart in Figure 1. Based on this, the aircraft enters either the Gate area or the Maintenance hangar.

The length of stay in the Gate environment, the turn time, is the minimum of a random variable from a left-truncated normal distribution (representing the operational time to prepare the aircraft for flight) and the time to the next scheduled flight (including the planned delay). To sample from the truncated normal, a random variable X is repeatedly sampled from a normal distribution with standard deviation “turnSTD” and mean μ until the X is greater than the truncation point. The parameters of the truncated normal depends on whether the aircraft requires refuelling, measured by the aircraft parameter “refuel”, as this increases the time to ready the aircraft for flight (Airbus, 2019). If the aircraft does not require refuelling, μ is “turnShort” and the first element of “minTurnTime” is used for the minimum. Otherwise, the parameters are “turnLong” and the second element of “minTurnTime”. These parameters can vary across the airports. The simulation assumes that an aircraft needs to be refuelled every other flight.

Once the turn time is calculated, the arrival delay is measured and added to the data sets of the simulation. If the arrival delay is more than 3 hours, passenger compensation is added to the cost (number of passengers $\times$ “delay3hrCost”). The compensation costs follow the rules laid out by Civil Aviation Authority (2015). The flight details of the aircraft are then updated for the next flight and the route is checked to ensure the aircraft is in the correct place (if not, the simulation terminates). On departing the gate, the delay cost (delay $\times$ “delayCostpm”), the penalty for overrunning (“exceedancePenalty” $\times$ any delay beyond the planned delay) and passenger compensation if the departure delay is greater than 2 hours (number of passengers $\times$ “delay2hrCost”) are added to the cost variable. The refuelling parameter is also updated.

The Maintenance area contains the same procedures as the Gate area. The length of stay is altered by the addition of the maintenance time to the operational time to prepare the aircraft. If the maintenance is scheduled (‘planned’=1) it is assumed to have a constant duration, given by “acheck”. The unscheduled maintenance time is assumed to come from a Gamma distribution with shape and scale parameters given by “fixShape” and “fixScale”, respectively. This distribution does not change depending on the type of failure. If the aircraft starts the simulation in maintenance, the time already used (measured by the “start” parameter) is considered, using a conditional Gamma distribution for the repair time. In addition to collecting delay statistics and updating the aircraft's flight parameters, the maintenance parameters of aircraft are updated: "maintenance" and "planned" parameters are set to 0; "flightTime" is set to 0; and a new failure time is generated from the Weibull distribution. If the maintenance was scheduled, "CheckTime" is updated to a future date (set here to be in 30 days time).

In both the Gate and Maintenance areas, the aircraft's individual random number stream is used for the sampling.

4 Details of the Simulation Optimisation

The simulation optimisation is performed using an adaptation of the STRONG algorithm proposed by Chang et al. (2013). The algorithm combines Response Surface Methodology with trust-region optimisation. It is a sequentially algorithm. In iteration j, with current solution $\mathbf{d}_j$, it approximates the objective function, $g(\mathbf{x}, \mathbf{d})$, using a local meta-model $\hat{q}_j(\mathbf{d})$ over a region, $\mathcal{B}_j$, of size Δ_j to propose a new solution, $\mathbf{d}_j^*$. This is accepted or rejected based on two tests. The meta-model is linear if $\Delta_j \geq \tilde{\Delta}$ and quadratic if $\Delta_j < \tilde{\Delta}$. In this section, we give some additional details that were left out of the main paper.

4.1 Building the Meta-model

The dimension of the simulation optimisation problem is n^+. The number of design points required to fit a linear model is $n^+ + 1$, whilst the quadratic model requires at least $2n^+ + n^+(n^+ - 1)/2 + 1$ design points. To give additional degrees of freedom when fitting the regression model, we added an extra $\lfloor n^+/2 \rfloor$ design points to each design.

The design matrix for the design in iteration j, $\mathbf{D}_j$, is built using a Coordinate Exchange Algorithm (Meyer and Nachtsheim, 1995). This algorithm sequentially moves design points around the feasible trust region, $\mathcal{D} \cap \mathcal{B}_j$, so as to maximise $\det(\mathbf{D}_j^T \mathbf{D}_j)$. The stopping criterion is when an iteration of moving all the design points produces an increase in $\det(\mathbf{D}_j^T \mathbf{D}_j)$ of less than ε_D. We chose ε_D to be 1%.

The number of observations at each design point, N_j^p, is chosen to be the smallest value satisfying:

$$\left| \frac{\hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_{j-1}) - \hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_j)}{\sqrt{S_{j-1}^2(\mathbf{d}_{j-1})/N_j^p + S_{j-1}^2(\mathbf{d}_j)/N_j^p}} \right| \geq 2 \qquad \text{subject to} \qquad N_j^p \in \{N_{\min}^p, ..., N_{\max}^p\}. \tag{7}$$

Here, $S_j^2(\mathbf{d})$ is the sample variance of the simulation observations at solution $\mathbf{d}$ in the j^{th} iteration, so the denominator is the standard deviation of the difference. The expression approximates the noise-to-signal ratio across $\mathcal{B}_j$. A small difference between $\hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_{j-1})$ and $\hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_j)$ or a large pooled variance means that more observations are required to get an adequate meta-model. The lower limit is used to ensure multiple replications are taken, whilst the upper limit is enforced as the overall budget is limited.

4.2 Proposing a New Solution

A new solution is found by approximately solving the trust-region sub-problem:

$$\min_{\mathbf{s}} \quad \hat{q}_j(\mathbf{d}_j + \mathbf{s}) \tag{8a}$$

$$\text{subject to} \quad \mathbf{d}_j + \mathbf{s} \in \mathcal{D} \tag{8b}$$

$$||\mathbf{s}||_\infty \leq \Delta_j. \tag{8c}$$

The solution proposed by Conn et al. (1988) is the Generalised Cauchy Step, which minimises $\hat{q}_j$ along the Projected Gradient Path (PGP). This is the projection of the path formed by moving along the negative gradient direction onto the feasible trust-region, $\mathcal{D} \cap \mathcal{B}_j$. As $\mathcal{D}$ and $\mathcal{B}_j$ are both hyper-boxes, the PGP is a series of straight line segments. For the linear model $\hat{q}_j$, the solution lies at a corner of $\mathcal{D} \cap \mathcal{B}_j$. For the quadratic model, we use Algorithm 17.3.1 of Conn et al. (2000). If you restrict $\hat{q}_j$ to a line segment of the PGP, you create a one-dimensional quadratic, so checking it for a local minimum is straightforward. Algorithm 17.3.1 checks each line segment in turn, and returns the smallest local minimum along the PGP.

4.3 Acceptance of the Proposed Solution

Once the algorithm has proposed a new solution, $\mathbf{d}_j^*$, it is evaluated using $N_j^c = \max\{2N_j^p, N_{\min}^c\}$ simulations to estimate $g(\mathbf{x}, \mathbf{d}_j^*)$ with the sample mean $\hat{g}_j(\mathbf{x}, \mathbf{d}_j^*)$. Two tests are then performed. The first is the Ratio Comparison test - a test common to all trust-region approaches:

$$\rho_j := \frac{\hat{g}_j(\mathbf{x}, \mathbf{d}_j) - \hat{g}_j(\mathbf{x}, \mathbf{d}_j^*)}{\hat{q}_j(\mathbf{d}_j) - \hat{q}_j(\mathbf{d}_j^*)} \geq \eta_0 > 0.$$

This compares the predicted reduction in g according to the meta-model with the observed reduction. The test is passed if this ratio is sufficiently large (i.e. greater than η_0). This indicates that the model may give a sufficient fit for prediction purposes. If $\rho_j < \eta_0$, it suggests the meta-model is not an adequate approximation over the trust region.

The second test accounts for the stochastic error in $\hat{g}_j(\mathbf{x}, \mathbf{d}_j^*)$. This is called the Sufficient Reduction test. A Welch's t-test (Welch, 1938) is used to test the following hypotheses:

$$H_{\text{null}} : g(\mathbf{x}, \mathbf{d}_j) - g(\mathbf{x}, \mathbf{d}_j^*) \leq \eta_0^2 \zeta_j \qquad H_{\text{alt}} : g(\mathbf{x}, \mathbf{d}_j) - g(\mathbf{x}, \mathbf{d}_j^*) > \eta_0^2 \zeta_j.$$

This test sees if there is statistical evidence that the proposed solution reduces the objective by a sufficient amount, $\eta_0^2 \zeta_j$. The quantity ζ_j is a measure of the possible reduction in the meta-model by the Generalised Cauchy Step. If the trust region is small and the gradient of the meta-model is shallow, ζ_j will be small, as a large improvement is not possible. On the other hand, if the gradient is steep or Δ_j is large, there is a greater scope for improvement, so ζ_j will be larger. A full definition can be found in Lemma 6.6.2 (for the linear meta-model case) and Lemma 6.6.3 (for the quadratic meta-model case) of Rhodes-Leader (2020, Chapter 6).

If the solution passes the Ratio Comparison test and the null hypothesis of the Sufficient Reduction test is rejected at the α_j significance level, the proposed solution $\mathbf{d}_j^*$ is accepted, and so $\mathbf{d}_{j+1} = \mathbf{d}_j^*$. If either test is not passed, the proposed solution is rejected and $\mathbf{d}_{j+1} = \mathbf{d}_j$.

Note that if a solution proposed by a quadratic model is rejected, STRONG uses an "Inner Loop" procedure. This is slightly difference from the rejection in the linear case, and involves a sharp increase in the sample sizes used. The key aim is to improve the quality of the estimations and enable convergence. This procedure rarely occurred in our experiments. See either Chang et al. (2013) or (Rhodes-Leader, 2020, Chapter 4) for more details.

4.4 Updating the Trust Region

The acceptance of the proposed solution suggests the meta-model is providing a reasonable local fit to the objective function and can be trusted over the trust region. Alternatively, rejecting $\mathbf{d}_j^*$ is an indication that $\hat{q}_j$ is a poor approximation over the current trust-region size. Trust-region algorithms, including STRONG, react to this information by changing the size of the trust region,

Table 4: System parameter values across all experiments (unless specified otherwise).

Parameter	Symbol	Value
Delay cost per minute	c_d	€50
Cost per minute of exceeding planned delay	c_p	€20
Non-passenger cancellation cost		€910
Compensation per passenger: 2 hour departure delay		€10
Compensation per passenger: 3 hour arrival delay		€250
Compensation per passenger: cancellation		€250
Maximum number of allowable tail number exchanges		50
Aircraft separation on take-off runways: 6am-11pm	$s(t)$	66 seconds
Aircraft separation on landing runways: 6am-11pm	$s(t)$	72 seconds
Aircraft separation on both runways: 11pm-6am	$s(t)$	180 seconds
Integer Program turn time (Problem 1)	$t_{\min}$	25 minutes
Integer Program turn time (Problem 2)	$t_{\min}$	20 minutes
Integer Program turn time (Problem 3)	$t_{\min}$	20 minutes

Δ_j. If either the Ratio Comparison test or Sufficient Reduction test is failed, the trust region shrinks, $\Delta_{j+1} = \gamma_0 \Delta_j$ with $\gamma_0 \in (0, 1)$, to improve the linear or quadratic approximation to g. If $\mathbf{d}_j^*$ is accepted and the meta-model is a good fit, indicated by $\rho_j > \eta_1 > \eta_0$, the trust region is expanded, $\Delta_{j+1} = \min\{\gamma_1 \Delta_j, \Delta_{\max}\}$, $\gamma_1 > 1$. This allows larger steps to be taken if possible. On the other hand, if $\rho_j \in (\eta_0, \eta_1)$, the model is providing an adequate fit, so the trust-region size is unchanged.

5 Details of Illustrative Problems

5.1 Problem Parameters

This section gives a brief description of each of the three problems. The full input data, original schedules and input parameters for the IP and simulation models can be found in the supplementary spreadsheets. Some of the parameters that hold across all problems and the turn time parameters for each problem are presented in Table 4. The runway service time parameters were selected after experimentation to ensure that the congestion at the largest airports was not overwhelming. The limiting assumption of all airports having two runways could lead to extremely long queues. Therefore, our values are slightly smaller than those in other parts of the literature.

Within each of the problems, the weather changes were very limited and set randomly, except for the changes at London Gatwick Airport (LGW), which is the cause of the disruption in Problem 3. The parameters are available in the supplementary data sets. In practice, an airline would link these to the very accurate weather forecasts available to guide their operations.

5.1.1 Problem 1

Problem 1 consists of a homogeneous fleet of 8 aircraft with no spare aircraft operating over a hub-and-spoke network of 15 airports in Western Europe. The hubs of the airline are Manchester Airport (MAN) and Birmingham Airport (BHX) in the U.K., where all aircraft must spend the night. At the beginning of the day, it is discovered that one aircraft at BHX will not be fit to fly its first flight. The expected ready time is 3 hours after that, at 8:50, and the repair time follows a

Gamma distribution with shape and scale parameters set to 4 and 50 respectively. The OCC must now reschedule the day of operations, involving 54 short-haul flights using the available aircraft.

5.1.2 Problem 2

Problem 2 considers a fleet of 48 aircraft over 57 European airports. There are several hub airports. An aircraft, A2, lands at 10am at Amsterdam Airport Schiphol (AMS) from its hub Luton Airport (LTN), and a fault is discovered. The repair time follows a Gamma distribution with shape and scale parameters set to 3 and 111 respectively. As AMS is another hub of the airline, there are options to exchange the aircraft assigned to future flights, but A2 is due back at LTN for overnight maintenance. There are 198 flights left in the operating day, which the OCC must now reschedule.

5.1.3 Problem 3

Problem 3 considers a fleet of 102 aircraft over 77 European airports. Bad weather at London Gatwick Airport (LGW) forces the airport to reduce its runway capacities by 80% between 9am and 11am, leading to high congestion levels. This is a hub airport for the fleet, and the reduced capacity will affect 11 aircraft whose flights arrive at or depart from LGW during this period, which could have consequences for subsequent flights. Whilst there is little that can be done about the flights directly delayed by the weather, other flights could be rearranged to reduce the impact of the disruption. There are 440 flights left in the day.

5.2 Simulation and Simulation Optimisation Algorithm Parameters

The parameters used for the adapted STRONG algorithm in all experiments are shown in Table 5. These values are either influenced by characteristics of the airline system or the values used in the evaluation of STRONG in Chang et al. (2013). The choice of $N^c_{\min}$ and $N^p_{\min}$ were chosen as the variability of the simulation output was fairly high. After some experimentation, $N^p_{\max}$ was chosen as a balance between the budget used to build the meta-model and good predictions from the meta-model. Results in Chapter 5 of Rhodes-Leader (2020) suggest there was little difference between $N^p_{\max} = 50$ and $N^p_{\max} = 25$.

For convergence, STRONG requires that $\sum_{j=0}^{\infty} \alpha_j < \infty$. We used $\alpha_j = 0.25 \times 0.98^j$.

The initial trust-region radius, Δ_0, was chosen to reflect that 15 minutes is often the slot size for runway scheduling. The values of $\tilde{\Delta}$ and $\Delta_{\max}$ were chosen in relation to this.

References

Airbus (2019). A319 Aircraft Characteristics - Airport and Maintenance Planning. Technical report, Airbus.

Allamraju, H. (2017). pyflightdata. Accessed March 11th, 2017. `https://github.com/supercoderz/pyflightdata`.

Chang, K.-H., Hong, L. J., and Wan, H. (2013). Stochastic trust-region response-surface method (STRONG) - a new response-surface framework for simulation optimization. *INFORMS Journal on Computing*, 25(2):230–243.

Civil Aviation Authority (2015). Your Rights When You Fly. Accessed March 20th 2020. `https://www.caa.co.uk/Passengers/Resolving-travel-problems/Delays-cancellations/Your-rights/Your-rights-when-you-fly/`.

Table 5: Parameter values for the simulation optimisation across all experiments.

Parameter	Symbol	Value
Maximum observations in simulation optimisation	$N_{\max}$	2000
Minimum replications at each design point	$N^p_{\min}$	5
Maximum replications at each design point	$N^p_{\max}$	50
Minimum replications for acceptance tests	$N^c_{\min}$	20
Replications for comparison with zero delay option	$N^c_{\mathbf{0}}$	50
Initial trust-region radius (minutes)	Δ_0	15
Threshold trust-region radius (minutes)	$\tilde{\Delta}$	10
Maximum trust-region radius (minutes)	$\Delta_{\max}$	25
Trust region shrinkage factor	γ_0	0.9
Trust region expansion factor	γ_1	1.11
Acceptance fraction for RC test	η_0	0.01
Acceptance fraction for expanding trust region	η_1	0.3
Significance level of first SR test	α_0	0.25
Percentage increase stopping criterion for CEA	ε_D	1%

Conn, A. R., Gould, N. I. M., and Toint, P. L. (1988). Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(2):433–460.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (2000). *Trust Region Methods.* Society for Industrial and Applied Mathematics, Philadelphia.

Flightradar24 AB (2017). Flightradar24. Accessed March 7th, 2017. `https://www.flightradar24.com`.

ICAO (2016). Doc. 4444 Procedures for Air Navigation and Air Traffic Managment (PANS-ATM). Technical Report November, 16th edition, International Civil Aviation Organitzation.

Laumanns, M., Thiele, L., and Zitzler, E. (2006). An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942.

Lee, L. H., Huang, H. C., Lee, C., Chew, E. P., Wikrom, J., Yong, Y. Y., Liang, Z., Leong, C. H., Tan, Y. P., Namburi, K., Johnson, E., and Banks, J. (2003). Discrete Event Simulation Model for Airline Operations: SIMAIR. In S. Chick et al., editor, *Proceedings of the 2003 Winter Simulation Conference*, pages 1656–1662, Piscataway, New Jersey. IEEE.

Meyer, R. K. and Nachtsheim, C. J. (1995). The coordinate-exchange algorithm for constructing exact optimal experimental designs. *Technometrics*, 37(1):60–69.

Rhodes-Leader, L. A. (2020). *Multi-fidelity modelling approach for airline disruption management using simulation.* PhD thesis, Lancaster University.

The AnyLogic Company (2017). Anylogic 8.2.3. Accessed July 23rd, 2018. `http://www.anylogic.com`.

Welch, B. L. (1938). The significance of the difference between two means when the population variances are unequal. *Biometrika*, 29(3):350–362.