Multi-fidelity Modelling Approach for Airline Disruption Management using Simulation

Luke Rhodes-Leader, M.Sci.(Hons.), M.Res

Submitted for the degree of Doctor of Philosophy at

Lancaster University.

June 2020

Abstract

Disruption to airline schedules is a key issue for the industry. There are various causes for disruption, ranging from weather events through to technical problems grounding aircraft. Delays can quickly propagate through a schedule, leading to high financial and reputational costs. Mitigating the impact of a disruption by adjusting the schedule is a high priority for the airlines. The problem involves rearranging aircraft, crew and passengers, often with large fleets and many uncertain elements. The multiple objectives, cost, delay and minimising schedule alterations, create a trade-off. In addition, the new schedule should be achievable without over-promising. This thesis considers the rescheduling of aircraft, the Aircraft Recovery Problem.

The Aircraft Recovery Problem is well studied, though the literature mostly focusses on deterministic approaches, capable of modelling the complexity of the industry but with limited ability to capture the inherent uncertainty. Simulation offers a natural modelling framework, handling both the complexity and variability. However, the combinatorial aircraft allocation constraints are difficult for many simulation optimisation approaches, suggesting that a more tailored approach is required. This thesis proposes a two-stage multi-fidelity modelling approach, combining a low-fidelity

Integer Program and a simulation. The deterministic Integer Program allocates aircraft to flights and gives an initial estimate of the delay of each flight. By solving in a multi-objective manner, it can quickly produce a set of promising solutions representing different trade-offs between disruption costs, total delay and the number of schedule alterations. The simulation is used to evaluate the candidate solutions and look for further local improvement. The aircraft allocation is fixed whilst a local search is performed over the flight delays, a continuous valued problem, aiming reduce costs. This is done by developing an adapted version of STRONG, a stochastic trust-region approach. The extension incorporates experimental design principles and projected gradient steps into STRONG to enable it to handle bound constraints. This method is demonstrated and evaluated with computational experiments on a set of disruptions with different fleet sizes and different numbers of disrupted aircraft. The results suggest that this multi-fidelity combination can produce good solutions to the Aircraft Recovery Problem.

A more theoretical treatment of the extended trust-region simulation optimisation is also presented. The conditions under which a guarantee of the algorithm's asymptotic performance may be possible and a framework for proving these guarantees is presented. Some of the work towards this is discussed and we highlight where further work is required.

This multi-fidelity approach could be used to implement a simulation-based decision support system for real-time disruption handling. The use of simulation for operational decisions raises the issue of how to evaluate a simulation-based tool and its predictions. It is argued that this is not a straightforward question of the real-world result being good or bad, as natural system variability can mask the results. This problem is formalised and a method is proposed for detecting systematic errors that could lead to poor decision making. The method is based on the Probability Integral Transformation using the simulation Empirical Cumulative Distribution Function and goodness of fit hypothesis tests for uniformity. This method is tested by applying it to the airline disruption problem previously discussed. Another simulation acts as a proxy real world, which deviates from the simulation in the runway service times. The results suggest that the method has high power when the deviations have a high impact on the performance measure of interest (more than 20%), but low power when the impact is less than 5%.

Acknowledgements

I would like to thank each of my academic supervisors, Dave Worthington, Stephan Onggo and Barry Nelson, for offering support, advice and reassurance throughout this project. I normally left our meetings feeling better than I went in, and have learnt a lot about research from each of you. I hope you didn't mind me eating so much of your time. Thanks to Stephan for your continued support and technical advice despite moving away from Lancaster. To Barry, thank you for your involvement, especially as it ended up being more than initially planned, and for sharing your knowledge and experience. To Dave, thank you for being my go to person for advice, for the discussions about general academic life, and for being happy to talk through any topic I wasn't sure about.

Much of our practical knowledge of the problem came via Richard Standing at Rolls-Royce. I am grateful for the time you put into this project, and for organising meetings with your various colleagues to help us understand the important realities of the problem.

The EPSRC funded STOR-i CDT has played an enormous role in writing this. Thank you to Professors Jonathan Tawn, Idris Eckley and Kevin Glazebrook for successfully creating a friendly, supportive and helpful environment to work in. The work has benefited from many conversations within STOR-i, but particular thanks goes to Lucy Morgan, Sam Tickle, Alex Fisch, Jamie Fairbrother, David Torres Sanchez, Aaron Lowther and Harjit Hullait.

This work would not have been possible without the financial support of the EPSRC funded EP/L015692/1 STOR-i Centre for Doctoral Training and Rolls-Royce Holdings plc. This research was partially supported by the National Science Foundation of the United States Grant Number DMS-1854562.

Finally, my family and friends for all their support throughout my time at university, especially my wife Rachel. Thank you for your love and support, for putting up with some long working hours and, most importantly, for reminding me of life outside the PhD. It would have been easy to forget that without you.

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

Early versions of the work presented in Chapters 3, 4 and 5 are presented in the following conference papers:

Rhodes-Leader, L., Onggo, B. S., Worthington, D. J., and Nelson, B. L. (2018). Airline Disruption Recovery using Symbiotic Simulation and Multi-fidelity Modelling. In A. Anagnostou et al., editor, *Proceedings of the Operational Research Society Simulation Workshop 2018 (SW18)*, pages 146–155, Birmingham, UK. The Operational Research Society.

Rhodes-Leader, L., Onggo, B. S., Worthington, D. J., and Nelson, B. L. (2018). Multi-fidelity Simulation Optimisation for Airline Disruption Management. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 2179–2190, Piscataway, New Jersey. IEEE.

The word count for this thesis is 50,890 words.

Luke Austin Rhodes-Leader

June 2020

Contents

List of Figures

List of Tables

List of Abbreviations

ARP	Aircraft Recovery Problem
CDF	Cumulative Distribution Function
CEA	Coordinate-Exchange Algorithm
CRN	Common Random Numbers
ECDF	Empirical Cumulative Distribution Function
ICAO	International Civil Aviation Organization
IATA	International Air Transport Association
i.i.d	Independent and Identically Distributed
IP	Integer Program
OCC	Operations Control Centre
OLS	Ordinary Least Squares
RC	Ratio Comparison
SR	Sufficient Reduction
STRONG	Stochastic Trust-Region Response Surface Method
w.p.	With Probability

List of Symbols

$\mathbb{E}[\cdot]$	Expectation.
$\Pr(\cdot)$	Probability.
$\mathbb{I}(\cdot)$	Indicator function.
$\chi(\mathbf{d}, \theta)$	A first-order criticality measure at solution $\mathbf{d}$. See Definition 6.2.3.
$\chi(\mathbf{d})$	Defined as $\chi(\mathbf{d}, 1)$.
$\pi(\mathbf{d})$	A first-order criticality measure at solution $\mathbf{d}$. See Definition 6.2.4.
$g(\cdot)$	Objective function.
∇	Gradient operator.
$\mathcal{P}_{\mathcal{X}}[\mathbf{d}]$	The orthogonal projection of vector $\mathbf{d} \in \mathbb{R}^n$ onto the set $\mathcal{X} \subset \mathbb{R}^n$.
$p(\cdot, \mathbf{d})$	Projected-gradient path from solution $\mathbf{d}$. See Equation (4.3.9).
ρ_j	Ratio Comparison test value. See Equation (4.3.12).

Chapter 1

Introduction

Disruption to schedules is one of the major issues faced by the airline industry. According to the Civil Aviation Authority (2019) punctuality statistics for 2018, of the flights that operated through Heathrow Airport, only 72.6% arrived or departed within fifteen minutes of their scheduled time, while this figure was only 69.1% for Gatwick Airport. The statistic for other major U.K. airports was less than 85%. This does not include all the flights that were cancelled. Despite the advances in schedule planning, allowing ever more efficient use of the resources available to airlines, it is rare that a flight programme is operated as intended. This is due to the complex and uncertain nature of the industry. Disruptive events have various causes, from weather conditions to aircraft technical failures requiring unplanned maintenance. 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 are operating at or near full capacity. In an investigation into the costs of disruption, Cook and Tanner (2015) found that every minute of delay generates an additional 0.8 minutes of delay for other flights reacting to the initial disruption.

When a disruption occurs, the Operations Control Centre (OCC) of the airline must alter its schedule to reduce the repercussions for its finances, reputation and passengers. Kohl et al. (2007) describe the practice of the OCC under these disrupted operations. The OCC wants to identify good actions and evaluate them from the perspective of the airline and the passengers. The disrupted flight is not an isolated system, so the consequences of an action could have wider implications on the schedule. These effects must be examined when considering an action. Large (and highly utilised) fleets operating over networks involving many airports and high levels of uncertainty make this a difficult problem.

The complexity of the industry can make it difficult to determine what the consequences of any schedule alterations may be. This is where operational research techniques can be of use, 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. However, these models have a limited capability of accounting for the various uncertain elements of the environment. Not accounting for uncertainty can affect the viability of a solution. An alternative is stochastic simulation modelling. In a simulation model, one can easily deal with both complexity and uncertainty. However, this does not provide a natural strategy for searching through the possible revised schedules. This thesis aims to explore the use of simulation for finding and evaluating solutions for airline disruption management.

1.1 Application Setting

There is a long history of using operational research methods within the aviation industry, with applications across many aspects of the business. These include customer modelling, revenue management, crew rostering and Air Traffic Control (ATC), all of which are mentioned by Barnhart and Smith (2012). Another prominent area is the production of the schedule. Clausen et al. (2010) discuss the planning process. Between a year and 6 months in advance, the preliminary flight schedules are produced, based on passenger demand and availability of aircraft movement slots at airports. Following this, a type of aircraft is allocated to individual flights and sequences of flights. This is known as fleet routing, and includes maintenance planning. Around 4 to 6 months ahead, crew are assigned based on the fleet type. The final stage is the assignment of specific aircraft to flights, which is known as the tail-assignment problem. This stage happens between five days (for long-haul flights) and one day (for short-haul flights) before the flight itself. This process typically involves several optimisation models. The aim is often to maximise revenue and resource utilisation.

However, there are many sources of uncertainty in the industry such as weather conditions, technical problems with aircraft and crew illness. Bratu and Barnhart (2006) classify the disruptions into two classes: shortages in airline resources (such as aircraft requiring unscheduled maintenance, crew absences and lack of ground resources); and shortages in airport resources (such as reductions in aircraft movements on runways due to poor weather conditions). The high utilisation of aircraft and airports can result in schedules that are sensitive to perturbations and disruption.

There are some ways of building slack into the schedule, such as leaving longer turn times and scheduling a higher block time (time between leaving the departure gate and arriving at the arrival gate) than is required for each flight. Some work has been done to improve the robustness of schedules (Lee et al., 2007). However, this buffer is expensive. The trade-off between cost and robustness makes it unattractive for airlines to create expensive schedules that minimise the chance of the slack being exceeded. When disruptions occur, operational decisions must be taken by the OCC to rearrange the schedule. The response can be critical in minimising the impact to passengers and the airline itself.

Rescheduling is a complex problem. There are many factors that must be considered. The state of the system involves: the position, disruption status and future maintenance requirements of all aircraft; the position, disruption status, duties and remaining legal flying time of all crew; and the position, disruption status, itinerary and planned arrival time of all passengers (Bratu and Barnhart, 2006). There are a range of key performance measures, such as resuming normal operations, disruption costs, total delay and schedule alterations. With a large fleet, this can be a lot of information for which to account. To reduce the problem size, the rescheduling is often split into three parts: schedule and aircraft recovery; crew recovery and passenger recovery (Kohl et al., 2007). The flight and aircraft recovery problem is often solved first, before considering this schedule from the crew and passenger perspectives. If the crew or passenger controllers' feedback indicates that this is infeasible, the flight and aircraft recovery is modified further. The infeasibility could arise from legal restrictions on crew, such as limitations on aircraft that can be flown and restricted duty hours. The iteration continues until the solution is feasible (Wang et al., 2019).

As this is the current framework for disruption management, this thesis will focus on the flight and aircraft recovery, also known as the Aircraft Recovery Problem (ARP). There are a variety of alterations that could be made including delaying or cancelling flights, exchanging aircraft, or ferrying aircraft (flying an empty aircraft to where it is needed). The latter is often prohibitively expensive and is not considered here.

Wang et al. (2019) found that in practice, these decisions are made based on controller experience and a set of rules from the airline. Where it is possible to get the flight operational, either by replacing the aircraft or by quickly carrying out the necessary repairs, this is normally the preferred option for airlines as it helps to prevent immediate delays. This can lead to short-term decisions, prioritising the immediate without necessarily accounting for the longer-term consequences of the change. Kohl et al. (2007) state that controllers are often seeking something viable rather than optimal, as they do not have time to consider lots of options.

This is where operational research could be highly valuable for disruption management. In a recent white paper discussing the use of data science within the airline industry, the International Air Transport Association (IATA) highlighted the potential of technology to aid decisions in recovery from irregular and disrupted operations (Section 2.4 of Goudarzi et al. (2019)). Kohl et al. (2007) suggest that automation could play a large part in the identification and evaluation of potential recovery actions. The ARP is a well-studied problem, with many publications proposing solution methods. Most of these take the form of Integer Programs (IPs), deterministic models extended from the models built for the original scheduling problem (Teodorović and Guberinić, 1984; Thengvall et al., 2000).

However, these models only have a very limited capability of accounting for the various uncertain elements of the environment. Turn times, airport queueing times, maintenance times and flight durations are all uncertain and excluding these could lead to solutions performing much worse than predicted. Thus simulation seems to be a natural way to model the airline's operations.

1.2 Simulation for Real-time Decision Making

The use of simulation for operational decision making has become more popular over the last twenty years. This has become possible through advances in computational power and data connectivity. The paradigm that exploits these advances more than others is symbiotic simulation (Fujimoto et al., 2002). Symbiotic simulation allows the simulation model to be adapted to new scenarios through a close interaction with the physical system. In the most basic form, this means matching the initial conditions of the physical system, but model parameters can also be adapted (Aydt et al., 2009; Onggo et al., 2018).

One of the difficulties of online simulation for decision making is that high-fidelity simulation models have a non-negligible computation time. This is problematic when searching through a large solution space. To make the most of the simulation, it must be used selectively on solutions believed to be worth evaluating. Perhaps a more important challenge is that allocating aircraft to flights has combinatorial constraints.

This is a difficult problem for many simulation optimisation methods. This thesis proposes a methodology aiming to overcome these challenges. It is a two-stage approach, using a deterministic model to handle the combinatorial constraints and point the simulation optimisation to areas that seem promising. This is called a multi-fidelity modelling approach.

The ARP is a problem that must be faced in real-time. Thus, the system we propose would be a simulation-based operational Decision Support System (DSS). As with any tool, a question that arises when considering a simulation-based DSS is how to evaluate the tool. The real world will only happen once, so finding out if that one realisation comes from the predicted distribution from the simulation is not possible. This means that checking to see if the simulation is giving a good enough prediction is difficult. If there is a systematic flaw, this could be very costly as the simulation is used repeatedly on a sequence of disruptions. The final contribution of this thesis is to discuss this problem and propose a simple method for combining a series of decisions to give a statistically meaningful detection of systematic errors.

1.3 Thesis Purpose and Overview

The primary purpose of this thesis is to consider how stochastic simulation could be used, in conjunction with other methodologies, to aid an airline in their response to a disruptive incident, particularly in rearranging the aircraft flight allocation and retiming or cancelling flights. It investigates how to generate, search through and select from a set of possible rescheduling options using information from a simulation model that accounts for the various sources of uncertainty within the operations. Furthermore, how such a decision support tool could be evaluated is considered.

The remainder of this section provides an overview of the thesis structure.

In Chapter 2, we first review the literature on airline disruption management, looking at both the deterministic models and the limited use of simulation to handle uncertainty in the ARP. This is followed by a review of the simulation optimisation literature, an area we wish to make use of to help improve solutions. As the method proposed in this thesis is a multi-fidelity method, combining both the deterministic approaches with simulation optimisation, we also review the relevant multi-fidelity modelling literature, looking at how simpler models have been used to guide the optimisation of a more complex model. A brief overview of the real-time simulation literature is then provided. Finally, the primary research question is stated and we provide an overview of the multi-fidelity modelling approach proposed in this thesis.

In Chapter 3, the deterministic low-fidelity model used in this thesis is described. A problem statement for the ARP is given, and the time-space network model, formulation and solution procedure are presented. Further theoretical considerations are also discussed.

The simulation optimisation method is developed in Chapter 4. An overview of the simulation model built for this research is also given (a more thorough description is given in Appendix A). This is followed by the main details of the simulation optimisation algorithm developed in this research. It is based on STRONG (Chang et al., 2013), a trust-region based simulation optimisation method. Extensions of this algorithm, developed to handle the bound constraints on the feasible region, are explained. Some considerations that would improve the practical performance of the algorithm are also discussed.

To evaluate the method, some empirical experiments based on a set of example problems reflecting different real-world disruptions are performed. The results of these are presented in Chapter 5. This chapter also demonstrates how the method could be used in practice.

Further details of the extensions to STRONG are given in Chapter 6. The results in Chapter 5 are dependent on the simulation model used. There are other factors not considered in Chapter 5 that one may wish to account for in the model, e.g., missed passenger connections, that would change the objective function. To show that the simulation optimisation has potential when other models are used, and for more general bound constrained problems, this chapter considers the theoretical properties of the algorithm. In particular, we present a pathway for proving its asymptotic convergence.

Chapter 7 argues that evaluating a simulation-based DSS for operational decisions is a difficult problem due to natural variability and each problem being unique. We propose a method to tackle a part of this problem, trying to detect whether the simulation predictions are systematically erroneous. We demonstrate the approach on test scenarios for the ARP in this thesis.

Chapter 8 summarises the contributions of this thesis and identifies possible research directions to develop further the research presented here.

Chapter 2

Literature Review

This chapter reviews the academic literature relevant to this thesis. Section 2.1 considers the literature concerning airline disruption management. The primary focus is on the Aircraft Recovery Problem (ARP), but we also draw upon works integrating the aircraft, crew and passenger recovery problems. Many of these models are deterministic, ignoring the uncertainties that are present in airline operations. There are also applications of simulation in this area.

We argue that simulation is a more relevant modelling approach for the ARP than the deterministic models previously proposed, thus making simulation optimisation a critical area for this research to review. Section 2.2 gives a summary of some of the main methodologies. We suggest that these methods are not sufficient for the combinatorial aspects of the ARP, and instead consider a multi-fidelity modelling approach, combining both the deterministic models and simulation optimisation. Multi-fidelity modelling applications and methods of combining them are reviewed in Section 2.3. As the ARP is an operational-level decision, as opposed to the strategic and tactical-level decisions for which simulation is typically applied, we discuss some of the ways in which simulation has recently been proposed for real-time decision making in Section 2.4. In particular, we discuss symbiotic simulation and validation considerations for real-time simulation. This is predominantly driven by applications. In Section 2.5, we conclude and present the principle behind the ARP solution method proposed in this thesis.

2.1 Airline Disruption Management

The airline disruption problem has received much attention in the academic literature over the last 35 years, starting with Teodorović and Guberinić (1984). The practical issues are reviewed by Kohl et al. (2007), who argue that the use of automation to identify and evaluate possible recovery options could enhance the experience and rule-based approach of current practice.

Much of the recent literature has researched the integration of different aspects of the recovery problem. This is an important area of research, particularly as constraints on one resource can make an optimal solution for another infeasible in the full picture. For example, the work of Jimenez Serrano and Kazda (2017) presses the importance of including passengers within the disruption problem. Despite the solution quality benefits of integrating the solutions, this also makes the problem much larger so it takes a lot longer to solve.

By working on projects directly with airlines, Kohl et al. (2007) and Wang et al. (2019) were able to describe the current practice in industry, which is primarily to separate the problem into aircraft, crew and passengers. The Operations Control Centre (OCC) often propose a new aircraft allocation and flight plan first, before checking if this meets the legal obligations for crew and passengers. If there are violations, a new aircraft allocation is found. Thus, this thesis focusses on the aircraft problem.

This section reviews the academic work in this area. Whilst the focus is on the ARP, research on integrated methods is included as this gives the wider context of the problem.

2.1.1 Deterministic Airline Recovery Problem

There have been many and varied deterministic models and solution methods proposed in the literature for the ARP. There are also numerous examples within the literature of integrating multiple aspects of the disruption problem, which are included as they are instructive to the state of the area. Most of these take the form of an Integer Program (IP). Some of the methods involve a single fleet (Thengvall et al., 2000), whilst others allow multiple fleets (Zhang et al., 2015). The operational constraints vary, with most papers including some combination of turn time constraints, airport capacity constraints (Rosenberger et al., 2003), aircraft balance (Bratu and Barnhart, 2006) and maintenance constraints (Liang et al., 2018). Clausen et al. (2010) provide a thorough review of the primary formulation types. These are time-space network models (also known as time-line networks), time-band network models and connection network models. There are a wide range of solution approaches, some exact and some heuristic. This section briefly describes each of the models and gives some examples of their use.

Time-space networks model a schedule as a network with nodes representing airports at a particular time and arcs representing flights between airports (flight arcs) and time spent on the ground (ground arcs). Each aircraft also has an input node, representing its current position and return nodes are possible positions at the end of the time horizon. To handle disruption and delayed flights, copies of each flight arc are added to the network. To obtain a new schedule, each aircraft in the fleet requires a continuous path through the network from the disruption time to the end of the time horizon considered. The arcs forming the path define to which flights the aircraft is allocated. An example can be found in Thengvall et al. (2000), who consider the ARP, using a time-space network with additional, lower cost flight arcs to encourage flights to be covered by the original aircraft. The model included delays, aircraft exchanges and cancellations. As IPs can be difficult to solve, the authors relax the integer constraints and solve a Linear Program (LP) followed by a heuristic policy to handle fractional variables.

The time-space network can also be used when integrating the aircraft, crew and passenger problems. Bratu and Barnhart (2006) propose a passenger-centric time-space network. This combines operational and passenger costs when deciding how to accommodate disrupted passengers, allowing flights to be delayed or cancelled and includes reserve crew decisions. The IP is solved exactly. Sinclair et al. (2014) use a time-space network during the repair phase of a Large Neighbourhood Search algorithm for an aircraft-passenger integrated problem. It allocates aircraft to flights to minimise passenger and operational costs. Time-space networks are used by Zhang et al. (2015) for integrating the aircraft and crew problems. The algorithm iterates between two models, one for each resource, until a satisfactory solution is found. The aircraft model is a time-space network multi-commodity flow formulation, based on fleet assignment, with a constraint enforcing a minimum connection time between flights with the same crew. Once the fleet is assigned, individual tail numbers are allocated using a greedy randomised search heuristic. The crew model (also based on a time-space network) has a similar constraint ensuring aircraft minimum turn times are respected. For a more detailed description of time-space networks, see Section 3.3.1.

Unlike the time-space network, the time-band network was developed specifically for disruption management. The recovery period is split into bands of equal time, e.g., 30 minutes. A node at an airport within a time band contains all of the activities (i.e., arrivals and departures) aggregated over the time band. The time label of the node is the earliest an aircraft becomes available within that time band. There are two types of arcs, one representing flights (from one airport to another) and the other connecting a node at an airport to the corresponding sink node at the end of the recovery window. The time-band structure allows these networks to be constructed dynamically, one band at a time. This approach means that only departure times at which an aircraft is actually available to be assigned to a flight are represented as nodes, unlike the time-space networks. This reduces the problem size, but at the expense of accuracy.

One of the early examples of a time-band network for the ARP was by Bard et al. (2001). An IP formulation is solved exactly to show aircraft flows through the network before aircraft are assigned. If this is not feasible, the time band is reduced and the IP is solved again. Quansheng et al. (2013) use a time-band network to generate potential aircraft routes that are used within a column generation solution method. Hu et al. (2015) develop a time-band network for modelling the integration of aircraft and passengers who are only transferred if their flight is cancelled. The nodes are placed at the beginning of the time band, and so the model tends to underestimate delays.

An alternative view is to place the activities (i.e., flights) on the nodes themselves. The arcs connecting two nodes indicate that these two flights can be flown in sequence by the same aircraft, removing the need for explicit turn time constraints. This type of model is called a connection network. A path through the network is a sequence of flights for an aircraft, known as a rotation. These networks grow exponentially in size with the number of flights and so it is impractical to assign aircraft to arcs, especially when delays become involved. Thus, rather than assigning aircraft to individual flights, methods generally aim to assign aircraft to rotations. As the number of feasible rotations, defined by the paths through the connection network, is very large, a challenge faced by these formulations is in generating all of the variables before the optimisation begins. The costs and delays of a route can be calculated during construction, as one can determine the necessary delay on each route.

This approach is used by Rosenberger et al. (2003) with a set-packing formulation, minimising costs and including runway slot constraints and airport capacity limits. To generate the rotations, the algorithm creates new rotations from the original by removing subsequences of flights, and performing a single swap with rotations of the other aircraft in the fleet. This is combined with a problem reduction heuristic to remove aircraft from the problem if they are undisrupted or cannot be involved in recovery. Whilst this approach does not cover all feasible rotations, it does produce many good ones. Thus, the problems can be solved fairly quickly using an IP solver. Wu et al. (2017) use a similar approach in trying to generate the rotations a priori. The authors propose two models, one to minimise cost and the other to maximise profit.

Generating the rotations can be very difficult, thus, rather than generating the rotations a priori, one could do it during the optimisation. Column generation is an optimisation technique that starts with a subset of the decision variables and iteratively adds new variables, in this case rotations, until the solution is optimal. The new variables are selected based on a sub-problem. In their rotation-based ARP formulation, Liang et al. (2018) include maintenance periods as nodes in the connection network. Variables are added based on finding rotations with a negative reduced cost in the master problem. In the sub-problem, the delay of each flight is a variable to be optimised, whilst incorporating airport capacity constraints. To improve speed, only the LP relaxation of the master problem is solved until all required rotations have been added. The authors also note that, as maintenance schedules include some slack, it is possible to swap maintenance periods as long as one adheres to the legal limits. Thus, the original maintenance plan is not treated as a hard constraint. Sarac et al. (2006) face a slightly different problem where one has to re-allocate flights to aircraft to return particular aircraft to maintenance stations after a disruption has changed their schedule. New rotations are selected by solving a constrained shortest path problem.

The aircraft and crew are integrated into one rotation-based model by Maher (2016). The author solved this using a column and row generation procedure. Similarly, Petersen et al. (2012) consider the whole integrated problem using a rotation-based model, but here the 'schedule' is recovered first by assigning fleets to rotations before individual aircraft, crew and passenger itineraries in a Bender's decomposition solution method. New rotations are added based on reduced cost. To reduce the problem size, the algorithm excludes 'non-disruptable' flights from the problem, and only includes delay possibilities where the problem structure changes.

Løve et al. (2005) proposed a different network model. Aircraft and flights are represented as nodes. Arcs connecting aircraft nodes to flight nodes indicate the aircraft allocation. A Steepest Ascent Local Search heuristic is applied with a profit objective accounting for revenue, delay and cancellation costs. The neighbourhood is defined by exchanging aircraft allocations. This was part of the DESCARTES project (Kohl et al., 2007), where separate solvers for each resource were developed.

Whilst the ARP is generally focussed on larger disruptions, smaller delays can cause as much of a problem as passengers can miss connections. This results in the decision of whether to wait for connecting passengers or not. Santos et al. (2017) develop an IP operating on a rolling time horizon to make these decisions, accounting for detailed airport capacity constraints (runways, taxiways and stands) and the impact on other passengers. Flights are only retimed, with no aircraft exchanges.

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). The multi-objective nature of the ARP was noted by Thengvall et al. (2000), particularly in the trade-off between maximum revenue and minimal schedule alterations. However, it is treated as a single objective problem with incentives for selecting 'protection arcs' representing a sequence of flights originally put together before the disruption. Liang et al. (2018) also note that minimising delay, cancellations, aircraft exchanges and maintenance exchanges improve the solution, but assume that an exact cost can be determined for each and include it within the cost objective. Hu et al. (2017) propose a full multi-objective formulation, trying to identify Pareto optimal solutions, with objectives considering the disruption cost, number of aircraft rerouted and the maximum delay. Their solution method used a local neighbourhood search heuristic combined with an ϵ-constraint method. A multi-objective approach is taken by both Liu et al. (2010) and Jeng (2012), applying a multi-objective genetic algorithm to find the Pareto frontier. Jeng (2012) includes the objectives of cost, number of delays and schedule alterations, whilst Liu et al. (2010) also adds variability of delays and number of long delays. Solution comparisons are based on whether one solution dominates the other across the objectives. Operational constraints are included in the fitness function.

The inherent uncertainty in the airline industry could mean that the optimal solution may change as conditions become clear later on, as concluded by Rosenberger et al. (2003), who mention unpredictable changes in weather conditions. Zhu et al. (2015) begin to incorporate this uncertainty by considering that the duration of unplanned maintenance is unknown. They propose a two-stage recourse stochastic program, the first stage decision is allocating aircraft to rotations, the second is flight retiming. The scenarios are times at which the aircraft requiring unplanned maintenance becomes available. This is solved using a greedy simulated annealing algorithm, with local search operations to generate new solutions. The weakness of such an approach is that if it were to be extended to include scenarios with other uncertain elements, the size of the problem would explode and become computationally intractable within the time constraints.

2.1.2 Simulation of Airline Operations

Whilst simulation has a long history of use within the airline industry (Gunn, 1964), considerations of disruption have been relatively recent. The initial use was for evaluating the robustness of a long term schedule to disruptions at the strategic planning stage. Rosenberger et al. (2000) developed a stochastic discrete event simulation model of airline operations, including aircraft and crew, known as SimAir. This simulates the schedule for a long period (weeks or months). Simple heuristics were included to create recovery options after a disruptive event. Both the model and the recovery options were extended by Rosenberger et al. (2002). Lee et al. (2003) made the software modular to enable other recovery modules to be used. The purpose of SimAir was twofold, to provide a method to evaluate recovery policies on a long term schedule and to monitor the robustness of that schedule. Similar simulations were proposed by Yan et al. (2005) and Lapp et al. (2008), the latter specifically for minor delays. Lee et al. (2007) went further, using SimAir to improve the robustness by retiming flights with a multi-objective genetic algorithm.

However, there is scope to use simulation at the operational level, rather than the tactical level. Kohl et al. (2007) say that a simulation to perform 'what if' analysis could be valuable to the OCC.

In the U.S.A., high levels of congestion in the airspace around an airport due to reduced capacity is dealt with by imposing departure delays on aircraft due to arrive at that airport. This is called a Ground Delay Program, and is imposed by the relevant Air Traffic Control (ATC) authority. To decide how much to delay each flight, Hutchison and Hill (2001) use a Simultaneous Perturbation Stochastic Approximation in a simulation optimisation process to reduce air delays across a network. The research takes a network perspective rather than that of an individual airline. The ground delay imposed by ATC could be considered part of a disruption to an airline, from which it must recover.

There are also applications of simulation within the airline disruption problem itself. Abdelghany et al. (2008) propose a deterministic network simulation to predict which flights will be disrupted by considering the consequences for both aircraft and crew. Only the identified flights are included in an IP model for integrating crew and aircraft, reducing the problem size. This is solved on a rolling horizon during the recovery window.

Stochastic turn times and flight durations within the ARP were considered by Arias et al. (2013) who used a constraint programming approach combined with a Monte Carlo simulation to minimise delay. The constraint programming approach considers a deterministic problem, where mean values are used in the place of uncertain quantities, and performs a local search. The simulation is used to evaluate the constraint program solution under a variety of scenarios by adding normal noise to the turn times and flight durations. If the solution is not considered robust across these replications, it is rejected and the process begins again. 'Robustness' could mean that the gap between the deterministic and simulated delay is sufficiently low, or that the variance of the output is proportional to the variance of the noise distributions added to the turn times and flight durations. Guimarans et al. (2015) expanded this approach in two ways. The first was combining the constraint program with a Large Neighbourhood Search heuristic to propose new solutions. The second was in using the simulation at each proposed solution to evaluate the acceptance criteria (known as *SimLNS*). 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.

Most recently, Wang et al. (2019) developed a discrete event dynamic system simulation model for the ARP. They argue that the mathematical programming approach makes it difficult to formulate airline specific constraints, whereas simulation can accommodate this fairly naturally. Historic and real-time data is fed into the simulation. Rather than using an optimisation technique, the authors use a set of rule-based methods to decide which flights to delay or cancel in two disruptive scenarios. Reallocation of aircraft is not allowed. The simulation is used to see which of the proposed solutions appear better.

In summary, there have been many methods proposed for tackling airline disruption, mostly based on deterministic models. Relatively little has been done to account for the uncertainty in the operational problem, though simulation has been proposed to help evaluate potential rescheduling options. This research aims to extend the use of simulation in the ARP further.

2.2 Simulation Optimisation

Simulation optimisation is the process of altering the (physically) controllable inputs for a simulation model to minimise (or maximise) a performance measure of the system represented by the simulation. Jian and Henderson (2015) formulate the general problem as

$$\min_{\mathbf{x}\in\mathcal{X}} g(\mathbf{x}) \tag{2.2.1}$$

where $\mathbf{x}$ is a decision, $\mathcal{X}$ is the set of possible decisions, $g\colon \mathcal{X} \to \mathbb{R}$ is an objective function and at least one of $g(\mathbf{x})$ and checking the feasibility of $\mathbf{x} \in \mathcal{X}$ involves some element of uncertainty and can only be estimated via the simulation. One example is to minimise the expectation of a random variable, i.e., $g(\mathbf{x}) = \mathbb{E}\left[Y(\mathbf{x})\right]$, whilst requiring that another random variable obeys a probabilistic constraint $\Pr(Z(\mathbf{x}) \leq \beta) \geq \alpha$. There are a number of tutorials and reviews of this area, such as Hong and Nelson (2009), Nelson (2014), Chau et al. (2014), Jian and Henderson (2015) and Amaran et al. (2016) as well as the handbook by Fu (2014).

What makes simulation optimisation so difficult is the randomness associated with the estimation of $g(\mathbf{x})$ which prevents (2.2.1) from being solved with certainty. Even in the simplest case, in which an exhaustive search is possible (a trivial problem in deterministic optimisation), one must produce many replications to reduce the uncertainty significantly until one solution appears to be the best. This cannot be done with complete certainty without infinite effort.

Another problem faced by simulation optimisation is that the simulation can be computationally expensive to run. Therefore, many methods aim to use information efficiently. This is particularly important when facing real-time problems.

There are many methods for dealing with these problems, each assuming a more specific structure of $\mathcal{X}$ or g. This section gives a brief overview of various approaches for simulation optimisation problems. It is worth noting that if feasibility is probabilistic (such as a chance constraint), the problem becomes very difficult. Our formulation for the ARP only contains deterministic constraints, so we can focus on methods for problems with this structure.

2.2.1 Categorical Discrete Problems

If the number of alternatives, $|\mathcal{X}|$, is small enough to allow an exhaustive evaluation, statistical Ranking & Selection (R&S) methods are highly appropriate. The solution space, $\mathcal{X}$, does not need to have a special structure or neighbourhood relationship as these methods perform an exhaustive search, simulating all solutions initially before allocating additional computational budget to make statistically meaningful comparisons. There are many versions of this, both frequentist (such as Kim and Nelson (2001)) and Bayesian (such as Frazier (2014)). The aim is to achieve a high probability of correct selection, a high probability of a good selection or to minimise some posterior loss between the selected solution and the best solution. Traditionally, R&S was for small solution sets, though modern methods can now handle large numbers of alternatives, more than 10,000, utilising parallel computing (Pei et al., 2018). However, they require one to list all possible solutions, which is not straightforward within the ARP.

2.2.2 Discrete Problems

There are many discrete problems for which an exhaustive search is not possible. In this case, some element of searching the space becomes vital. These methods consider the problem where

$$\mathcal{X} \subseteq \mathbb{Z}^n$$

and often make use of $\mathcal{X}$ being an ordered set. The many issues of deterministic optimisation now become important to the problem, such as not finding the optimum. Methods can largely be grouped into local and global optimality seekers.

One important class of algorithms, covering both local and global optimisation, are the Adaptive Random Search (ARS) algorithms, for which Hong et al. (2014), Andradóttir (2014) and Amaran et al. (2016) provide reviews. At iteration j these methods sample a set of solutions, $\mathcal{X}_j$, from $\mathcal{X}$ according to some distribution $F_j(\,\cdot\,|\mathcal{M}_j)$, where $\mathcal{M}_j$ is the past information kept, such as the estimated objective value of some recently simulated solutions. The solutions in $\mathcal{X}_j$ are allocated replication budgets to estimate their performance or another value function, $v(\mathbf{x})$. The solution with the best value is stored as the current solution, $\mathbf{x}_j^*$. The differences between algorithms come in the sampling rule (both the distribution and size of $\mathcal{X}_j$), the information retained, $\mathcal{M}_j$, and the value function $v(\mathbf{x})$.

Global ARS keeps $F_j(\,\cdot\,|\mathcal{M}_j)$ positive over much of $\mathcal{X}$ throughout the process, as it must balance the exploitation of a good region with the possibility that another region may be better. Examples of this are the stochastic ruler (for example, see Nelson (2014)) and simulated annealing (for example, see Andradóttir (2014)), both of which take a single sample from $F_j(\cdot |\mathcal{M}_j)$.

By reducing the target to local optimality, local ARS methods need not keep exploring the whole of $\mathcal{X}$. Instead, methods like COMPASS (Hong and Nelson, 2006) and AHA (Xu et al., 2013) focus on a 'most promising area', $\mathcal{X}_j^*$, outside which $F_j(\cdot |\mathcal{M}_j)$ is zero. These areas are designed to include points near the best current solution, $\mathbf{x}_j^*$, but excluding other points that have already been visited. COMPASS takes $\mathcal{X}_j^*$ to be all the points closer to $\mathbf{x}_j^*$ than other visited points (by Euclidean distance). AHA uses the largest hyper-box containing $\mathbf{x}_j^*$ with no other visited points in the interior of $\mathcal{X}_j^*$. Both of these examples have proven convergence properties in the sense that

$$\Pr\{\mathbf{x}_j^* \text{ is not locally optimal infinitely often}\} = 0.$$

Furthermore, these have useful stopping criteria where one can statistically test whether $\mathbf{x}_j^*$ is locally optimal or not using a standard hypothesis test to help confidence in the finite-time results.

An approach that has grown in popularity over recent years is the use of Gaussian Random Fields. The SKOPE method proposed by Xu (2012) uses a stochastic kriging meta-model within AHA to select solutions 'more likely' to add improvement. At each iteration, a Latin Hypercube Design is combined with all previous observations to fit the meta-model over $\mathcal{X}_j^*$. Solutions are sampled uniformly from $\mathcal{X}_j^*$, but rather than simulating all of these, full simulation observations are only made at the solutions with a high probability of being the best, assessed by Monte Carlo sampling from the conditional distribution defined by the meta-model. Results suggest this approach improves the performance of AHA. Salemi et al. (2019) goes further, proposing the direct use of a Gaussian Markov Random Field (GMRF) which simplifies the dependence structure by only allowing adjacent solutions to have non-zero relations in the model's precision matrix. The GMRF parameters are fitted using a starting design, after which the Expected Improvement (accounting for all sources of uncertainty) is used to select the new point. This is proven to converge under mild conditions.

An alternative approach to a discrete problem is to use linear interpolation to make it a 'continuous' problem. One example of this is R-SPLINE (Wang et al., 2013). The interpolation allows gradient information to be used to produce a new point via a line search. This is followed by an evaluation of the solution's neighbourhood in $\mathcal{X}$.

Whilst the ARP can be formulated as an optimisation problem over $\mathcal{X} \subset \mathbb{Z}^n$, as it is for all IP formulations, there are difficulties in applying either ARS, GMRFs or interpolation methods. These primarily arise from the essentially categoric, rather than integer ordered, variables and combinatorial constraints associated with aircraft allocation, which mean that a single variable may not be independently varied whilst retaining a feasible solution.

In addition to these specialist simulation optimisation techniques, there are a range of heuristic methods. Many are inspired by metaheuristic methods in deterministic optimisation, some of which are known to be appropriate for deterministic combinatorial optimisation problems. These include simulated annealing, evolutionary computing, ant colony optimisation and tabu search. The extension of these metaheuristics to stochastic combinatorial optimisation problems is reviewed by Bianchi et al. (2009), including when the objective function must be estimated by simulation.

Many applications of simulation optimisation to discrete or combinatorial problems use metaheuristics as the base algorithm. In a recent review of simulation optimisation in semiconductor manufacturing, Ghasemi et al. (2018) list many papers applying genetic algorithms, particle swarm algorithms and tabu search. As part of their study of using symbiotic simulation for semiconductor manufacturing, Aydt et al. (2011) use an evolutionary algorithm to optimise tool operations. Can et al. (2008) investigate the use of genetic algorithms for the Buffer Allocation Problem using various selection and combination methods. The authors find that whilst the algorithms generally perform well, they are sensitive to population size and the number of 'elite' solutions. Lee et al. (2007) use a multi-objective genetic algorithm for retiming flights in an airline schedule to improve its robustness.

The primary problem with using metaheuristics is that they do not naturally account for uncertainty in the estimation of the objective function. Some implementations just fix a replication number for evaluation throughout. This could be a considerable problem in scenarios where there are many sources of variability, such as semiconductor manufacturing (Ghasemi et al., 2018). Bianchi et al. (2009) discuss this issue and mention that the method used for comparing two solutions has a big impact on performance. If the uncertainty is not sufficiently controlled then the algorithms can produce very bad solutions. Methods for error control include adding statistical hypothesis tests before accepting a solution, growing the sample size with each iteration and adapting the sample size according to the results of hypothesis tests. In the case of simulated annealing, Ball et al. (2018) propose a method to sequentially choose the number of replications at each solution whilst maintaining the core properties of the deterministic version of the algorithm. Other approaches include simheuristics (Juan et al., 2015), which aim to extend the metaheuristics to simulation optimisation in a different manner. These are discussed further in Section 2.3.1. The uncertainty issue is often not handled well in implementations of some metaheuristics, such as those commercially available. To aid this, Section 2.8 of Hong et al. (2014) considers three ways to improve the performance. Firstly, use a preliminary experiment to estimate the number of replications required to distinguish between a good and a bad solution. Secondly, restart the optimisation multiple times. Thirdly, use a R&S procedure to give some statistical confidence that the chosen solution is the best of those the algorithm found (Boesel et al., 2003).

2.2.3 Continuous Simulation Optimisation

A very different set of algorithms have been developed for the case when $\mathcal{X} \subset \mathbb{R}^n$. The algorithms can be grouped into three main categories: Sample Average Approximation, Stochastic Approximation and meta-modelling based approaches.

The Sample Average Approximation (SAA) turns the stochastic optimisation problem (2.2.1) into a deterministic one. For a full discussion of this method, see Kim et al. (2014). Suppose we could write

$$g(\mathbf{x}) = \mathbb{E}\left[Y(\mathbf{x}, \xi)\right]$$

where Y is now a deterministic function and ξ contains all the random elements of the problem. Suppose the distribution of ξ is independent of $\mathbf{x}$. The SAA method is to sample $\xi_1, ..., \xi_N$ (i.i.d. samples of ξ) before the optimisation begins and then solve the problem

$$\min_{\mathbf{x}\in\mathcal{X}} \hat{g}_N(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{X}} \frac{1}{N}\sum_{i=1}^{N} Y(\mathbf{x}, \xi_i). \tag{2.2.2}$$

If the sample functions $Y(\,\cdot\,, \xi_i)$ (and thus $\hat{g}_N$) have a special structure, then (2.2.2) can be solved using a specialised deterministic optimisation algorithm. The key to the applicability of SAA is in showing that the sample functions have some exploitable properties on most of $\mathcal{X}$, such as convexity, that is preserved in the limiting function of $\hat{g}_N(\mathbf{x})$ as $N \to \infty$. Once this is achieved one must select the number of samples N and the deterministic algorithm used. Kim et al. (2014) discuss the convergence properties and results, particularly as $N \to \infty$.

Stochastic Approximation (SA), see for example Chau and Fu (2014), is the analogue of deterministic gradient descent optimisation (Chapter 3 of Nocedal and Wright (2006)). It makes use of gradient estimations to create a new solution by

$$\mathbf{x}_{j+1} = \mathcal{P}_{\mathcal{X}}\left[\mathbf{x}_j - a_j\widehat{\nabla} g(\mathbf{x}_j)\right] \tag{2.2.3}$$

where $\{a_j\}_{j=1}^{\infty}$ is a sequence of positive real numbers, $\mathcal{P}_{\mathcal{X}}[\mathbf{x}]$ is the projection of $\mathbf{x} \in \mathbb{R}^n$ onto $\mathcal{X}$, and $\widehat{\nabla} g(\mathbf{x})$ is an estimate of the gradient $\nabla g(\mathbf{x})$. The primary choices are in a_j and the gradient estimator, both of which strongly affect performance. Many variants of SA attempt to reduce the sensitivity to a_j in particular. The gradient estimators are often based on finite differencing, such as the Simultaneous Perturbation Stochastic Approximation method (Spall, 1992). In this case, a vector $\mathbf{s}_j = (s_j^1, ..., s_j^n)^T$ is sampled from a zero-mean, symmetric distribution (with finite inverse moments) and used to perturb each component of $\mathbf{x}_j$ simultaneously, with the estimate defined component-wise

$$\widehat{\nabla} g(\mathbf{x}_j)^k = \frac{Y(\mathbf{x}_j + c_j\mathbf{s}_j, \xi_j^+) - Y(\mathbf{x}_j - c_j\mathbf{s}_j, \xi_j^-)}{2c_j s_j^k}.$$

This means only two function evaluations are required, as opposed to $2n$ for the traditional 'one at a time' finite differencing. These can be noisy, though the use of multiple replications and Common Random Numbers (CRN), $\xi_j^+ = \xi_j^-$, can reduce estimator variance. An alternative is the use of direct, or white box, gradient estimators. These are not always available as they rely on knowledge of the simulation model structure itself. An example is Infinitesimal Perturbation Analysis (IPA), as discussed by Johnson and Jackman (1989). Here one considers the effect of a small change of a decision variable on a sample path from the simulation, assuming this small perturbation retains the order of the simulation events. The subsequent effect on the objective observation can be considered an observation of a partial derivative at the current solution. Combining over perturbations in each decision variable and averaging over multiple sample paths gives a gradient estimate.

Another variation is the choice of the final solution. Some algorithms average over the iterated solutions $\mathbf{x}_j$ (or the last m of them), aiming to surround the optimum. This reduces the sensitivity to the initial step size.

SA algorithms are focussed on gradient information, so cannot easily use the curvature information to improve the proposed new step.

The final class of algorithms is related to the use of meta-models to approximate the objective function. As simulations can be expensive to run, meta-modelling is an important tool for simulation analysis, with the meta-model being built based on a few simulation observations and giving an indication of the performance of solutions not yet evaluated (Barton, 2015). In simulation optimisation, the meta-models are used as a surrogate objective function and are optimised to produce a new solution. There are many forms of meta-model. Polynomial regression models (see for example Kleijnen (2005)) perform well at a local level, but rarely produce good global models. For global meta-modelling, stochastic kriging models (see for example Staum (2009)) are popular, though quadratic interpolation (Deng and Ferris, 2006), artificial neural networks and symbolic regression have also been used (Can and Heavey, 2012).

The use of global meta-models for optimisation has seen increasing attention in recent years, generally through Bayesian Optimisation (see for example Frazier (2018)). This was developed for optimisation of computationally expensive functions and has its roots in Bayesian Decision Theory (Frazier, 2010). The objective is treated as a random function, about which one learns through evaluating feasible solutions. There is 'extrinsic' uncertainty in the function at any $\mathbf{x}$ not yet evaluated, but $g(\mathbf{x})$ can be predicted through its relationship to previously evaluated solutions. A Gaussian Process prior probability distribution is placed on the objective function. Thus, given a set of previously evaluated solutions, the extrinsic uncertainty of $g(\mathbf{x})$ at any solution $\mathbf{x}$ not yet evaluated is characterised by the posterior distribution (which is often Normal). At each iteration, an acquisition function measuring the 'benefit' of evaluating g at $\mathbf{x}$ is maximised to select which solution to simulate next. Once the new solution is evaluated, the Gaussian Process is updated to include the new information, updating the extrinsic uncertainty at each unevaluated solution. Much of the early work was focussed on deterministic functions.

For stochastic simulation, additional (intrinsic) noise must be considered, and so the meta-model also includes a white-noise element and the acquisition function must account for this. Huang et al. (2006b) augment the deterministic Expected Improvement acquisition function (Jones et al., 1998) to approximately account for intrinsic variability and allow the best solution to be re-evaluated (reducing the uncertainty in the prediction at the current best). Salemi et al. (2019) suggest a criterion which exactly averages over both the ‘extrinsic’ and ‘intrinsic’ uncertainty, known as the Complete Expected Improvement. An alternative approach uses the Knowledge Gradient acquisition function, proposed by Frazier et al. (2009). Unlike Expected Information, Knowledge Gradient allows for the fact that sampling a new solution may change the current best by changing the shape of the meta-model, even if the new solution is worse than the current best. Scott et al. (2011) extended the Knowledge Gradient method to continuous simulation optimisation, showing that asymptotically, the uncertainty in the meta-model tends to zero. Knowledge Gradient is often more complex to calculate than the augmented Expected Information acquisition function. Frazier et al. (2009) resort to discretising the space whilst Scott et al. (2011) propose an approximation. Both Expected Improvement and Knowledge Gradient algorithms were extended to account for Input Uncertainty by Pearce and Branke (2017). Other acquisition functions are reviewed by Frazier (2018) and Jalali et al. (2017), the latter investigating the relative performance of algorithms in the setting of heterogeneous noise.

The downside of Bayesian Optimisation is that it has high computational overheads. Fitting and updating the Gaussian Process meta-model is expensive in high dimensional problems, and becomes more so as new observations are added. Therefore, Huang et al. (2006b) recommend this simulation optimisation approach when the simulation is more expensive than fitting the meta-model.

For local meta-models, Response Surface Methodology (RSM) is a common approach, originally proposed by Box and Wilson (1951) for physical experiments. It assumes that the observation noise is independent, homoscedastic and Normally distributed, assumptions that are not usually valid within simulation (Kleijnen, 2014). At each iteration the algorithm fits a local first-order polynomial using a Resolution III experimental design (Montgomery, 2009) and Ordinary Least Squares (OLS) regression. This model is used to define a search direction (the negative gradient). The search takes steps in this direction, evaluating solutions with the simulation, until there is no further decrease. Then a new meta-model is built around the best solution found so far. This continues until curvature is detected in the meta-model, such as the R^2 statistic being too low. At this point a Central Composite Design (CCD) is used to fit a second-order model, which is differentiated to find an estimated optimal.

There are several important issues that must be resolved before or during implementing RSM. How large is the region over which a design is laid? How large are the steps to find the next solution? How many replications at each design point are required? One way to address some or all of these issues is through combining RSM with deterministic trust-region optimisation (Conn et al., 2000).

Deterministic trust-region optimisation sequentially approximates the objective function g around the current solution $\mathbf{x}_j$ with a meta-model r_j, usually a quadratic obtained from the gradient and Hessian of g. A sub-problem of minimising r_j over a local region, known as the trust region, is (approximately) solved to propose a new solution $\mathbf{x}_j^*$. This is accepted based on a ratio comparison test,

$$\rho_j := \frac{g(\mathbf{x}_j) - g(\mathbf{x}_j^*)}{r_j(\mathbf{x}_j) - r_j(\mathbf{x}_j^*)}, \tag{2.2.4}$$

comparing the true decrease with the predicted decrease. If $\rho_j \geq \eta_0 > 0$, $\mathbf{x}_{j+1} = \mathbf{x}_j^*$, otherwise $\mathbf{x}_j^*$ is rejected and $\mathbf{x}_{j+1} = \mathbf{x}_j$. The trust-region size is adapted based on ρ_j. If $\rho_j < \eta_0$, the trust region shrinks. If $\rho_j \geq \eta_0$ the trust region could be expanded or stay the same. Incorporating a trust region into RSM then determines the size of the experimental region and the step size. There are two approaches combining both trust-region optimisation and RSM.

Deng and Ferris (2006) propose the noisy Unconstrained Optimisation By Quadratic Approximation (UOBYQA) method, mixing RSM with trust-region ideas. Rather than regression, the meta-model $\hat{r}_j$ is a quadratic interpolation between a set of design points. To handle the noise in the meta-model estimation, noisy UOBYQA sequentially allocates additional replications to each design point using a Bayesian approach. The uncertainty in the position of $\mathbf{x}_j^*$ is approximated by sampling meta-models from the posterior distribution of $\hat{r}_j$, and considering the variance of the corresponding proposed solutions. Once this variance is small enough (relative to the size of the trust region) the meta-model is used to propose a new solution. The uncertainty in whether to accept the new solution is dealt with using a simplified Optimal Computing Budget Allocation (OCBA) R&S (see for example Chau et al. (2014)), until the probability of correct selection is above a certain threshold.

The Stochastic Trust-Region Response Surface Method (STRONG) proposed by Chang et al. (2013) uses a more traditional RSM approach. It uses first-order models when the trust region is large, switching to a quadratic below a threshold, both fitted using OLS on either fractional factorial designs or CCD. The uncertainty in the meta-model is reduced in the 'inner-loop' mechanism which augments the CCD with new design points and more replications as the trust region shrinks. To be accepted, a new point must pass both $\rho_j \geq \eta_0$ and a Welch's t-test (Welch, 1938). Unlike in noisy UOBYQA, the Type I error rate decreases as the algorithm progresses. A more detailed description of STRONG is given in Chapter 4.

There are further variants of STRONG. STRONG-X (Chang, 2015) is designed to improve its efficiency. Firstly it allows the use of CRN within the experimental design. Secondly it scales the experimental design to reduce the meta-model bias, whilst retaining the same trust region. The design is also built sequentially, starting with a Resolution III design, and building up to a Resolution V if required. This was shown to outperform STRONG in numerical experiments. Chang (2015) also extended STRONG-X to handle non-normal distributions, by removing the Welch's t-test and ensuring that the number of replications used to estimate ρ_j grows as j increases. A downside of building fractional factorial designs is that they become computationally expensive when the dimension of the solution space is large. Chang et al. (2014) integrated a screening step into STRONG to create STRONG-S, using experimental designs to pick out the important factors influencing g in the neighbourhood of $\mathbf{x}_j$ before fitting meta-models including only important dimensions. This extension allowed STRONG-S to handle hundreds of decision variables. Wang et al. (2016) propose an alternative, RCB-STRONG, which partitions the dimensions into blocks. At each iteration, a block of coordinates is selected at random and the standard STRONG mechanism is applied exclusively to this group of coordinates.

It is possible to make use of direct gradient observations in RSM as well. STRONG allows these to be used for the gradient estimation instead of regression. Li and Fu (2018) propose an algorithm to augment RSM by fitting the linear model to both simulation and direct gradient observations, with the user providing a weighting. Initial experimental results suggest this additional information can be very valuable when the simulation observations are noisy.

2.3 Multi-fidelity Modelling

Due to complexity and many sources of uncertainty within the airline industry, simulation is a very natural modelling paradigm for airline operations. Thus, simulation optimisation should be a natural means of approaching the ARP. There are many and varied methods for simulation optimisation, some of which are reviewed in Section 2.2. However, there are few methods that can directly handle the combinatorial constraints of aircraft allocation. Combinatorial problems have received a lot of attention in the deterministic optimisation literature, as demonstrated in Section 2.1. The deterministic ARP approaches cannot account for the uncertainty of the real problem but can offer valid information about the qualities of a good solution to the full problem. Therefore, the method proposed in this thesis makes use of optimisation techniques for both deterministic and simulation models. This cross-disciplinary approach to a problem is known as multi-fidelity modelling. This section reviews the methods for combining models (particularly when one is a simulation model) within the literature.

Multi-fidelity modelling aims to make use of a combination of models varying in complexity and fidelity. Here, ‘fidelity’ refers to the ability of the model to reproduce the input-output relationships of the real-world system. It is often the case that making simplifying assumptions to a problem can reduce the computational complexity of evaluating the model at a specific solution. Due to the assumptions and approximations within the simplified model, it is likely that there will be a significant error in the performance estimation of a solution. However, if the model contains sufficient detail, it may hold useful information about how a solution performs relative to other feasible solutions. In multi-fidelity modelling, this is exploited by using a model with many simplifications (known as the low-fidelity model or LFM) to quickly identify promising candidate solutions. These solutions can then be evaluated in a more detailed, high-fidelity model (HFM), capable of producing a much more accurate picture of the solution’s performance. If the HFM is computationally expensive, multi-fidelity modelling allows the HFM to be used selectively, guided by the LFM.

The use of different levels of fidelity can enhance the understanding of the problem domain. Whilst considering the problem of positioning bicycles at stations for a bike sharing problem, Jian and Henderson (2015) use three models of quite different characteristics: a fluid model, a continuous time Markov chain model and a discrete event simulation. Whilst each has varying fidelity and simplifications, each of the first two expose a characteristic of a good solution and so can be used to provide a good starting point for the simulation optimisation.

A simple approach used in some cases is to optimise the LFM and evaluate the solution using a HFM. Glankwamdee et al. (2008) solve either a stochastic program or a minimax formulation for the production distribution of gas products, before simulating the resulting solution for a more detailed evaluation, with distributions over demand and failures.

However, Xu et al. (2016) point out that a low-fidelity model may be a poor performance predictor due to the error caused by the simplifications. More importantly, the error may not be uniform across the solution space: the LFM may underestimate the performance in some areas whilst overestimating in others. This unknown and unpredictable bias implies that the optimum within the LFM-based problem may not be optimal for the HFM problem. Thus, solving the simplified problem and simulating the results is naïve as it could miss out on much better solutions. Applying the HFM selectively at multiple solutions during the search phase encourages the algorithm to consider the discrepancies between the models rather than assuming a priori that the errors do not change the ranking of solutions.

Linz et al. (2017) propose a two stage Random Search approach. In the first stage, a Random Search is applied to optimise the LFM. From this, a set containing sufficiently promising solutions is approximated using hyper-boxes. The random search is repeated on this set using the HFM. If the performance of the models are similar, focussing on this promising region increases the performance of the Random Search algorithm compared to using the Random Search over the entirety of $\mathcal{X}$. This is applied to deterministic problems.

2.3.1 Iterative Methods and Simheuristics

One common approach is to iterate between the models until a satisfactory solution (as defined by the simulation output) is found. The satisfactory solution is normally based around robustness or resilience. These methods include simheuristics, a class of algorithms designed to combine simulation and deterministic models to handle stochastic combinatorial optimisation problems (Juan et al., 2015). A metaheuristic search is applied to the deterministic version of the problem and if a promising solution is found, a small number of simulation replications are used to evaluate the solution quality and feasibility (should there be probabilistic constraints). The solution is then ranked amongst previous solutions. If more time is available, the search continues. Once the search budget is used up, elite solutions are simulated more intensely for a refined quality assessment. There are many applications of simheuristics to real problems involving combinatorial constraints and random elements. The most recent review of simheuristics for problems in transportation and logistics is given by Juan et al. (2018). This section also gives examples of iterative methods that share some, but not all, features with simheuristics.

Hatami et al. (2018) use a simheuristic for job scheduling in a parallel flowshop with stochastic processing times. An iterated local search is used to find promising solutions to the deterministic problem (processing times set to their mean), looking to minimise the makespan. A Monte Carlo simulation is then applied to estimate either the expected makespan or a percentile. Which is better depends on the user preference. A similar approach using a local search heuristic is applied by Onggo et al.

(2019) to agri-food supply chains with stochastic demand and perishable products, involving both inventory control and vehicle routing. In addition, a construction heuristic is used to ensure a good initial solution is found. The approaches of Arias et al. (2013) and Guimarans et al. (2015) to the ARP discussed in Section 2.1.2 are also simheuristics.

Some iterative methods use the simulation output to modify the low-fidelity problem slightly. Gonzalez-Martin et al. (2018) use a simheuristic when considering the capacitated arc routing problem with stochastic demands. A metaheuristic is applied to the deterministic version with expected demand, though the vehicle capacities are reduced by a factor of k. The unused capacity acts as safety stock when the solution found by the metaheuristic is simulated with Monte Carlo simulation. The value of k is then adjusted, making the deterministic problem more conservative. The best solution from each round is returned. In an A&E staffing problem, Izady and Worthington (2012) use an $M(t)/G/\infty$ queueing model to estimate the required staffing levels at each station to achieve a desired quality of service β. The solution is then simulated to see if the percentage of patients discharged within 4 hours achieved the target. If not, β is altered to provide a new staffing schedule. This is repeated until the target is met. The physics of the low-fidelity model is used to guide the optimisation. Scala et al. (2017) use a combination of a deterministic model and simulation when scheduling aircraft movements at an airport on a rolling time window. As this is a combinatorial problem, simulated annealing is used to solve the deterministic model, producing an optimal solution that is evaluated by the simulation. If the number of aircraft conflicts is high when stochasticity is included, the window size is shortened before the deterministic model is solved again. The work of Scala et al. (2018) improves this by updating the parameters of the metaheuristic in each iteration. In an application to production planning in semiconductor manufacturing, Bang and Kim (2010) change the aggregation structure and flow-time parameters in a low-fidelity (by aggregation) LP if the LP and HFM simulation predictions of the throughput and flow-time differ by more than 10%. This appears to benefit in both computation time and quality more than simply changing the LP aggregation structure.

2.3.2 Using the Simulation in the Search

The methods discussed in the previous subsection iterate between the LFM and the HFM. This does not necessarily account for the unpredictable bias associated with the LFM described by Xu et al. (2016). Other approaches aim to directly utilise the simulation model as part of the search.

Xu et al. (2014) propose a method known as MO^{2}TOS. This begins by ranking all solutions in $\mathcal{X}$ according to the LFM evaluation. This has two benefits: it highlights promising solutions and transforms the solution space to a one-dimensional problem. In the second stage the method applies the HFM to randomly sampled solutions. To counteract the unknown bias, the probability of selecting a solution is proportional to its low-fidelity ranking, thus potentially exploring the whole space. Unfortunately, this work assumes that all options are exhaustively searched and evaluated with the low-fidelity model and the output has no noise. Neither of these assumptions are valid for the ARP problem.

Osorio and Bierlaire (2013) and Osorio and Chong (2015) use multi-fidelity models together in constructing a meta-model for a traffic signalling problem. Microscopic traffic simulators can be extremely expensive and so hinder optimisation problems. The LFM is an analytical finite capacity queueing model. The meta-model is constructed from the analytical objective, $T(\mathbf{x})$, and a quadratic error term:

$$\hat{r}_j(\mathbf{x}) = \hat{\alpha}T(\mathbf{x}) + \hat{\beta}_0 + \sum_{k=1}^{n} \hat{\beta}_k x^k + \sum_{k=1}^{n} \hat{\beta}_{k+n} \left(x^k\right)^2 . \tag{2.3.1}$$

The parameters $\hat{\alpha}$ and $\hat{\beta}_k$ are estimated using weighted least squares regression from all observations at each solution simulated so far, with weights inversely proportional to the distance from the current solution. The overall algorithm uses a trust-region framework, where each new solution is simulated once and the meta-model parameters updated after each iteration. Results suggest that this combination outperforms the algorithm based solely on a quadratic meta-model.

This work was extended to include two levels of simulation (one modelling the local area and the other a wider region) by Osorio and Selvam (2017). The boundary conditions cause the local model to be less accurate than the regional model, but is much more computationally efficient. The algorithm must now choose which simulation to run at each iteration. The analytical queueing model is used to estimate the error caused by the changes in boundary condition, along with a quadratic error term, similar to Equation (2.3.1). If the predicted error between the two models is small enough, the local model is used, otherwise the regional simulation is used. The meta-model for the objective function is a quadratic.

The MFSKO algorithm (Huang et al., 2006a) models the relative errors between a set of m models of varying fidelity with a kriging meta-model. Modelling the errors produces a Gaussian Process meta-model for the highest fidelity model that uses information from all systems. An initial Latin Hypercube design is used to fit initial models and the kriging hyper-parameters. The next solution is proposed using an augmented Expected Improvement that balances the cost advantages of using the LFM with the reduced quality information. Inanlouganji et al. (2018) proposed a similar approach, modelling the low-fidelity output and the bias between the models as Gaussian Processes, though in this case the Expected Improvement only accounts for the HFM, and a hypothesis test is used to see if the bias relationship holds in a statistical sense at the new position. If not, the HFM is evaluated to update the bias Gaussian Process.

Multi-fidelity modelling can also be used the other way round. In an application to multi-skill call centre staffing, Avramidis et al. (2010) use a simulation model to improve an IP model. The cost objective is deterministic, based on the number of agents used. However, the service level constraints are probabilistic. An SAA approach is used to estimate a sub-gradient of these constraints, which can then be added to the IP as a cut, reducing the feasible region. It is possible that this may cut out the optimal solution, but the probability of this reduces as the number of observations increases.

2.4 Using Simulation for Real-Time Control

Over the last 20 years, researchers have begun to apply simulation to operational problems in real-time, rather than system design problems. For example, Cheng (2007) considers the relocation of Fire Service vehicles when areas are left uncovered due to vehicles attending a major incident nearby. The vehicles need to be moved to reduce the expected fatality rates across the region. This is a combinatorially hard problem with legal restrictions that a solution must be found within one minute. Some examples were discussed in Section 2.1.2 for airline disruption management. In the aviation industry, Scala et al. (2018) propose the use of simulation for scheduling aircraft movements at airport runways based on a rolling time window. One prevalent methodology for real-time decision making with simulation is symbiotic simulation. This section introduces symbiotic simulation, briefly describes some applications and then discusses the issue of validating these systems.

2.4.1 Symbiotic Simulation

Symbiotic simulation is a paradigm involving a close interaction between a simulation model and the physical system it is modelling (Fujimoto et al., 2002; Aydt et al., 2008a). The simulation receives measurements and observations from the physical system to improve its representation. In turn, the physical system can use the information gained from the simulation analysis to improve its performance. A diagram of the interaction between the physical and simulation systems is shown in Figure 2.4.1. When the simulation is triggered, it can evaluate a number of possible options through

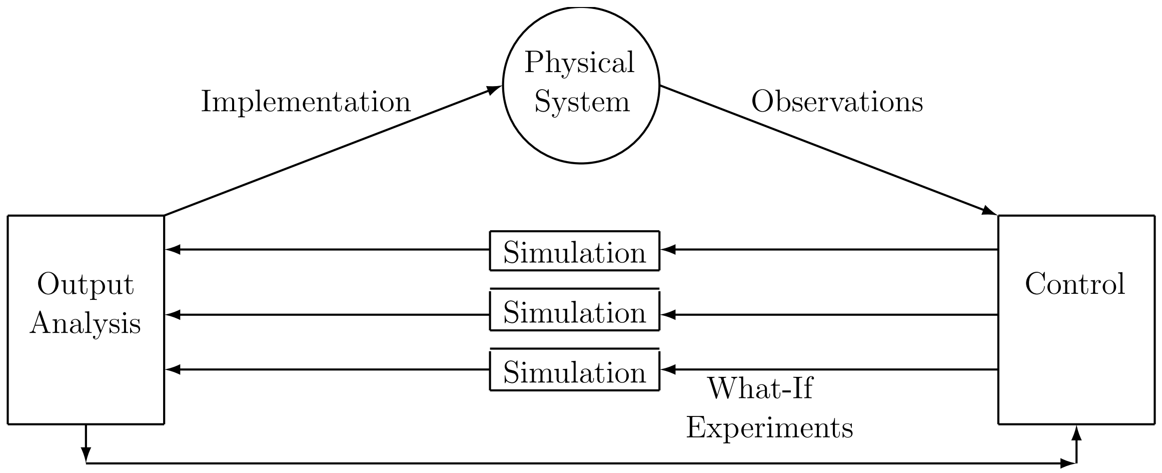

Figure 2.4.1: A diagram of a symbiotic simulation system. Adapted from Fujimoto et al. (2002).

a simulation optimisation process to aid decision makers. In a fully automated system, the findings could be implemented automatically. The simulation could be used reactively, periodically (pro-actively) or to prevent a forecasted issue (Aydt et al., 2008b). Aydt et al. (2009) review the important aspects of symbiotic simulation. Onggo et al. (2018) describe a symbiotic simulation system as a hybrid system, linking simulation with many other fields such as Machine Learning, Optimisation, Statistics and areas of computer science such as data acquisition.

The work in this thesis is based on reactive triggering to disruptive events, though it may also have value in periodic use. Furthermore, the aim is to aid decision makers, and thus have a direct impact on the physical system, known as a 'closed-loop' system (Aydt et al., 2009). There are other forms of symbiotic simulation, as discussed by Aydt et al. (2009), but these are not discussed here.

There have been many and varied applications of symbiotic simulation within the literature. The remainder of this section describes some of these, focussing on those acting as a decision support system.

Several papers consider the manufacturing sector. Aydt et al. (2011) apply symbiotic simulation to a semiconductor manufacturing plant consisting of a series of stations, each with a set of machines. Periodically, the simulation is used to help schedule configuration changes at each station for the next period, depending on the demand forecast. Fanchao et al. (2009) apply a symbiotic simulation to a lubricant supply chain. Two problems are faced. The customer order problem is triggered when certain critical states are entered. The inventory control management is triggered periodically, aiming to find the minimal number of reorder points whilst meeting the demand in a dynamic way. Chiroma et al. (2018) describe the development and implementation of a symbiotic simulation system at an automotive manufacturing plant.

There are many applications of symbiotic simulation to the transport and logistics sector (Fujimoto et al., 2016). Vu et al. (2013) propose a symbiotic traffic simulation to aid traffic management in the event of a road incident. If an incident occurs, a set of potential paths for a car is simulated using traffic forecast information. However, if every car in the city used such a device and received the same advice, the congestion may simply be moved to a different part of the city. Aydt et al. (2012) propose the use of a similar system in which data from a group of cars could be used to globally optimise the traffic system by symbiotic simulation. Sunderrajan et al. (2016) describe a symbiotic simulation that uses GPS data from traffic to control a ramp-metering mechanism for managing the flow of traffic onto a junction. The simulation is a macroscopic traffic model. Huang and Verbraeck (2009) consider how best to use new data from the physical system to perform an online calibration of a data-driven rail transport simulation system.

In a theoretical study one cannot test a symbiotic simulation system on a real physical system to evaluate its performance. Thus, the symbiotic simulations are generally run alongside another simulation designed to emulate the real world, acting as a proxy. In some cases, the proxy real world is more detailed and complex than the simulation used to make the decisions. For example, both Sunderrajan et al. (2016) and Vu et al. (2013) use a microscopic traffic simulator as the proxy real world to their simulation, which are macroscopic and mesoscopic, respectively. Using a different simulation model mimics the practical implementation, where a simulation tool will not account for all aspects of the physical system. This concept is used in Chapter 7 when considering the evaluation of simulation-based decision support systems.

2.4.2 Validation of Real-time Simulation Systems

The validation of reusable simulation models is difficult, as the initial conditions usually play a very important role in how the simulation may behave. Oakley et al. (2020) develop a symbiotic simulation to model bed occupancies across a hospital. This is run periodically to predict occupancies over the next week. To validate the model, the authors propose a method that looks at the changes in bed occupancy over time, rather than the absolute values. Looking at the changes in the system means that one does not need to validate each possible initial condition against the limited data that starts in the same state. The distribution of the change in occupancy seemed to be relatively stable across various initial conditions, making this an appropriate method for validation. The emphasis is focussed on reinforcing trust in the model rather than hypothesis testing.

If a reusable model is used over a long period of time, the model may need re-validation or re-calibration to reflect possible changes in the physical system (Aydt et al., 2011). Part of the power of symbiotic simulation is the ability to adapt the model based on new data from the real world (Onggo et al., 2018). A number of studies have looked into how to calibrate simulations to help accommodate this evolution. Huang and Verbraeck (2009) and Huang et al. (2010) consider model validation and calibration of rail transit simulation models. The primary mechanism is to periodically compare the simulation and real world outputs and, if they deviate sufficiently, update the directly observable parameters (such as train location) with the real values, and use these to infer updates for the latent values (such as velocity and acceleration). The authors suggest some mechanisms for this, but acknowledge that for stochastic output, this is much more difficult. This assumes that the model goes through a phase of validation before it is used to make predictions and decisions. A new validation phase commences if the state of the simulation deviates too much from the real world.

Papathanasopoulou et al. (2016) consider the online calibration of microscopic traffic simulators. The method proposed optimises the parameters of a (deterministic) 'car-following' model to minimise the difference between the observed and predicted speed. Unlike the static calibration method which would fit the parameters across large amounts of historical data, the online version re-fits the parameters at each time step. The results show that this method allows greater prediction power for vehicle speed than the static approach. Hashemi et al. (2017) consider a similar system, but split the problem of updating parameters between multiple ‘agents’, who control a subset of the parameters. Rather than updating all parameters each time, a reinforcement learning algorithm is used to let the agents decide whether or not to update the parameters, based on the current extent of the discrepancy and past rewards for the correction. This helps to prevent unnecessary model adjustments, reducing the number of false positives. In both cases, the experiments were on historical data rather than a real-world emulator.

2.5 Conclusions

This chapter has reviewed the current methods for the aircraft recovery problem. There are many sources of uncertainty in the airline industry which these methods cannot handle. This motivates the main research question in this thesis:

> Can stochastic simulation be used to help airlines react to and manage disruption to their schedules by finding and evaluating good potential rescheduling options?

To generate solutions, some form of search algorithm which incorporates uncertainty is needed. The simulation optimisation literature includes both discrete and continuous methods, but few of these are able to handle the combinatorial constraints and the stochasticity present in this problem well. An alternative approach is multi-fidelity modelling, and a variety of approaches and applications are discussed in Section 2.3.

In this thesis, we propose a multi-fidelity modelling approach to the ARP with two stages. Stage one uses a deterministic time-space network IP with discrete time as a LFM. The IP aims to allocate aircraft to flights and give an initial value for the planned delay of each flight with three objectives: to minimise disruption costs; minimise schedule alterations; and minimise total delay. By solving the IP in a multi-objective manner, it produces a set of rescheduling options with different priorities for these three objectives. This allows the method to solve the combinatorial aircraft allocation aspect of the problem in a deterministic manner, separating it from the simulation optimisation. This work is described in Chapter 3.

The second stage uses each solution from the LFM as a starting point for a local search for improvement in cost using simulation optimisation. This search enables the more detailed information from the simulation to direct the optimisation process. We fix the aircraft allocation for each solution and vary planned delays only. Fixing the aircraft allocation removes the combinatorial constraints from the problem. The delays are treated as continuous variables, allowing the use of continuous simulation optimisation methods and gradient information. We wish the optimisation method to be applicable beyond the specific simulation model used in this thesis, so we assume the simulation is a black box system. This discounts SAA, and concerns over high noise suggest that the more robust gradient estimators of RSM may be more appropriate than SA methods. Therefore, we apply an adapted version of STRONG with the aim to reduce the expected costs of the rescheduling option. The main description of the simulation optimisation approach is in Chapter 4, with additional details given in Chapter 6.

In addition, this chapter has considered the use of simulation for operational-level decisions. Symbiotic simulation and some of its applications were introduced in Section 2.4. Validation and re-calibration of these systems have been briefly discussed. Whilst the re-calibration methods allow the simulation to change with time, they assume that any deviation between the simulation and the real world comes from incorrect parameter settings and the detection of the discrepancy does not allow for much natural variability. Furthermore, they do not consider how to evaluate a decision support tool, which to the best of our knowledge is not discussed in the literature. Chapter 7 investigates some aspects of this question.

Chapter 3

Deterministic Disruption Problem

3.1 Introduction

As has been discussed in Chapter 2, much of the work on airline disruption management has involved the use of deterministic methods, such as integer programming. These models are well suited to representing large problems with combinatorial constraints, which are a struggle for current simulation optimisation methods. Whilst deterministic models have a limited capacity to account for the various uncertain elements of the environment, they have the potential to quickly find promising starting points within a multi-fidelity modelling approach, as proposed in this thesis. This allows the complexities of the combinatorial constraints of aircraft allocation to be dealt with in a deterministic manner.

Kohl et al. (2007) describe the practice of the Operations Control Centre (OCC) under disrupted operations. There are a variety of alterations that could be made including delaying or cancelling flights, exchanging aircraft, or ferrying aircraft (flying an empty aircraft to another airport). The latter is often prohibitively expensive so will not be considered here. The OCC first want to identify good actions, usually done by considering the three parts of the problem, aircraft, crew and passengers, sequentially. The solutions are evaluated from the perspectives of each problem component. The integration may require changes to be made due to legal constraints on each resource. Here, the focus will be on the Aircraft Recovery Problem (ARP).

Managing disruption to an airline has multiple objectives. Whilst the airline will want to minimise cost, it must also consider the impact on its passengers and potential effects that major schedule changes could have on the aircraft maintenance plans. For this reason, solving the low-fidelity model will follow a multi-objective optimisation approach, aiming to produce revised schedules on the Pareto frontier of the Integer Program problem. An additional benefit of this approach to the ARP is that several alternative schedules are produced, meaning that when the airline begins to integrate the ARP solution with the crew and passenger components it has multiple options and can implement the solution that most easily integrates across the various elements.

This chapter describes the multi-objective Integer Program (IP) developed for this research and is organised as follows. In Section 3.2, a formal problem statement for the Aircraft Recovery Problem is given. The IP model used is introduced in Section 3.3. The multi-objective optimisation approach for the IP is presented in Section 3.4. Following this is a discussion of further improvements that could benefit this method in Section 3.5. Conclusions are in Section 3.6.

3.2 Aircraft Recovery Problem

Suppose that a disruptive incident has occurred that means an airline's schedule cannot operate as intended. The airline wishes to take action to minimise the impact of this disruption, returning to normal operations within a certain time period known as the recovery window. For a short-haul or medium-haul carrier, the natural recovery window is to the end of the operating day. For an airline with long-haul flights, the choice may be more arbitrary, indicating a desired time by which normal operations will be resumed. Let A be the fleet of aircraft involved and F be the set of flights originally covered by A, 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) in advance of the departure time. Let $\mathbf{x}$ be the aircraft allocation (to be defined more precisely later) 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 its costs, delays and the number of alterations to the original schedule. The costs arise directly from delays, passenger compensation and cancellation charges. Furthermore, this plan should be achievable, avoiding over-promising which could create discrepancy, leading 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 costs are accounted for by a penalty if the actual delay exceeds d^f.

3.3 Low Fidelity Integer Program

The low-fidelity model used here is an IP adapted from the aircraft recovery model of Zhang et al. (2015). The primary simplification in the model compared with reality is the removal of all stochastic elements. The model allocates aircraft to flights, allowing delays, aircraft re-assignments and cancellations, whilst aiming to minimise costs, schedule changes and the total schedule delay.

3.3.1 Time-Space Network

A flight schedule can be represented as a sequence of flight arcs from airport to airport, and a ground arc connecting a landing to a subsequent take-off. The departure and arrival of each flight is a node representing an airport at a particular time. To optimise over possible recovery schedules, the network is augmented with additional flight arcs that start at times later than the scheduled departure (called "flight delay arcs"), the corresponding nodes, and additional ground arcs connecting consecutive nodes. The resulting directed network is known as a "time-space" network and is the basis of the IP described here. An individual aircraft's schedule is a path of flight delay arcs and ground arcs through the network. An example network of two aircraft operating between three airports is shown in Figure 3.3.1; the top panel is the original schedule, the bottom includes the delay options after a disruption with the recovery window ending at 18:00. This example is discussed in more detail later in this section.

Let V be the set of nodes, ν, each representing an airport at a potential arrival or departure time. Each potential delay of flight f is represented by a flight delay arc

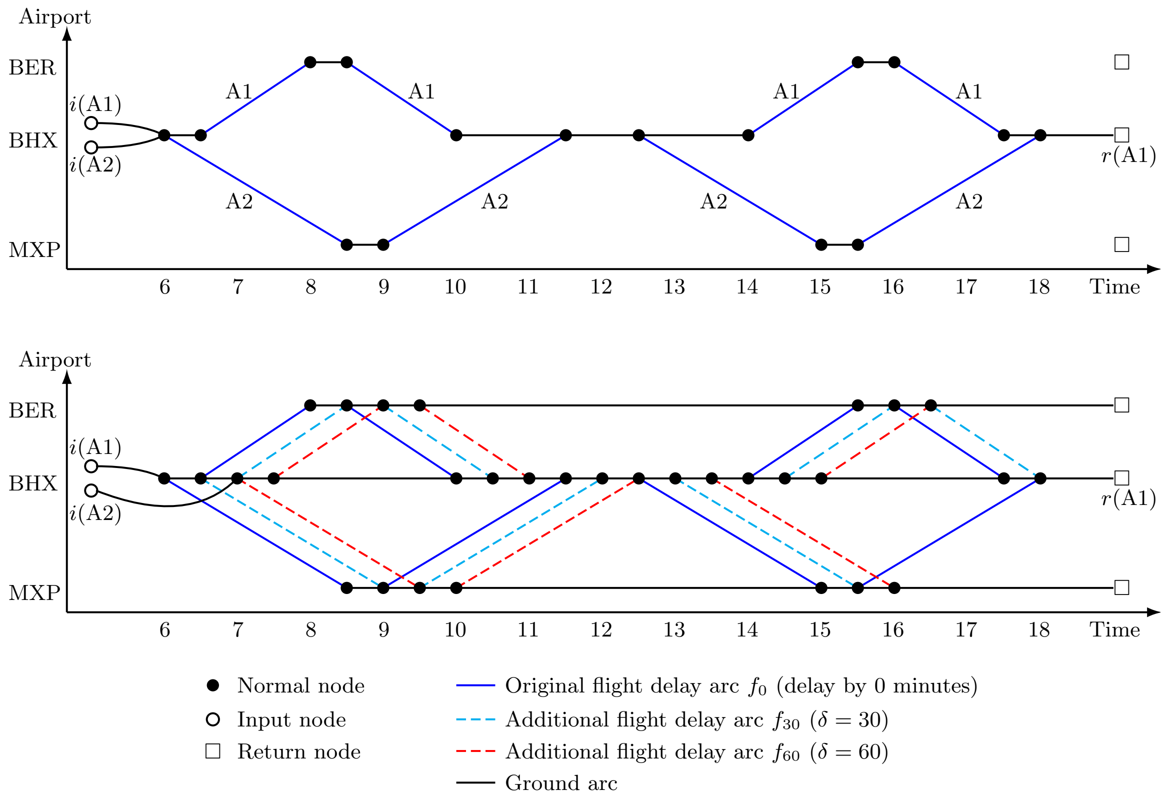
Airport
BER
BHX
MXP
i(A1)
i(A2)
A1
A2
r(A1)
Time
6
7
8
9
10
11
12
13
14
15
16
17
18
● Normal node
○ Input node
□ Return node
Original flight delay arc f_0 (delay by 0 minutes)
Additional flight delay arc f_{30} ($\delta = 30$)
Additional flight delay arc f_{60} ($\delta = 60$)
Ground arc

Figure 3.3.1: The top panel gives the original flight network, with flights labelled by original allocation. The bottom panel shows the time-space network used by the IP under a disruption.

f_δ, where δ is the intended delay of f. These potential delays are spaced at intervals of size m up to a maximum allowable delay M, that is $\delta \in \{0, m, 2m, ..., M\}$. A smaller step size leads to better solutions as the problem is less constrained, but it also increases the problem size substantially. Let L denote the set of flight delay arcs, while the set of flight delay arcs associated with flight f is L^f. Between flight delay arcs, aircraft take ground arcs, $\gamma \in G$, which connect consecutive nodes at the same airport. For each flight delay arc, its arrival and departure points are added to V, unless it coincides with an existing node.

Each aircraft a has an input node, $i(a) \in V_I \subset V$, representing its location at the beginning of the recovery window. For an aircraft in flight, this will be its destination airport, as any future flights it could be assigned to must be from its destination. No flight delay arcs exit this node and the ground arcs only connect to the first node at which a is available. Furthermore, all aircraft will end the recovery window at a return node, $r \in V_R \subset V$. 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. All nodes not in V_I or V_R are known as "normal nodes" and are contained in the set V_T. Note that $V = V_T \cup V_I \cup V_R$.

Time-Space Network Example A small example of the network structure is shown in Figure 3.3.1. The top panel shows the original network of flights. Here there are two aircraft, $A = \{\text{A1,A2}\}$, both starting at Birmingham Airport (BHX). This is represented by the two clear circular nodes labelled i(A1) and i(A2). The filled circular nodes are the departure and arrival points of the schedule, and the square nodes are the return nodes. A1 is required back at BHX at the end of the day, hence the return node of BHX is labelled r(A1), so $A_R = \{\text{A1}\}$. The day holds two flights from BHX to BER and back for A1 and two flights from BHX to MXP and back for A2, ending at 18:00.

Suppose that a slight disruption occurred preventing A2 taking off until 7:00, and let 18:00 be the end of the recovery window. The bottom panel shows the network used for the IP with $M = 60$ minutes and $m = 30$ minutes. The two additional flight delay arcs for each flight are represented by the dashed lines. This adds more potential arrival and departure nodes $\nu \in V$. Note that delaying the final BER to BHX and MXP to BHX flights of the day by 60 minutes and 30 minutes respectively would take the arrival time beyond the end of the recovery window. Thus, these flight delay arcs are not considered. The horizontal arcs represent each of the ground arcs in the model. Whilst i(A1) is connected to the node (BHX,6), i(A2) is connected to (BHX,7), as 7:00 is the earliest possible take-off time.

3.3.2 Model Constraints

The aircraft must flow continuously through the network, so we define the following sets. Let L^{ν}_{in} and G^{ν}_{in} be the sets of flight delay arcs and ground arcs incident on node $\nu \in V$, and L^{ν}_{out} and G^{ν}_{out} be the sets of flight delay arcs and ground arcs exiting $\nu \in V$. These sets will be used to ensure that an aircraft takes a continuous path through the network by forcing each aircraft to take the same number of arcs in and out of a node.

The aircraft have a minimum turn time between flights, $t_{\min}$, which is assumed here to be uniform across all airports. To deal with these turn time constraints, for each node ν, let $V_{t_{\min}}^{\nu}$ be the set of all nodes at the same airport that are less than $t_{\min}$ later than ν. The choice of $t_{\min}$ for this model affects the solution qualities. If the absolute minimum physical turn time is chosen (i.e., the lower bound of the turn time distribution), the solution will take advantage of all the available scheduled buffer, but will not be robust to uncertain turn times in the simulation. A larger choice for $t_{\min}$ would correspond to a higher quantile of the turn time distribution, leading to more robust solutions. However, if $t_{\min}$ is too large, it may introduce delays to flights that are not disrupted (as some schedules may not leave much buffer between flights).

An important part of a recovery plan is to ensure that the schedule beyond the end of the recovery window is feasible with little or no disruption. For example, a short-haul airline may have a recovery window to the end of the day, so each airport will require sufficient aircraft to fly tomorrow's schedule. This is known as aircraft balance, with r_A being the number of aircraft required to end their path through the network at return node $r \in V_R$.

An airline may only be allowed to use an airport runway during particular time slots due to the airport's runway capacity constraints. We will refer to a time period at an airport with a restriction on the number of aircraft movements as a "slot" and list these slots in the set S. Let K_s be the maximum number of aircraft movements allowed by the airline in slot s and L_s be all flight delay arcs impacting slot s. This mechanism can also be used to deal with curfew times at airports, for example, to prevent an aircraft taking off during the night.

3.3.3 Model Objectives and Variables

Our model takes into account operational delay charges from the airport and also explicit passenger compensation costs. Let the cost of delaying flight f by δ minutes be

$$c^{f_\delta} = c_d\delta + P^f(\delta). \tag{3.3.1}$$

The operational (non-passenger) costs are assumed to be linear in the delay time, with a cost of c_d per minute. As the model is deterministic, the planned delay is never exceeded and so the model does not consider the penalty charge. The compensation costs, $P^f(\delta)$, are step functions. For example, the Civil Aviation Authority (2015) state that short-haul passengers receive compensation for a 2 hour departure delay and further compensation for a 3 hour arrival delay. As the IP assumes that a flight takes the entire scheduled block time, there is no distinction between departure and arrival delays. Flight f also has a cancellation charge, C^f, comprising various costs, including passenger compensation (EUROCONTROL, 2018).

To measure the deviations from the schedule, we consider the number of flights not flown by their original aircraft. Let $o_a^f \in \{0, 1\}$ indicate whether aircraft a was assigned to flight f before the disruption occurred.

The final objective is to minimise total schedule delay, which is measured simply by noting which set of flight delay arcs is used in the solution.

The decision variables for the IP are all binary variables. Let $x_a^{f_\delta} = 1$ if and only if aircraft a is assigned to flight delay arc f_δ, $y^f = 1$ if and only if flight f is cancelled, and $z_a^\gamma = 1$ if and only if aircraft a uses ground arc γ.

3.3.4 Notation Glossary

Here is a list of the notation used in the IP.

Sets:

A :	set of all aircraft, index a
V :	set of all nodes, index ν; $V = V_T \cup V_I \cup V_R$
V_I :	set of all input nodes, $i(a)$ is the input node of aircraft a
V_R :	set of all return nodes, $r(a)$ is the return node of aircraft a
V_T :	set of all non-input and non-return nodes, i.e., $V_T = V \backslash (V_I \cup V_R)$
A_R :	set of aircraft with nodes they need to return to
F :	set of all flights in the flight schedule, index f
L :	set of all flight delay arcs, e.g. $f_\delta \in L$ delays flight f by δ
L^f :	set of flight delay arcs of flight f
G :	set of all ground arcs, index γ
L_{in}^ν :	set of all flight delay arcs arriving at node ν
L_{out}^ν :	set of all flight delay arcs departing node ν
G_{in}^ν :	set of all ground arcs arriving at node ν
G_{out}^ν :	set of all ground arcs leaving node ν
$V_{t_{\min}}^\nu$:	set of all nodes ν' at the same airport as ν and less than $t_{\min}$ later

S :	set of all slots, index s
L_s :	set of all flight delay arcs affecting slot s

Parameters:

n :	number of flights in the schedule
c_d :	cost of delaying a flight by 1 minute
$P^f(\delta)$:	passenger compensation for delaying flight f by δ
c^{f_δ} :	total cost of delaying flight f by δ
C^f :	cost of cancelling flight f
o_a^f :	indicates whether aircraft a was originally allocated to flight f
K_s :	maximum number of aircraft movements in slot s
r_A :	required number of aircraft at return node $r \in V_R$

Binary decision variables:

$x_a^{f_\delta}$:	indicates if aircraft a is assigned to flight delay arc f_δ
z_a^γ :	indicates if aircraft a is assigned to ground arc γ
y^f :	indicates if flight f is cancelled

3.3.5 Model 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).

Unlike these models, which deal with fleet assignment first and then aircraft allocation, the IP aims to allocate specific aircraft to flights. Therefore an additional explicit turn time constraint is required. The complete formulation is as follows.

$$\min \quad \sum_{a\in A}\sum_{f_\delta\in L} c^{f_\delta} x_a^{f_\delta} + \sum_{f\in F} C^f y^f \tag{3.3.2}$$

$$\min \quad \sum_{a\in A}\sum_{f_\delta\in L} (1-o_a^f) x_a^{f_\delta} \tag{3.3.3}$$

$$\min \quad \sum_{a\in A}\sum_{f_\delta\in L} \delta x_a^{f_\delta} \tag{3.3.4}$$

subject to

$$\sum_{a\in A}\sum_{f_\delta\in L^f} x_a^{f_\delta} + y^f = 1 \qquad \forall f \in F \tag{3.3.5}$$

$$\sum_{a\in A}\sum_{f_\delta\in L_s} x_a^{f_\delta} \le K_s \qquad \forall s \in S \tag{3.3.6}$$

$$\sum_{f_\delta\in L_{\text{in}}^\nu} x_a^{f_\delta} + \sum_{f_\delta\in L_{\text{out}}^\nu} x_a^{f_\delta} + \sum_{\nu'\in V_{t_{\min}}^\nu}\sum_{f_\delta\in L_{\text{out}}^{\nu'}} x_a^{f_\delta} \le 1 \qquad \forall a \in A, \forall \nu \in V_T \tag{3.3.7}$$

$$\sum_{\gamma\in G_{\text{out}}^i} z_a^\gamma = 1_{\{i=i(a)\}} \qquad \forall a \in A, \forall i \in V_I \tag{3.3.8}$$

$$\sum_{f_\delta\in L_{\text{in}}^\nu} x_a^{f_\delta} + \sum_{\gamma\in G_{\text{in}}^\nu} z_a^\gamma - \sum_{f_\delta\in L_{\text{out}}^\nu} x_a^{f_\delta} - \sum_{\gamma\in G_{\text{out}}^\nu} z_a^\gamma = 0 \qquad \forall a \in A, \forall \nu \in V_T \tag{3.3.9}$$

$$\sum_{f_\delta\in L_{\text{in}}^{r(a)}} x_a^{f_\delta} + \sum_{\gamma\in G_{\text{in}}^{r(a)}} z_a^\gamma = 1 \qquad \forall a \in A_R \tag{3.3.10}$$

$$\sum_{a\in A}\sum_{f_\delta\in L_{\text{in}}^r} x_a^{f_\delta} + \sum_{a\in A}\sum_{\gamma\in G_{\text{in}}^r} z_a^\gamma \ge r_A \qquad \forall r \in V_R \tag{3.3.11}$$

$$x_a^{f_\delta}, z_a^\gamma, y^f \in \{0,1\} \qquad \forall a \in A, f\in F, f_\delta \in L, \gamma \in G \tag{3.3.12}$$

Objective (3.3.2) relates to the cost of the recovery action, objective (3.3.3) to the number of changes made to the aircraft allocation and objective (3.3.4) to the total planned delay. Constraint (3.3.5) ensures that each flight is either flown once or cancelled. Constraint (3.3.6) is the slot constraint. Constraint (3.3.7) prevents a turn time of less than the minimum allowable turn time; if aircraft a uses a flight delay arc incident on node ν, it cannot then use a flight delay arc exiting ν or any other node ν' within $t_{\min}$ of ν. Constraint (3.3.8) is a flow constraint for the input nodes, ensuring that only one ground arc is chosen from the input node. Constraint (3.3.9) is a flow constraint for normal nodes; it ensures that the number of arcs used by aircraft a to arrive at node ν equals the number of arcs used by a to leave ν. As (3.3.8) dictates that each aircraft leaves its input once, and thus enters a specific node exactly once, (3.3.9) develops a recursion to ensure that an aircraft can enter a node either once or not at all. Constraint (3.3.10) ensures flow at return nodes when this is specified for an aircraft. Constraint (3.3.11) ensures aircraft balance at the end of the recovery window.

3.4 Solving the Integer Programming Model

The aim of the IP is to generate a set of promising solutions to be used as starting points in the simulation optimisation process. This will result in a set of aircraft allocations and initial values for the planned delays. As the overall problem has multiple conflicting objectives (for example, fewer aircraft exchanges gives less flexibility for reducing cost), we take the view that the importance of each objective is not known a priori, so presenting a range of solutions allows the decision maker to decide on the preferences. The ϵ-constraint method (Haimes et al., 1971) is used to produce several solutions by treating all but one objective as problem constraints. In this case, the objectives of minimising the number of aircraft reassignments (3.3.3) and minimising total delay (3.3.4) are added to the IP as constraints:

$$\sum_{a\in A}\sum_{f_\delta\in L}(1-o_a^f)x_{f_\delta}^a \in [l_E, u_E] \tag{3.4.1}$$

$$\sum_{a\in A}\sum_{f_\delta\in L}\delta x_{f_\delta}^a \in [l_D, u_D]. \tag{3.4.2}$$

The limits of these constraints determine the priority levels of each objective by forcing the IP solver to search particular areas of the feasible region. Thus, by changing these limits systematically and solving the IP problem for each set of limits (with cost as the objective), one can find multiple Pareto optimal solutions corresponding to different priority levels for the objectives.

In theory, if all possible combinations of l_E, u_E, l_D, and u_D are used, the entire Pareto frontier will be discovered. However, this can be a very time consuming process, sometimes leading to the same solution being found multiple times, so several algorithms have been designed to systematically vary the limits l_E, u_E, l_D and u_D to find the Pareto frontier quickly. Here we use an efficient method proposed by Laumanns et al. (2006). An example of this process is shown in Figure 3.4.1, each plot showing the result of one iteration of the algorithm.

In iteration 1, the IP is solved without (3.4.1) and (3.4.2) constraining the problem, or with a maximum value considered a limit by the airline (which may be particularly relevant for aircraft exchanges). Due to the nature of the quantities, having no constraints 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

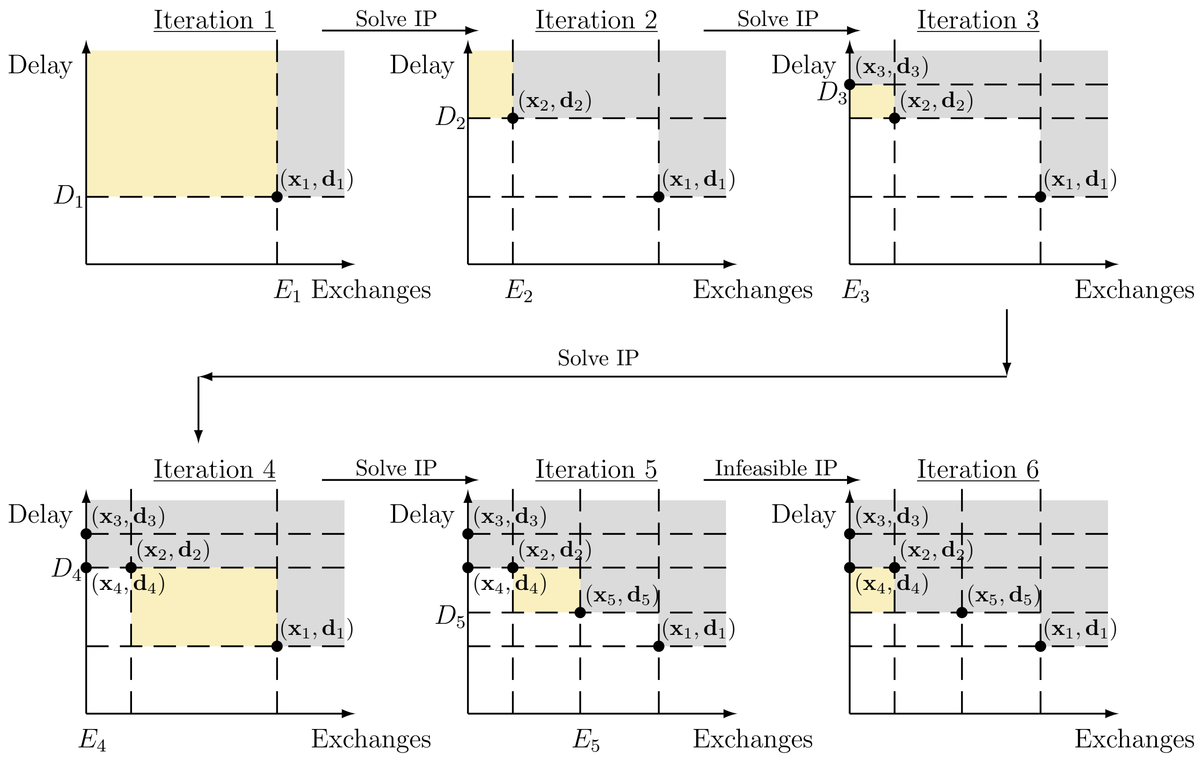

Figure 3.4.1: An example of the ϵ-constraint approach proposed by Laumanns et al. (2006).

operated using their original aircraft). This results in an optimal cost solution $(\mathbf{x}_1, \mathbf{d}_1)$ (the first plot in Figure 3.4.1). Say that this solution has E_1 aircraft exchanges and a total delay of D_1. These values are then used to partition the objective space into four sectors:

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

The solution $(\mathbf{x}_1, \mathbf{d}_1)$ is the best solution in $[E_1, n] \times [D_1, nM]$ (top right in the first plot of Figure 3.4.1) as no other solution in this region could have lower costs, exchanges or delay. Therefore, this region is considered searched, indicated by the gray shading. The algorithm does not choose limits within this region again.

In each iteration, the algorithm must select one of the unsearched sectors to define a new set of limits for constraints (3.4.1) and (3.4.2). To do this, it considers the individual sectors of the grid resulting from the partitions defined by the solutions found so far (the dashed lines in Figure 3.4.1). The selection procedure begins in the sector with the highest delay and exchanges (top right in each plot of Figure 3.4.1). It then iteratively moves right to left (reducing the number of exchanges whilst retaining the delay limits) through the sectors of a row, a row at a time, until it reaches an unsearched sector. Note that it will return an individual sector, not a combination of sectors. In the first iteration of Figure 3.4.1, moving left (and thus reducing exchanges) from $[E_1, n] \times [D_1, nM]$ leads to $[0, E_1) \times [D_1, nM]$, which is unsearched and so selected as the next region (shaded in yellow). In Iteration 2, starting in the top right and moving left along the top row takes us through one searched area and into $[0, E_2) \times [D_2, nM]$, which is unsearched and so selected for the next iteration. In the third iteration of Figure 3.4.1, the top row is entirely searched (as the solution at $E_3 = 0$), and so the selection procedure moves left along the second row of sectors; the first two are searched and so $[0, E_2) \times [D_2, D_3)$ is selected. Choosing to reduce the exchanges before the delays means that finding solutions with fewer exchanges is prioritised. Note that the ends of the interval may be open to prevent the optimisation reaching a previous solution again.

Once the region has been selected, the IP is solved with the corresponding constraint limits. If the solver finds a new solution, with E' exchanges and a total delay D', this is added to the list of solutions producing the grid partitioning the space and the searched region is extended by the sector $[E', u_E) \times [D', u_D)$ (no better solution could exist in this sector). The partitions of the space become finer with each iteration, though the minimum granularity is restricted by the discrete nature of both exchanges and delay in the IP. For our example, in Iteration 2 of Figure 3.4.1, the new solution $(\mathbf{x}_2, \mathbf{d}_2)$ has E_2 exchanges and a total delay of D_2. Therefore, the region $[E_2, u_E) \times [D_2, u_D)$ is labelled searched and more partitions are added (see the second plot in Figure 3.4.1, which now has nine regions). On the other hand, if the solver identifies the region as infeasible, and so contains no solution, its entirety will be labelled searched. Otherwise the grid does not change from the previous iteration. In our example, the transition from Iteration 5 to Iteration 6 in Figure 3.4.1, the solver could not find any solutions in $[E_2, E_5) \times [D_5, D_4)$, and so this entire region is shaded gray in Iteration 6.

Once a solution has been found (or the sector labelled infeasible) and the partitions have been updated, the algorithm selects a new unsearched region (an area shaded in yellow). This process of selecting a sector, solving to find a new solution, splitting the sectors further and blocking off searched regions is repeated, corresponding to iterations 3, 4, 5 etc. In practise, this procedure produces solutions where the number of exchanges is decreased at the expense of cost and delay, until the zero-exchange option is found (if feasible), see Iterations 1 to 3 in Figure 3.4.1. This takes at most E_1 iterations. This first 'round' is followed by a sequence of iterations with a stricter limit on the total delay objective, each working its way toward a zero exchange option before making u_D smaller again and repeating.

Given sufficient time, i.e., polynomial in the number of Pareto optimal solutions, the algorithm will eventually search the whole space and find all of the solutions on the Pareto frontier (Laumanns et al., 2006). In our application, however, the process stops after a time limit. For each sector, we use the Gurobi Optimizer 7.0.2 (Gurobi Optimization, LLC., 2017) to find the optimal solution for the IP, or the best solution so far if the time limit is reached. Due to the time limit, the algorithm is likely to leave some sectors unsearched or with a sub-optimal solution so that we do not have the complete Pareto optimal set. Furthermore, it may take time to find the first Pareto optimal result. The solver may find a sequence of cost optimal solutions that are only weakly Pareto optimal, as fewer aircraft exchanges can be made whilst achieving the same cost (this is often due to reallocating flights that are not part of the disruption). This is demonstrated in the computational experiments in Chapter 5. Thus, within the time limit, Pareto optimal solutions cannot be guaranteed.

The result is a set of solutions, $\mathcal{X}$, each with an aircraft allocation, a set of cancelled flights, and a delay value for all flights in the programme. Due to the time limitations, it is not possible to guarantee the properties of the solutions within $\mathcal{X}$. However, if more than one solution is produced, we will have a cost optimal solution and multiple aircraft allocations to offer options to the airline. On the other hand, if a cost optimal Pareto solution is found quickly this method can find a number of Pareto optimal solutions that represent the trade-off between the objectives. Whilst the solutions in $\mathcal{X}$ may be incomplete and possibly sub-optimal, they are nevertheless the result of a purposeful search over the solution space of the low-fidelity problem. As such, the solutions will be used as a set of sensible starting points within the simulation optimisation procedure, described in Chapter 4.

3.5 Improvements for the Low-Fidelity Model

The focus of this thesis is on the development and evaluation of a simulation optimisation approach to solving the ARP. The multi-fidelity approach can in theory be improved by enhancing the modelling and optimisation methods used in both the low-fidelity and the high-fidelity problems. As the remainder of the thesis will primarily consider the high-fidelity aspects, this section simply recognises and briefly describes ways in which ideas in the literature could be developed to improve the solutions to the low-fidelity problem.

3.5.1 Size and Properties of the Integer Program

Both Thengvall et al. (2000) and Hu et al. (2015) have proven their formulations of the ARP, both based on time-space networks, to be NP-hard. This is achieved by showing the relationship to another well known network IP problem which is itself NP-hard, such as the integer single-commodity network with side constraints. This complexity is widely conjectured to be a general property of the ARP. For our model, the problem size is dependent on the number of flights, n, the number of aircraft, $|A|$, and the number of possible delay options, $M/m + 1$.

The number of flight delay arcs is bounded by:

$$|L| \leq n\left(\frac{M}{m} + 1\right)$$

$$\Rightarrow \text{Number of } x \text{ variables} \leq n|A|\left(\frac{M}{m} + 1\right).$$

The inequality here is because flight delay arcs that extend beyond the recovery window are not included in the model.

The worst case scenario for the number of nodes is that every flight delay arc produces a unique pair of nodes. Each input and normal node will have an associated ground arc leaving them. In this case, the number of ground arcs is given by:

$$|G| = |A| + 2|L| \leq |A| + 2n\left(\frac{M}{m} + 1\right)$$
$$\Rightarrow \text{Number of } z \text{ variables} \leq |A|^2 + 2n|A|\left(\frac{M}{m} + 1\right).$$

Empirical observation when producing the results in Chapter 5 suggests that, in practice, the factor of 2 in front of the second term can often be replaced by a smaller constant, frequently less than 1. With discretised time and a large number of flights, the flight delay arcs that are added to the network may have departures or arrivals that coincide with those of flights that already exist (as is the case in Figure 3.3.1). This is particularly true at hub airports. This means that multiple flight delay arcs may share nodes, reducing the average number of nodes per flight delay arc to less than 2. This in turn reduces the number of ground arcs. The number of y variables is simply the number of flights, as any flight can be cancelled. It is also important to note that, unless there is a significant number of spare aircraft in the problem, n grows approximately linearly with $|A|$, particularly when all flights are of short-haul distances.

The value of m plays a significant part in the problem size. The experimental results in Thengvall et al. (2000) suggest that in their time-space network a smaller m value would not just increase the problem size, but also increase the proportion of non-integer solutions to the Linear Programming (LP) relaxation. The LP relaxation is the first stage of solving any IP by relaxing the integer constraints (3.3.12) to bound constraints on the interval $[0, 1]$. This in turn increases the solution time as a branch-and-cut algorithm (or similar) must then be used. The benefit of a smaller m is that extra opportunities for aircraft exchanges may be found to reduce the delay. An alternative approach that may reduce the complexity would be a more intelligent selection of flight delay arcs, rather than uniformly distributed. One example of this is considered in Petersen et al. (2012), who consider an *event-driven* method to creating flight strings (sequences of flights to be flown by one aircraft). The aim is to select arcs that arrive or depart at times where the conditions or options change. This is applied in a flight-string-based model (where decision variables indicate an aircraft is assigned to a sequence of flights) so additional research would be required before it was immediately applicable to this model where aircraft are assigned to individual flights.

3.5.2 Solution Methods and Problem Reduction

As shown in the previous section, the size of the fleet, $|A|$, plays a substantial role in the problem size. This suggests that there are advantages to cutting out aircraft that are unlikely to offer any potential schedule changes. For the larger problems tackled in this thesis, we have used a simple heuristic to reduce the number of aircraft considered. After identifying the set of directly disrupted aircraft, A^*, we collect the set of airports they are set to visit during the recovery window. Other aircraft are added to the problem based on whether they visit any of these airports during the recovery window.

For much larger problems, more sophisticated heuristics to reduce the problem size might be needed. An example of such an algorithm is suggested by Rosenberger et al. (2003), where a network representing the schedule is searched for cycles involving disrupted aircraft to reduce the set of aircraft included in the problem. This approach is likely to lead to a greater reduction in the size of the problem, giving it computational advantages.

A greater study of the problem structure may also allow specialised cuts to be added to the problem during a branch-and-cut procedure to improve the time to solve over a generic algorithm. Alternatively, looking at heuristic solution methods, such as Large Neighbourhood Search algorithms (Sinclair et al., 2014), may help to produce good, if not Pareto optimal, solutions more quickly than exact solution methods.

3.5.3 Extending the IP Model

Whilst the IP model covers many of the aspects of the real-world problem, there are additional features that could be included. For example, one may want to consider the total amount of passenger delay, weighted by class, as an objective, allowing one to prioritise busier or more profitable flights. Whilst we have assumed a linear delay cost model (except for compensation), other models could be used. In particular, Cook and Tanner (2015) studied the cost per minute of delay in much detail, splitting it into constituent parts and finding a non-linear relationship. This could easily be accounted for in the objective of the IP, Equation (3.3.2), by changing the definition of c^{f_δ} in Equation (3.3.1). This would not require a change in solution methodology.

Other recovery actions such as changing the cruise speed of the aircraft on certain flights could be added by the inclusion of additional flight delay arcs arriving earlier with a higher cost assigned. This action could be desirable as it can reduce delays but may increase costs due to additional fuel consumption.

It has been assumed that all aircraft are homogeneous. Additional costs and constraints could be added to account for a heterogeneous fleet, as is the case in, for example, Petersen et al. (2012). However, this can add significant complexity to a problem, and may not always be feasible due to crewing considerations.

3.6 Conclusions

This chapter describes a deterministic integer programming model and a multi-objective solution approach for the ARP. The IP model is based on a time-space network model and an ϵ-constraint based method is used to search the solution space. This approach is used to find a set of starting solutions for the simulation optimisation process to be described in Chapter 4. We believe that these solutions provide promising rescheduling options, exploring different possible aircraft allocations.

Extensions and improvements to the IP model and solution methodology are possible and have been discussed. As the main focus of this thesis is on treating the stochastic elements of airline disruption, these were not pursued.

Chapter 4

Simulation Optimisation for the Aircraft Recovery Problem

4.1 Introduction

Whilst an Integer Program (IP) can encode some of the complexities of the airline industry, it has a limited capacity for handling uncertain elements of the environment. Aircraft turn times, airport queueing times, maintenance times and flight durations all involve uncertainty, which may alter the performance of a solution. This suggests that a model of higher fidelity should be used in conjunction with the deterministic models. Probabilistic models that achieve the levels of detail required are unlikely to be analytically tractable. Thus simulation seems to be a natural way to model the airline's operations.

In some applications of multi-fidelity modelling, the simulation is primarily used for checking the robustness of promising solutions to stochasticity. However, Xu et al.

(2016) point out that the low-fidelity model predictions could have significant bias or variability. To account for this, they propose that some form of simulation optimisation takes place after the low-fidelity model has been used. Whilst their proposed method based on Ordinal Transformation is not appropriate here, the principle that there is value in utilising the simulation for further improvement motivates a search around the candidate solutions from the IP.

The purpose of this chapter is to describe a simulation optimisation approach to search for local improvements around the rescheduling options found by the IP. As the aircraft allocation involves combinatorial constraints, a difficult problem for simulation optimisation, this aspect of the solution remains fixed, whilst the disruption cost is optimised over the planned delays. A trust-region based simulation optimisation algorithm called STRONG (Chang et al., 2013) is used to perform this optimisation. As STRONG is designed for unconstrained problems, it has been tailored and extended to search within a space with bound constraints, arising from the non-negativity of delays, drawing on ideas from the deterministic constrained trust-region and experimental design literatures.

This chapter is organised as follows. The simulation model used is introduced in Section 4.2. The simulation optimisation algorithm and how it relates to the IP solutions is presented in Section 4.3. Following this, Section 4.4 discusses what is required for this method to be applicable in practice. Conclusions are in Section 4.5.

4.2 High Fidelity Simulation Model

The input 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 $\mathbf{d}$ are linked to the IP variables via the following relationships:

$$x_a^f = \sum_{f_\delta \in L^f} x_a^{f_\delta}, \tag{4.2.1}$$

$$d^f = \sum_{a \in A} \sum_{f_\delta \in L^f} \delta x_a^{f_\delta}, \tag{4.2.2}$$

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 expected cost of the disruption:

$$g(\mathbf{x}, \mathbf{d}) = \mathbb{E}\left[\sum_{f \in F} \left(c_d D^f + c_p (D^f - d^f)^+ + P^f(D^f)\right)\right] + C(\mathbf{x}). \tag{4.2.3}$$

Here, c_d is the cost per minute of delay, c_p is the penalty per minute for the actual delay D^f exceeding the planned delay d^f, $P^f(D^f)$ represents the compensation associated with passenger delays of flight f and $C(\mathbf{x})$ is the cost of the cancellations in the allocation $\mathbf{x}$. Unlike in the IP, in the simulation $P^f(D^f)$ does distinguish between arrival and departure delays, as the flight may not take its entire scheduled block time, that is the time between the scheduled departure and arrival time often contains a buffer period. This means that a flight may depart more than 3 hours after its scheduled time, but not incur the 3 hour arrival delay compensation. This is one of the few examples of the IP overestimating the cost.

The simulation model is built within AnyLogic 8.2.3 (The AnyLogic Company, 2017). 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 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), obtained using the python package (Allamraju, 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. The distribution and parameter assumptions are made as our current 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. 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 (Flightradar24 AB, 2017). The service time represents the spacing required between aircraft and is assumed deterministic given the weather conditions and time of day, taking the form

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

where $s(t)$ is the required separation time in normal operating conditions at time 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 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.

For a more in depth reporting of the simulation model, see Appendix A. Whilst the simulation does not model all the details of an airline's operations, it does contain the key features necessary to evaluate our multi-fidelity approach.

4.3 Simulation Optimisation Process

Once the IP has produced a set of revised schedules, $(\mathbf{x}, \mathbf{d}_0) \in \mathcal{X}$, each one is used as the starting point for a simulation optimisation procedure, searching for an improvement around that solution. As simulation optimisation often struggles with combinatorial constraints, we reduce the complexity of the solution space by keeping the aircraft allocation, $\mathbf{x}$, fixed throughout the process. This leaves a continuous, ordered solution space of planned delays, $\mathbf{d} \in \mathcal{D} \subset \mathbb{R}^n$. As delays cannot be negative and may have an upper limit due to other schedule considerations, $\mathcal{D}$ is a hyper-box with bound constraints on each variable. For further problem size reduction, only the n^+ flights that the IP has allocated a non-zero delay in $\mathbf{d}_0$, $F^+ = \{f : d_0^f > 0\}$, are varied in the optimisation. The solution space is now greatly simplified from the original, with the following continuous simulation optimisation problem:

$$\min_{\mathbf{d}} \quad g(\mathbf{x}, \mathbf{d}) \tag{4.3.1}$$

$$\text{subject to} \qquad d^f \in [0, u^f], \qquad \forall f \in F^+ \tag{4.3.2}$$

$$d^f = 0, \qquad \forall f \notin F^+ \tag{4.3.3}$$

where g is defined in (4.2.3), searching for a local optimum conditional on $\mathbf{x}$. As $g(\mathbf{x}, \mathbf{d})$ cannot be evaluated exactly, it is estimated by performing multiple replications from the simulation at $(\mathbf{x}, \mathbf{d})$ and taking the mean of the output, $\hat{g}(\mathbf{x}, \mathbf{d})$.

To approach the simulation optimisation problem, we extend the STRONG algorithm (Chang et al., 2013). STRONG combines classical Response Surface Methodology with ideas from trust-region optimisation for continuous unconstrained problems. The extensions are designed to enable STRONG to handle bound constraints, as we believe it is likely that the optimal solution will lie on the boundary of $\mathcal{D}$. The primary changes to STRONG are fourfold. The first is the choice of trust region. The second is the direct application of experimental design principles to build a good design matrix in a non-standard region-of-interest to estimate the meta-model. This occurs within the algorithm itself rather than offline. The third is the use of a projected-gradient to define the step, allowing movement along a boundary. The final change is the use of a different criticality condition, generalising to optimality in the constrained setting.

Trust-region algorithms are iterative optimisation techniques based on a local search (see, for example, Conn et al. (2000) or Chapter 4 of Nocedal and Wright (2006)). The j^{th} iteration uses a meta-model r_j, usually a low-dimensional polynomial, to approximate the objective g over a local region $\mathcal{B}_j$, known as the *trust region*, around the current solution, $\mathbf{d}_j$. Unlike line search optimisation techniques, r_j is not solely used to define a search direction (after which the effort is to calculate a step length); instead, r_j is only trusted to be valid within the trust region. Instead, the search takes place within $\mathcal{B}_j$, seeking a minimiser for the meta-model r_j over the trust region. This is called the sub-problem. The size of the trust region varies depending on the accuracy of the meta-model in predicting the performance of a proposed solution.

For bound constraints, Conn et al. (1988) suggests the use of a hyper-box trust region using the ℓ_∞ norm with half-width Δ_j, rather than the hypersphere used within STRONG. This choice is particularly convenient as it aligns the trust-region boundaries with the constraints. This alters the sub-problem as any possible step, $\mathbf{s}$, must also satisfy the additional feasibility constraints:

$$\begin{aligned} & \min_{\mathbf{s}} \quad r_j(\mathbf{d}_j + \mathbf{s}) \\ \text{subject to} \quad & ||\mathbf{s}||_\infty \leq \Delta_j \\ & \mathbf{d}_j + \mathbf{s} \in \mathcal{D}. \end{aligned} \tag{4.3.4}$$

4.3.1 The Sub-problem

As is the case in many continuous decision-variable optimisation algorithms, the most common choice of meta-model within trust-region optimisation is a quadratic (see Chapter 4 of Nocedal and Wright (2006)). In deterministic optimisation, the gradient is often directly available, and there are a number of methods to approximate the Hessian, such as the BFGS Hessian estimator independently proposed by Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970). Convergence of trust-region algorithms to a stationary point does not require second-order derivative information, but a good Hessian estimate does aid convergence rates.

In the simulation optimisation setting, it is rare that either the gradient or the Hessian are known, and so the meta-model must be estimated. Therefore, STRONG does not always use a quadratic model to approximate the objective, instead following the classical Response Surface Methodology simulation optimisation approach (Kleijnen, 1998), which generally uses a linear model until curvature is detected, at which point a quadratic model is used. This reduces the computational load when a linear model is sufficient to achieve improvement, as fewer parameters, and thus fewer design points, are required. There is a threshold for the size of the trust region, $\tilde{\Delta}$, above which a linear response surface is used. The value of $\tilde{\Delta}$ could be chosen based on practical knowledge of the problem or based on parameter tuning. We define the estimated response surfaces as

$$\hat{r}_j(\mathbf{d}_j + \mathbf{s}) = \begin{cases} \hat{g}_j(\mathbf{x}, \mathbf{d}_j) + \widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)^T \mathbf{s} & \text{if } \Delta_j > \tilde{\Delta}; \\ \hat{g}_j(\mathbf{x}, \mathbf{d}_j) + \widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T \widehat{H}_j(\mathbf{x}, \mathbf{d}_j)\mathbf{s} & \text{if } \Delta_j \leq \tilde{\Delta}. \end{cases} \tag{4.3.5}$$

Here, $\hat{g}_j(\mathbf{x}, \mathbf{d}_j)$ is the sample mean estimate of $g(\mathbf{x}, \mathbf{d}_j)$ in the j^{th} iteration, and $\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)$ and $\widehat{H}_j(\mathbf{x}, \mathbf{d}_j)$ are estimates of the derivative and Hessian with respect to $\mathbf{d}$ of g at the point $\mathbf{d}_j$. STRONG itself does not specify which estimators should be used for $\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)$ and $\widehat{H}_j(\mathbf{x}, \mathbf{d}_j)$, one suggestion being the BFGS Hessian estimator (Chang et al., 2013). The estimator used depends on the problem structure and the amount of information available. In this work, the simulation is treated as a black box. For this case, Chang et al. (2013) suggest the use of experimental design and regression analysis, where we simulate different points within the trust region and build a response surface meta-model. An iteration using the linear model is called a Stage I iteration, whereas the quadratic model is used in a Stage II iteration.

The next three subsections describe how the meta-model is built using methods from experimental design, the estimators themselves, and how the model is used to propose a step for testing.

(i) Experimental Design

Let $\mathbf{D}_j$ denote the design matrix used for fitting the meta-model over the trust region centred at $\mathbf{d}_j$. Each row of $\mathbf{D}_j$ corresponds to a design point, with each column representing the value of a term (e.g., intercept, main effect or interaction) in the meta-model at the corresponding design point (relative to $\mathbf{d}_j$). Suppose that there are two dimensions in the problem, i.e., $n^+ = 2$. A design matrix for a linear model with K design points $\mathbf{d}_{j1}, ..., \mathbf{d}_{jK}$ would take the form of a $(2+1) \times K$ matrix

$$\mathbf{D}_j = \begin{bmatrix} 1 & (\mathbf{d}_{j1} - \mathbf{d}_j)^T \\ 1 & (\mathbf{d}_{j2} - \mathbf{d}_j)^T \\ \vdots & \vdots \\ 1 & (\mathbf{d}_{jK} - \mathbf{d}_j)^T \end{bmatrix} = \begin{bmatrix} 1 & (d_{j1}^1 - d_j^1) & (d_{j1}^2 - d_j^2) \\ 1 & (d_{j2}^1 - d_j^1) & (d_{j2}^2 - d_j^2) \\ \vdots & \vdots & \vdots \\ 1 & (d_{jK}^1 - d_j^1) & (d_{jK}^2 - d_j^2) \end{bmatrix}$$

where d_{jk}^f is the planned delay for flight f at design point k.

For a quadratic model, the design matrix also includes columns for the interaction and squared terms:

$$\mathbf{D}_j = \begin{bmatrix} 1 & (d_{j1}^1 - d_j^1) & (d_{j1}^2 - d_j^2) & (d_{j1}^1 - d_j^1)(d_{j1}^2 - d_j^2) & (d_{j1}^1 - d_j^1)^2 & (d_{j1}^2 - d_j^2)^2 \\ 1 & (d_{j2}^1 - d_j^1) & (d_{j2}^2 - d_j^2) & (d_{j2}^1 - d_j^1)(d_{j2}^2 - d_j^2) & (d_{j2}^1 - d_j^1)^2 & (d_{j2}^2 - d_j^2)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & (d_{jK}^1 - d_j^1) & (d_{jK}^2 - d_j^2) & (d_{jK}^1 - d_j^1)(d_{jK}^2 - d_j^2) & (d_{jK}^1 - d_j^1)^2 & (d_{jK}^2 - d_j^2)^2 \end{bmatrix}.$$

Due to the constraints on $\mathcal{D}$, when the current solution is near the boundary, the whole trust region cannot be guaranteed to be feasible. In such cases, it can be difficult to achieve the desirable properties of balance and orthogonality obtained in many experimental designs, such as 2^k fractional factorial designs and Central Composite Designs. A design with good properties is required to produce a good gradient estimator. Firstly, this means that there must be more design points than parameters in the model. For the linear model, the design must consist of at least $K \geq n^+ + 1$ points, whereas a design to fit a quadratic model in Stage II must have at least $K \geq 1 + 2n^+ + n^+(n^+ - 1)/2$ points. There is a large literature on producing designs for non-standard regions, with an array of optimality measures. One such measure is D-optimality (Montgomery, 2009), which measures the generalised variance of the model parameters:

$$\mathbb{D}(\mathbf{D}_j) = \det((\mathbf{D}_j^T \mathbf{D}_j)^{-1}). \tag{4.3.6}$$

Minimising this quantity, or equivalently maximising $\det(\mathbf{D}_j^T \mathbf{D}_j)$, minimises the generalised variance, and thus the confidence region volume of the model parameters. As the aim of the design is to produce a good gradient estimator, this is an appropriate objective for the design. Maximising $\det(\mathbf{D}_j^T \mathbf{D}_j)$ is a large, highly non-linear and non-convex optimisation problem which is difficult to solve using exact methods. As an alternative, heuristics known as Exchange Algorithms were developed, in which one design point at a time is swapped or moved until $\det(\mathbf{D}_j^T\mathbf{D}_j)$ begins to converge (Mitchell, 1974). That is, each design point is moved to maximise $\det(\mathbf{D}_j^T\mathbf{D}_j)$ given the position of all the other design points in $\mathbf{D}_j$.

For a region of interest that can be considered as the Cartesian product of several sets, Meyer and Nachtsheim (1995) introduced a further simplification, known as the Coordinate-Exchange Algorithm (CEA). In the case of bound constraints, the use of the ℓ_∞ norm to define the trust region produces an n^+ dimensional hyper-box, which is the Cartesian product of n^+ intervals. Then, for each coordinate within the point being exchanged, the optimisation of $\det(\mathbf{D}_j^T\mathbf{D}_j)$ occurs separately. This reduces the task of building a good design to a series of one dimensional optimisation problems. The process is repeated until passing through all of the design points of $\mathbf{D}_j$ produces less than $\varepsilon_D\%$ increase in $\det(\mathbf{D}_j^T\mathbf{D}_j)$. Whilst this heuristic can get caught at poor local optima, the speed of this procedure is important as it must be performed at each iteration of the STRONG algorithm. For more details of the algorithm, see Section 6.3.

As well as the design itself, the number of replications at each design point, N_j^p, must be chosen. For simplicity, this value is uniform across all design points $\mathbf{d}_{jk}$ in the design $\mathbf{D}_j$. Here there is a trade-off between taking N_j^p to be large enough to achieve a good parameter estimation and the computational effort. As this trade-off may be different across the feasible region, we choose the value of N_j^p by considering the signal-to-noise ratio across the trust region. Let $S_j^2(\mathbf{d})$ be the sample variance of the cost observations of solution $\mathbf{d}$ up until iteration j. To approximate the signal-to-noise ratio we compare the difference in mean between the current solution $\mathbf{d}_j$ and the previous solution $\mathbf{d}_{j-1}$ with its estimated standard deviation:

$$\left| \frac{\hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_{j-1}) - \hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_j)}{\sqrt{S^2_{j-1}(\mathbf{d}_{j-1})/N^p_j + S^2_{j-1}(\mathbf{d}_j)/N^p_j}} \right| \geq 2.$$

This is similar to the suggestion made in Equation (16) of Chang et al. (2013) for STRONG. Thus, we select

$$N^p_j = \left\lceil 4 \, \frac{S^2_{j-1}(\mathbf{d}_{j-1}) + S^2_{j-1}(\mathbf{d}_j)}{\left(\hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_{j-1}) - \hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_j)\right)^2} \right\rceil \tag{4.3.7}$$

as long as $N^p_j \in \{N^p_{\text{min}}, ..., N^p_{\text{max}}\}$. The upper limit N^p_{max} is used due to the simulation budget. If the previous iteration failed to produce a new solution, that is the proposed solution from iteration $j-1$, $\mathbf{d}^*_{j-1}$, is rejected, then $\mathbf{d}_j = \mathbf{d}_{j-1}$ and (4.3.7) would be unbounded. To avoid this, we replace $\mathbf{d}_{j-1}$ in (4.3.7) with $\mathbf{d}^*_{j-1}$:

$$N^p_j = \left\lceil 4 \, \frac{S^2_{j-1}(\mathbf{d}^*_{j-1}) + S^2_{j-1}(\mathbf{d}_j)}{\left(\hat{g}_{j-1}(\mathbf{x}, \mathbf{d}^*_{j-1}) - \hat{g}_{j-1}(\mathbf{x}, \mathbf{d}_j)\right)^2} \right\rceil .$$

(ii) Model Estimators

Once the design has been created, N^p_j replications are made at each design point $\mathbf{d}_{jk}$, $k = 1, ..., K$, and all observations are taken to produce the response vector $\mathbf{Y}_j$. STRONG advocates the Ordinary Least Squares (OLS) estimator to estimate the meta-model. In the linear model case, the OLS estimator of the gradient is

$$\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j) = (\mathbf{D}^T_j \mathbf{D}_j)^{-1} \mathbf{D}^T_j \mathbf{Y}_j. \tag{4.3.8}$$

For the quadratic model, the design matrix, $\mathbf{D}_j$, also includes columns for the interaction and quadratic terms. In this case the components of both $\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)$ and $\widehat{H}_j(\mathbf{x}, \mathbf{d}_j)$ are estimated simultaneously using the OLS estimator.

(iii) Generalised Cauchy Step

The meta-model $\hat{r}_j$ is used as the objective function for the sub-problem (4.3.4). Rather than solving the sub-problem exactly, STRONG uses the Cauchy Point as an approximate solution. This minimises $\hat{r}_j$ along the maximum negative gradient direction subject to the trust-region constraints. With the problem constraints (4.3.2), this step is not guaranteed to be feasible, and truncating it will not guarantee a sufficient decrease in $\hat{r}_j$ (convergence of trust-region methods rely on achieving at least a minimum reduction in $\hat{r}_j$). For the bound constrained problem, Conn et al. (1988) propose an alternative, known as the Generalised Cauchy Point. This minimises the meta-model along the *projected-gradient path*.

The projected-gradient path from the current solution $\mathbf{d}_j$ is simply the projection of the path created by travelling in the direction $-\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)$ onto the feasible trust region:

$$\begin{aligned} p(\tau, \mathbf{d}_j) &= \mathcal{P}_{\mathcal{D}\cap\mathcal{B}_j}[\mathbf{d}_j - \tau\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)] \qquad \tau \geq 0 \\ &= \arg\min_{\mathbf{d}\in\mathcal{D}\cap\mathcal{B}_j} ||\mathbf{d} - (\mathbf{d}_j - \tau\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j))||_2 \qquad \tau \geq 0. \end{aligned} \tag{4.3.9}$$

The path $p(\tau, \mathbf{d}_j)$ may have an end, denoted $p(\tau_m, \mathbf{d}_j)$. Then $p(\tau, \mathbf{d}_j) = p(\tau_m, \mathbf{d}_j)$ for all $\tau \geq \tau_m$. Note that, as $\mathcal{D} \cap \mathcal{B}_j$ is always a hyper-box, $p(\tau, \mathbf{d}_j)$ is a sequence of straight lines, with breaks defined by the points at which each variable leaves the feasible trust region. A step along the projected-gradient path is:

$$\mathbf{s}_j(\tau) = p(\tau, \mathbf{d}_j) - \mathbf{d}_j. \tag{4.3.10}$$

The Generalised Cauchy Step is the step $\mathbf{s}_j(\tau^*)$ along the projected-gradient path that minimises the meta-model $\hat{r}_j$ along $p(\tau, \mathbf{d}_j)$, for some $\tau^* \in [0, \tau_m]$. This is a one-dimensional optimisation problem. For ease of notation, we denote the Generalised Cauchy Point as:

$$\mathbf{d}_j^* = \mathbf{d}_j + \mathbf{s}_j(\tau^*) = p(\tau^*, \mathbf{d}_j).$$

In the linear model, where travelling further gives you a greater reduction, $\tau^* = \tau_m$ and $\mathbf{s}_j(\tau^*)$ is simply defined component-wise by

$$\mathbf{s}_j(\tau^*)^f = \begin{cases} \max\{-\Delta_j, -d_j^f\} & \text{if } \widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)^f > 0; \\ \min\{\Delta_j, u^f - d_j^f\} & \text{if } \widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)^f < 0; \\ 0 & \text{if } \widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)^f = 0. \end{cases} \tag{4.3.11}$$

This is the exact solution to the sub-problem (4.3.4). Note that the final case of $\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)^f = 0$ is a probability 0 event in the stochastic problem.

In the quadratic meta-model case, the value of the model along the projected-gradient path, $\hat{r}_j(\mathbf{d}_j + \mathbf{s}_j(\tau))$, is a piecewise quadratic function in one variable, τ. The minimum within the feasible trust region is therefore well defined. The method for finding τ^* is based on Algorithm 17.3.1 of Conn et al. (2000) (page 791), although we do not stop at the first local minima, which may not be the smallest local minimum of the function. Therefore, we search along each straight line in sequence for local minima, by checking the curvature and gradient in each segment, and choose the smallest of these (the reason for this will be clear in Section 6.6). For further details of this algorithm, see Section 6.4.

The Generalised Cauchy Step guarantees a level of decrease in $\hat{r}_j$ which we will denote as ζ_j and will be defined precisely in Lemmas 6.6.2 and 6.6.3. This decrease changes form for the linear and quadratic meta-models and depends on the size of the trust region and the distance from first-order optimality. Guaranteeing this decrease is what guarantees that trust-region methods converge (see, for example, Section 4.2 of Nocedal and Wright (2006)).

4.3.2 Acceptance of the Proposed Step

In STRONG, once a new solution has been proposed, the current solution, $\mathbf{d}_j$, and proposed step to the Generalised Cauchy Point, $\mathbf{d}_j^*$, are simulated N_j^c times. The proposal is only accepted if it passes two tests based on the N_j^c observations. If the signal-to-noise ratio is small, a larger sample size should be used to improve the performance estimates. Furthermore, Chang et al. (2013) suggest that N_j^c should be larger than N_j^p and have a minimum size. Thus, we choose to take $N_j^c = \min\{2N_j^p, N_{\min}^c\}$.

Ratio Comparison Test

In any trust-region optimisation procedure, a proposed solution, $\mathbf{d}_j^*$, is only accepted if the actual improvement obtained is at least a pre-defined positive fraction, $\eta_0 > 0$, of the reduction predicted by the meta-model. The value of η_0 determines the willingness to accept a marginal improvement; a larger η_0 indicates that the user requires more certainty that $\mathbf{d}_j^*$ provides a reduction. It can be chosen based on the values used by Chang et al. (2013), η_0=0.01, or the deterministic approach (Nocedal and Wright (2006) take $\eta_0 \in [0, 1/4)$). This is a measure of how much trust can be put in the model. In STRONG, the following ratio is considered:

$$\rho_j := \frac{\hat{g}_j(\mathbf{x}, \mathbf{d}_j) - \hat{g}_j(\mathbf{x}, \mathbf{d}_j^*)}{\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j^*)}. \tag{4.3.12}$$

We must have $\rho_j > \eta_0$ before we consider accepting the proposed point. This is called the Ratio Comparison (RC) test. In deterministic trust-region approaches, the closer ρ_j is to 1, the better the prediction of $g(\mathbf{x}, \mathbf{d}_j^*)$ is, giving a firm indication that the meta-model provides a good fit. In STRONG, due to sampling error ρ_j is only an estimated quantity. So whilst it does give an indication of model fit, this is subject to uncertainty.

Sufficient Reduction Test

To account for the estimation of the objective function via the sample average of simulation replications, in addition to the Ratio Comparison test, STRONG performs a Sufficient Reduction (SR) hypothesis test. This test is only used if the Ratio Comparison test is passed and looks at whether there is statistical evidence that the solution $\mathbf{d}_j^*$ does give a significant drop in the mean:

$$H_0: \ g(\mathbf{x}, \mathbf{d}_j) - g(\mathbf{x}, \mathbf{d}_j^*) \leq \eta_0^2 \zeta_j \qquad \text{versus} \qquad H_1: \ g(\mathbf{x}, \mathbf{d}_j) - g(\mathbf{x}, \mathbf{d}_j^*) > \eta_0^2 \zeta_j.$$

The quantity ζ_j is linked to the reduction guaranteed by the Generalised Cauchy Point. (The reason why η_0 is squared is in the proof of Lemma 6.6.6 in Section 6.6.3). To test the hypothesis, STRONG uses the Welch statistic (Welch, 1938)

$$T^* = \frac{\hat{g}_j(\mathbf{x}, \mathbf{d}_j) - \hat{g}_j(\mathbf{x}, \mathbf{d}_j^*) - \eta_0^2 \zeta_j}{S_j}, \tag{4.3.13}$$

an approximate solution to the Behrens-Fisher problem of testing for different means of two Normal distributions with unknown and heterogeneous variances. S_j^2 is a pooled variance

$$S_j^2 = \frac{S_j^2(\mathbf{d}_j)}{N_j} + \frac{S_j^2(\mathbf{d}_j^*)}{N_j^c}.$$

where $N_j \geq N_j^c$ is the total number of observations of $\mathbf{d}_j$. Assuming that the simulation output is Normally distributed, T^* approximately follows a $t-$distribution with

$$\hat{\phi} = \left\lceil S_j^4 \left[\frac{(S_j^2(\mathbf{d}_j)/N_j)^2}{N_j - 1} + \frac{(S_j^2(\mathbf{d}_j^*)/N_j^c)^2}{N_j^c - 1} \right]^{-1} \right\rceil$$

degrees of freedom. The significance level of the hypothesis test, α_j, decreases with every iteration. To ensure convergence of STRONG, it must decrease quickly enough to satisfy

$$\sum_{j=1}^{\infty} \alpha_j < \infty.$$

In the computational experiments of Chang et al. (2013) $\alpha_j = \alpha_0 \times 0.98^j$. The null hypothesis is rejected if T^* exceeds the critical value. In this case $\mathbf{d}_{j+1} = \mathbf{d}_j^*$. Otherwise $\mathbf{d}_{j+1} = \mathbf{d}_j$.

Updating the Trust-Region Size

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 $\hat{r}_j$ 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, Δ_j. If either the RC test or SR 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.

If a point proposed by the quadratic model is rejected, STRONG uses a mechanism called the *inner loop* to ensure that a new solution is found (if not currently at a local optimal) without $\Delta_j \to 0$. Details of this are given in the next section.

4.3.3 The Inner Loop

The inner loop is the primary mechanism for STRONG's convergence. If there is an unsuccessful iteration using a quadratic meta-model (either the RC test or the SR test is failed) STRONG begins its inner loop.

At each iteration i of the inner loop the trust-region size, Δ_{j_i}, is updated and new design points are added to the design. All previous points are kept, and the design matrix is augmented. The sample size for both the new design points and for the acceptance tests grow with each iteration:

$$N^d_{j_{i+1}} = \left\lceil \gamma_0^{-4} \sum_{k=1}^{i} N^d_{j_k} \right\rceil \tag{4.3.14}$$

and

$$N^c_{j_{i+1}} = \left\lceil (\gamma_0^{-4} + 1) N^c_{j_i} \right\rceil . \tag{4.3.15}$$

This reduces the sampling error in the SR test, the variance in the OLS gradient estimator and also reduces the bias coming from the quadratic assumption, by effectively giving more weight to design points that are closer to $\mathbf{d}_j$ where the error in the quadratic meta-model is smaller. As Δ_{j_i} decreases towards 0 and the number of observations increase, the OLS estimator will improve, increasing the ability of the algorithm to find a better solution.

The Generalised Cauchy Point for the new model, $\hat{r}_{j_i}$, is found and $N^c_{j_i}$ replications are made of the proposed solution $\mathbf{d}^*_{j_i} = \mathbf{d}_j + \mathbf{s}_{j_i}(\tau^*_i)$ and $\mathbf{d}_j$. The trust-region size, $\Delta_{j_{i+1}}$, is updated just using the value of ρ_{j_i} from Ratio Comparison test.

This process repeats until the proposed solution passes the Ratio Comparison test and the Sufficient Reduction test. At this point, $\mathbf{d}^*_{j_i}$ is accepted as the new solution, the inner loop ends and the trust region returns to its size before the inner loop was initiated, $\Delta_{j+1} = \Delta_j$.

4.3.4 Criticality Measure

STRONG is an algorithm for unconstrained optimisation. Thus, it searches for a local stationary point, where $\nabla_{\mathbf{d}} g(\mathbf{x}, \mathbf{d}) = \mathbf{0}$. Here $||\nabla_{\mathbf{d}} g(\mathbf{x}, \mathbf{d})||_2$ acts as a first-order criticality measure. In constrained optimisation, the notion of optimality must be extended beyond that of a stationary point, as an optimal solution may lie on the boundary of $\mathcal{D}$ without being a stationary point of g. The following quantity was introduced by Conn et al. (1993) as an alternative criticality measure:

$$\chi(\mathbf{d}) := \left| \min_{\mathbf{s}} \left\{ \nabla_{\mathbf{d}} g(\mathbf{x}, \mathbf{d})^T \mathbf{s} : \mathbf{d} + \mathbf{s} \in \mathcal{D}, ||\mathbf{s}||_2 \leq 1 \right\} \right| \qquad \mathbf{d} \in \mathcal{D}.$$

This can be interpreted as the best possible improvement in a linear model of g at $\mathbf{d}$ on a unit ball whilst remaining feasible. $\chi(\mathbf{d})$ will equal 0 when $\mathbf{d}$ is either a stationary point, $||\nabla_{\mathbf{d}} g(\mathbf{x}, \mathbf{d})||_2 = 0$, or on the boundary of $\mathcal{D}$ and the best possible step is $\mathbf{s} = \mathbf{0}$ (the negative gradient is pointing directly out of $\mathcal{D}$). In the unconstrained case $\chi(\mathbf{d})$ reduces to $||\nabla_{\mathbf{d}} g(\mathbf{x}, \mathbf{d})||_2$ as the minimum is achieved by $\mathbf{s} = -\nabla_{\mathbf{d}} g(\mathbf{x}, \mathbf{d})/||\nabla_{\mathbf{d}} g(\mathbf{x}, \mathbf{d})||_2$.

In the simulation optimisation setting, the concept of $\chi(\mathbf{d})$ remains of value, although can only be estimated. At iteration j we therefore propose:

$$\hat{\chi}_j(\mathbf{d}_j) := \left| \min_{\mathbf{s}} \left\{ \widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)^T \mathbf{s} : \mathbf{d}_j + \mathbf{s} \in \mathcal{D}, ||\mathbf{s}||_2 \leq 1 \right\} \right|.$$

For mathematical convenience within the theoretical treatment of the algorithm in Chapter 6, we will work with

$$\hat{\pi}_j(\mathbf{d}_j) := \min\{\hat{\chi}_j(\mathbf{d}_j), 1\}. \tag{4.3.16}$$

For more details on these measures, see Section 6.2.

4.3.5 Comparison with No Delay

The algorithm described in Sections 4.3.1 to 4.3.3 continues until either $\hat{\pi}_j(\mathbf{d}_j)$ is below a tolerance ϵ or the number of simulation calls has exceeded some maximum (an iteration that begins before this limit is allowed to complete) and produces a final solution $(\mathbf{x}, \mathbf{d}^*)$. Due to the discretisation of time in the IP, some delays may have been added that were not necessary. Thus $(\mathbf{x}, \mathbf{d}^*)$ is compared with $(\mathbf{x}, \mathbf{0})$ (using $N^c_{\mathbf{0}}$ observations), that is the choice to not make any planned delays under the aircraft allocation $\mathbf{x}$. As in the SR test, we use a one-sided Welch's t-test to compare:

$$H_0: \quad g(\mathbf{x}, \mathbf{0}) \geq g(\mathbf{x}, \mathbf{d}^*), \qquad H_1: \quad g(\mathbf{x}, \mathbf{0}) < g(\mathbf{x}, \mathbf{d}^*).$$

If $(\mathbf{x}, \mathbf{0})$ is found to be an improvement, it is set as the final solution. Potentially a better approach would be to test individual delays separately. However, this would require a Ranking & Selection procedure to ensure sufficient confidence in selecting the best.

4.3.6 Algorithm Details

The main framework of the proposed algorithm is in Algorithm 4.1. Details of Stage I and Stage II iterations are found in Algorithms 4.2 and 4.3 respectively. The inner loop mechanism is described in Algorithm 4.4.

Once this optimisation has been completed, we are left with a set of improved solutions, each one originating from a solution generated by the IP. The OCC decision makers can then consider each of these options in combination with crew and passenger recovery to produce a recovery plan that can be submitted to the relevant authorities.

4.4 Considerations for Practical Use

This section discusses potential improvements that could be made to the simulation and the simulation optimisation algorithm. The focus is on improved estimation in the simulation as well as computational factors that could increase the speed of the optimisation.

4.4.1 Simulation Model

There are a number of aspects of the simulation model that are not representative of the real processes. Whilst we believe this should not affect the performance of the simulation optimisation algorithm proposed and that many of the key system features

Algorithm 4.1 Main Framework Algorithm

```
1: Set $j = 0$. Select constants $0 \leq \eta_0 < 1/4 \leq \eta_1 < 1$, $0 < \gamma_0 < 1 < \gamma_1$, $\alpha_0 \in (0,1)$,
   $\tilde{\Delta}$, and $N_0^c$. Select stopping criterion, $\epsilon$, for criticality measure. Select initial
   solution, $(\mathbf{x}, \mathbf{d}_0)$ from IP solution, and $\Delta_0 > \tilde{\Delta}$.
2: **repeat**
3:     **if** $\Delta_j > \tilde{\Delta}$ **then**
4:         Go to Algorithm 4.2 for Stage I linear model
5:     **else**
6:         Go to Algorithm 4.3 for Stage II quadratic model
7:     **end if**
8:     $\alpha_{j+1} \leftarrow \alpha_j \times 0.98$
9:     $j \leftarrow j + 1$
10: **until** reached maximum number of simulation replications or $\hat{\pi}_j(\mathbf{d}_j) < \epsilon$
11: Compare $(\mathbf{x}, \mathbf{d}_j)$ with $(\mathbf{x}, \mathbf{0})$ using Welch's t-test.
```

Algorithm 4.2 Stage I - Linear Model

```
1: Build design matrix $\mathbf{D}_j$ using CEA
2: Perform $N_j^p$ observations at each design point
3: Use equation (4.3.8) to estimate $\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)$
4: Find the Generalised Cauchy Step, $\mathbf{s}_j(\tau^*)$, from (4.3.11)
5: Take $N_j^c$ replications of $\mathbf{d}_j$ and $\mathbf{d}_j^*$
6: Evaluate the RC $\rho_j$ using (4.3.12)
7: Conduct the SR Test with Type I error $\alpha_j$ using equation (4.3.13).
8: if $\rho_j < \eta_0$ or SR Test is failed then
9:     $\mathbf{d}_{j+1} \leftarrow \mathbf{d}_j$ and $\Delta_{j+1} \leftarrow \gamma_0 \Delta_j$
10: else if $\eta_0 \leq \rho_j < \eta_1$ and SR Test is passed then
11:     $\mathbf{d}_{j+1} \leftarrow \mathbf{d}_j^*$ and $\Delta_{j+1} \leftarrow \Delta_j$
12: else if $\rho_j \geq \eta_1$ and SR Test is passed then
13:     $\mathbf{d}_{j+1} \leftarrow \mathbf{d}_j^*$ and $\Delta_{j+1} \leftarrow \gamma_1 \Delta_j$
14: end if
15: Return to Algorithm 4.1
```

Algorithm 4.3 Stage II - Quadratic Model

```
1: Build design matrix $\mathbf{D}_j$ using CEA
2: Perform $N_j^p$ observations at each design point
3: Use equation (4.3.8) to estimate $\widehat{\nabla}_{\mathbf{d}} g_j(\mathbf{x}, \mathbf{d}_j)$ and $\widehat{H}_j(\mathbf{x}, \mathbf{d}_j)$
4: Find the Generalised Cauchy Step, $\mathbf{s}_j(\tau^*)$, of the sub-problem using Algorithm
   6.3
5: Take $N_j^c$ replications of $\mathbf{d}_j$ and $\mathbf{d}_j^*$
6: Evaluate the RC $\rho_j$ using (4.3.12).
7: Conduct the SR Test with Type I error $\alpha_j$ using equation (4.3.13).
8: if $\rho_j < \eta_0$ or SR Test is failed then
9:     Go to Algorithm 4.4 for the Inner Loop
10: else if $\eta_0 \leq \rho_j < \eta_1$ and SR Test is passed then
11:     $\mathbf{d}_{j+1} \leftarrow \mathbf{d}_j^*$ and $\Delta_{j+1} \leftarrow \Delta_j$
12: else if $\rho_j \geq \eta_1$ and SR Test is passed then
13:     $\mathbf{d}_{j+1} \leftarrow \mathbf{d}_j^*$ and $\Delta_{j+1} \leftarrow \gamma_1 \Delta_j$
14: end if
15: Return to Algorithm 4.1
```

Algorithm 4.4 Inner Loop

```
1: Set $\hat{g}_{j_1}(\mathbf{x}, \mathbf{d}_j) = \hat{g}_j(\mathbf{x}, \mathbf{d}_j)$, $\Delta_{j_1} = \gamma_0 \Delta_j$, $\mathbf{d}^* = \mathbf{d}_j$, $\mathbf{D}_{j_1} = \mathbf{D}_j$ and $i = 1$.
2: repeat
3:     Augment the design matrix $\mathbf{D}_{j_i}$ using an additional, smaller design with $N^p_{j_i}$ observations, as in Section 4.3.3
4:     Use equation (4.3.8) to estimate $\widehat{\nabla}_{\mathbf{d}} g_{j_i}(\mathbf{x}, \mathbf{d}_j)$ and $\widehat{H}_{j_i}(\mathbf{x}, \mathbf{d}_j)$
5:     Find the Generalised Cauchy Step, $\mathbf{s}_{j_i}(\tau_i^*)$, of the sub-problem
6:     Take $N^c_{j_i}$ replications of $\mathbf{d}_j$ and $\mathbf{d}^*_{j_i}$
7:     Evaluate the RC $\rho_{j_i}$ using (4.3.12)
8:     if $\rho_{j_i} < \eta_0$ then
9:         $\Delta_{j_{i+1}} \leftarrow \gamma_0 \Delta_{j_i}$
10:    else
11:        Conduct the SR Test with Type I error $\alpha_j$ using equation (4.3.13).
12:        if SR is passed then
13:            $\mathbf{d}^* \leftarrow \mathbf{d}^*_{j_i}$
14:        else if $\rho_{j_i} \geq \eta_1$ then
15:            $\Delta_{j_{i+1}} \leftarrow \gamma_1 \Delta_{j_i}$
16:        else
17:            $\Delta_{j_{i+1}} \leftarrow \Delta_{j_i}$
18:        end if
19:    end if
20:    $i \leftarrow i + 1$
21: until $\mathbf{d}^* \neq \mathbf{d}_j$ or maximum simulation replications reached
22: $\mathbf{d}_{j+1} \leftarrow \mathbf{d}^*$, $\Delta_{j+1} \leftarrow \Delta_j$ and return to Algorithm 4.3
```

are captured in the simulation, better performance estimation could be obtained by improving the model. The assumption of two runways at each airport, their independence and fixed service times for all aircraft does not accurately reflect the congestion levels at all airports. We have also assumed a static deviation-from-schedule distribution, whereas this is likely to depend on the time of day. Using real-time data sources such as Flightradar24 AB (2017) to monitor other aircraft may give a better estimate of arrival times to airports. Ideally, an airline would want to model the airport operations in more detail, considering many of the regulations described in ICAO (2016).

Access to specific data would allow a big improvement in the simulation. An airline would have data to improve upon the turn time, repair time and time to failure distributions in the model. Airlines also have access to detailed weather forecasts and models of how these may effect airport congestion and flight durations. An airline wishing to put the method proposed here into practice would have access to this data, allowing a more tailored model and enabling a better estimation of delays and costs associated with a rescheduling option.

4.4.2 Complexity of the Simulation Optimisation

Running the simulation itself is the most time consuming aspect of the simulation optimisation. In particular, the main computational burden arises in performing the replications of design points to fit the meta-models. The size of the linear models is linear in n^+. But the quadratic meta-models contain $2n^+ + n^+(n^+ - 1)/2 + 1$ parameters, and so needs at least this many design points to build the meta-model.

The inner-loop of the algorithm is also very expensive, as the number of replications needed at each design point grows exponentially with each inner-loop. This means that the choice of threshold trust-region size for switching between Stages I and II, $\tilde{\Delta}$, has a large impact on the convergence. If $\tilde{\Delta}$ is too large, the inner loop could be entered often, increasing the time to run an iteration, as well as increasing the bias in the quadratic meta-model. On the other hand, a small $\tilde{\Delta}$ can mean several inadequate linear meta-models are built, using much of the simulation budget, before the necessary quadratic is used. In our experimental results, it was rare for the inner-loop to be used, which could be due to the choice of algorithm parameters, particularly the simulation budget. We therefore have limited experience of the efficacy of the inner loop, which we suggest be investigated in future.

In reality, much of the algorithm could be run in parallel. This is particularly true for the experimental design. Once the design has been specified, each design point could be simulated multiple times on different processors. If this was the case, the primary bottleneck in the algorithm would be in the Coordinate-Exchange Algorithm. In certain situations this could be removed and replaced with a standard design developed offline. Examples of this situation could be when the trust region is entirely feasible or when some constraints are binding and others do not impact the trust region. Using many random starts or the perturbation approach of Palhazi Cuervo et al. (2016) could lead to better designs by reducing the tendency of Coordinate-Exchange Algorithms to converge to local optima.

4.5 Conclusions and Further Work

This chapter presents the simulation optimisation component of the proposed multi-fidelity modelling approach to the Aircraft Recovery Problem. In conjunction with the IP model from Chapter 3, the algorithm allows for high-fidelity simulations to be used to evaluate solutions in a highly complex environment whilst balancing this with the combinatorial constraints and computational issues associated with simulation optimisation. Chapter 5 presents some experimental results from the proposed method.

Any extensions made to the deterministic model, such as the non-linear delay costs and changes in cruise speed described in Section 3.5.3, could easily be incorporated into the simulation model.

The simulation optimisation algorithm presented involves an extension of the trust-region based method, STRONG, to enable it to handle bound constrained problems. This extension uses the projected-gradient step to allow the proposed solution to always remain feasible. This has necessitated using a different optimality measure. We have also incorporated methods for building experimental designs into the simulation optimisation algorithm. This is a step towards addressing more general constrained simulation optimisation problems. Both projected-gradient steps and Coordinate-Exchange Algorithms have potential to improve the way algorithms deal with deterministic convex constraints within simulation optimisation.

Part of the nature of the ARP is that it is a reoccurring problem for airlines. The use of symbiotic simulation as discussed by Aydt et al. (2009), in which the model is allowed to adapt to new information from the system it is modelling, could have great value in the area of airlines operation. However, this is likely to mostly include changes in the initial conditions and system state, rather than adaptations of the processes and distributions themselves. Future work could include considering how the repeated nature of the problem could be exploited to improve both the models and the optimisation process in a symbiotic manner. Symbiotic simulation could also be used to give new information on the same problem, allowing the disruption to evolve in time. For example, if there were changes to the maintenance status of aircraft, this could potentially be used to improve the decisions based on the latest information. This has not been considered within this thesis.

Chapter 5

Computational Results for the Multi-Fidelity ARP Approach

5.1 Introduction

This chapter investigates the performance of the multi-fidelity solution method proposed in Chapters 3 and 4. Three example problems of varying size are described in Section 5.2. Section 5.3 demonstrates how the system could work in practice, using Problem 1. As the simulation optimisation results depend on the initial seed of the algorithm, the algorithm's behaviour cannot be evaluated based on a single sample path. Therefore, Section 5.4 discusses more rigorous empirical evaluation on all three problems by performing macro-replications of the optimisation and evaluating the solutions found. A small study of the effect of some parameters is presented in Section 5.5. Section 5.6 briefly presents results from other problems tackled, followed by conclusions in Section 5.7.

Some parameters remain fixed across all experiments, unless otherwise stated. These values are either influenced by the airline system or the values used in the evaluation of STRONG in Chang et al. (2013) and are given in Table 5.1.1.

5.2 Problem Descriptions

The following examples are based on flight schedules extracted from the open source dataset Flightradar24 AB (2017) obtained using the Python package pyflightdata (Allamraju, 2017). These three problems were chosen to test the algorithm on different but common types of disruption and different sized fleets. The original schedules and other input data for these three problems (and those described in Section 5.6) can be found in Rhodes-Leader (2020) and are also available on request from the author.

5.2.1 Problem 1: Small Fleet with a Single Aircraft Grounded

The first problem 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 are meant to spend the night. At the beginning of the day, it is discovered that one aircraft at BHX, $A1$, will not be fit to fly its first flight. The expected ready time is 3 hours after that, at 8:50. The OCC must now reschedule the day of operations, involving 54 short-haul flights using the available aircraft. Its options include delaying flights, cancelling flights and exchanging aircraft.

Table 5.1.1: Parameter values across all experiments.

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
Maximum observations in simulation optimisation	$N_{\max}$	2000
Minimum replications at each design point	$N_{\min}^p$	5
Minimum replications for acceptance tests	$N_{\min}^c$	20
Replications for comparison with zero delay option	$N_{\mathbf{0}}^c$	50
Initial trust-region radius (minutes)	Δ_0	15
Threshold trust-region radius (minutes)	$\tilde{\Delta}$	10
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%

5.2.2 Problem 2: Large Fleet with a Single Aircraft Grounded

The second problem considers a fleet of 125 A319 aircraft over 82 western European airports. There are several hub airports. A typical day consists of over 700 flights. An aircraft, $A2$, lands at 10am at Amsterdam Airport Schiphol (AMS) from its hub Luton Airport (LTN), and a fault is discovered which will take several hours to fix. 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 503 flights left in the operating day, which the OCC must now reschedule.

5.2.3 Problem 3: Large Fleet with Multiple Aircraft Grounded

The third problem consists of the same fleet as the second problem. This time, 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 now 564 flights left in the day.

5.3 Demonstration

To demonstrate the algorithm, it is applied to Problem 1 and the results are compared with the 'No Action' option. 'No Action' does not make any changes to the aircraft allocation, the airline simply waits for $A1$ to be repaired and become available to fly. Whilst this is a very unlikely option to take, it provides a benchmark for comparison as a baseline option. The minimum time between flights allowed in the original schedule is 30 minutes, though this includes some slack. After some experimentation, a more optimistic turn time of $t_{\min} = 25$ minutes was selected for the IP, attempting to exploit the slack built into the system. A time discretisation of $m = 5$ minutes and a maximum delay of $M = 180$ minutes are used. This set of parameters implies 1975 flight delay arcs and 1435 ground arcs. The IP is set up and solved using the algorithm described in Section 3.4, with a maximum of 600 seconds used by the Gurobi Solver, with 8 processors available. This process finds five solutions, with different aircraft allocations and varying amounts of delay. Between four and six flights are delayed (so the simulation optimisation will have $n^{+}=$ 4, 5 or 6 variables) and no flights are cancelled. The IP performance measures for each of the solutions are shown in Table 5.3.1. The first five solutions are found quickly, each within 30 seconds. The next set of constraints chosen leads to an infeasible problem, and the next is not solved within the time limit. In solution 5, one flight is delayed beyond 2 hours and so includes some passenger compensation. No other flights require compensation meaning that the cost and delay objectives are closely correlated for all of these solutions.

Figure 5.3.1 shows the results of each stage of the process, using two of the objectives, cost and tail number exchanges. Each solution has a corresponding colour. The triangles represent the solutions found by the IP and evaluated within the IP. This does not account for any stochastic elements of the system. The Pareto frontier between cost and reducing the number of schedule alterations is clear, as expected.

Table 5.3.1: IP solutions for Problem 1. Maximum allowable runtime was 600 seconds.

Solution	Cost	Tail Number	Delay	Cancellations	Solution Time
	(€1000)	Exchanges	(minutes)		(seconds)
1	15.00	30	300	0	4.23
2	15.00	15	300	0	15.67
3	15.75	12	315	0	12.61
4	16.50	8	330	0	25.66
5	25.25	4	495	0	18.71

These solutions are evaluated using the simulation to account for the uncertainty. Each solution was simulated 1000 times using Common Random Numbers (CRN). The mean is represented by the squares, with the interval going up to the empirical 0.95 quantile. The IP clearly underestimates the cost, which is not surprising as it removes stochasticity from the system. However, the ordering is very similar, helping to justify a multi-fidelity modelling approach.

Each of the solutions found by the IP is used as a starting point for the simulation optimisation process. Each starting point is simulated $N^c_{\min}$ times to provide an initial estimate of its mean and variance, before the algorithm begins. The linear models are built using $n^+ + \lfloor n^+/2 \rfloor + 1$ design points and the quadratic models use $2n^+ + \lfloor n^+/2 \rfloor + n^+(n^+ - 1)/2 + 1$ design points. The circles in Figure 5.3.1 show the means of the resulting solutions, with the interval again going up to the empirical 0.95 quantile. They are shifted horizontally for visual clarity, the aircraft allocations

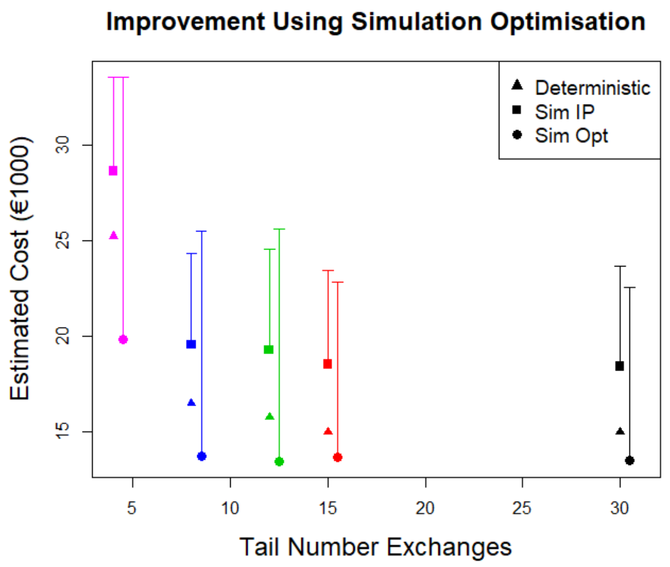

Figure 5.3.1: Aircraft exchanges against cost estimates under different models for each solution. Interval is from mean to the 0.95 quantile.

remain the same. We see a substantial decrease in the mean as hoped for, though an increase in standard deviation also emerges.

Table 5.3.2 provides more detail on the performance of the solutions before and after the simulation optimisation. This confirms the decrease in mean, but increase in standard deviation. Furthermore, there is an improvement in the 0.9 quantile suggesting that the solutions are robust, though the tails appear to be longer, and the 0.95 quantiles are not consistently improved. The final column shows an estimate of the probability that the solution $(\mathbf{x}, \mathbf{d}^*)$ will lead to a lower cost than $(\mathbf{x}, \mathbf{d}_0)$, $P_I(\mathbf{d}^*)$. As we have used CRN, the replications can be paired, as the same scenario is faced by both solutions. Let $G_i(\mathbf{x}, \mathbf{d}_0)$ and $G_i(\mathbf{x}, \mathbf{d}^*)$ be the cost observation of the solutions $(\mathbf{x}, \mathbf{d}_0)$ and $(\mathbf{x}, \mathbf{d}^*)$ using the i^{th} set of CRN, respectively. Then we can estimate $P_I(\mathbf{d}^*)$

Table 5.3.2: The estimated means, $\hat{g}(\mathbf{x}, \mathbf{d})$, with 95% confidence interval halfwidths, standard deviations, $\hat{\sigma}$, 0.9 and 0.95 quantiles, $\hat{q}_{0.9}$ and $\hat{q}_{0.95}$, and the probability of improved cost $\hat{P}_I(\mathbf{d}^*)$, for the cost (€1000) performance for the solutions.

Solution	Initial Solution				Improved Solution				
	$\hat{g}(\mathbf{x}, \mathbf{d}_0)$	$\hat{\sigma}$	$\hat{q}_{0.9}$	$\hat{q}_{0.95}$	$\hat{g}(\mathbf{x}, \mathbf{d}^*)$	$\hat{\sigma}$	$\hat{q}_{0.9}$	$\hat{q}_{0.95}$	$\hat{P}_I(\mathbf{d}^*)$
No Action	37.3 (0.97)	15.7	58.9	66.0					
1	18.4 (0.18)	2.89	21.0	23.7	13.5 (0.29)	4.60	19.0	22.6	0.949
2	18.5 (0.26)	4.16	21.0	23.5	13.7 (0.33)	5.40	19.2	22.9	0.940
3	19.3 (0.19)	3.12	22.1	24.6	13.4 (0.38)	6.07	21.9	25.6	0.874
4	19.6 (0.19)	3.05	22.1	24.4	13.7 (0.36)	5.87	21.8	25.5	0.887
5	28.7 (0.18)	2.96	31.1	33.6	19.8 (0.43)	6.86	29.2	33.6	0.919

by

$$\hat{P}_I(\mathbf{d}^*) = \frac{1}{1000} \sum_{i=1}^{1000} \mathbb{I}\left(G_i(\mathbf{x}, \mathbf{d}^*) < G_i(\mathbf{x}, \mathbf{d}_0)\right), \tag{5.3.1}$$

where $\mathbb{I}(\cdot)$ is the indicator function. The results suggest that there is improvement in over 85% of the replications, which adds evidence that the solutions keep some level of robustness but with a longer tail than the initial solutions (this will be discussed further in Section 5.4.1).

The Empirical Cumulative Distribution Functions (ECDFs) of the initial and resulting solutions based on the 1000 CRN replications are shown in Figure 5.3.2. The IP solutions appear quite robust across the 1000 replications, as shown by the initial steep gradients. This suggests that the planned delays leave enough slack in the

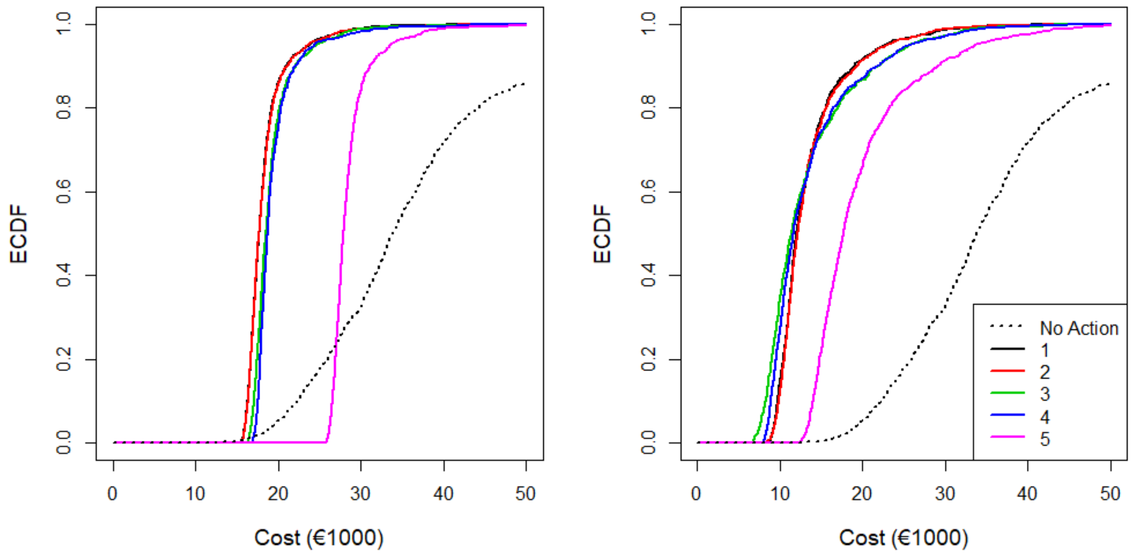

Figure 5.3.2: ECDFs for Problem 1, based on 1000 CRN replications. Left plot shows solutions from the IP, the right shows solutions from the simulation optimisation. Both are compared to taking no action at all. Colours match those in Figure 5.3.1.

new schedule to absorb most of the variability in turn times and repair time without much extra cost. This slack also prevents earlier take-off when possible, which the simulation optimisation appears to exploit. An ideal set of results for the simulation optimisation would see each ECDF moving to the left (reducing the mean) and having a steeper gradient (reducing the standard deviation) when compared to the ECDF of the corresponding IP solution. The results suggest that the simulation optimisation process improves the mean performance. However, increases in standard deviation are also seen. All solutions show an improvement in mean and standard deviation over the 'No Action' response, showing that improvements can be made using this solution method.

As the solutions faced the same 1000 scenarios (via CRN), the distributions of $(G_i(\mathbf{x}, \mathbf{d}^*) - G_i(\mathbf{x}, \mathbf{d}_0))$ can be estimated and are shown in Figure 5.3.3 for each plan.

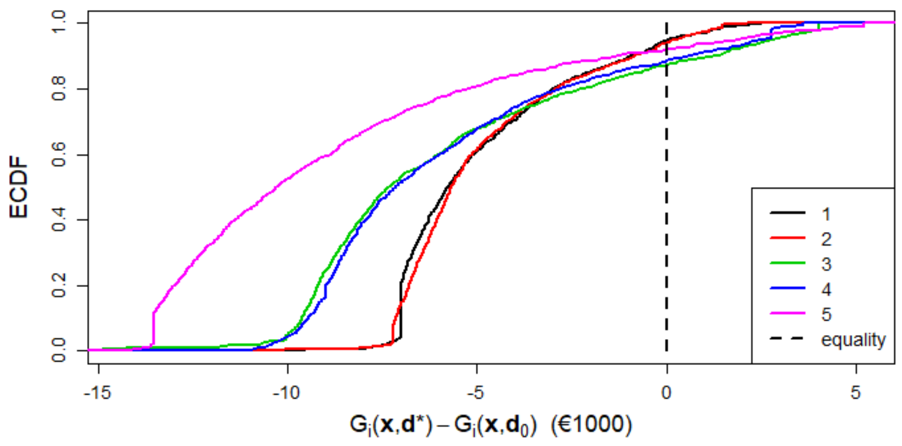

Figure 5.3.3: ECDFs of the improvement over the IP solution for Problem 1, based on 1000 CRN replications. Colours match those in Figure 5.3.1.

This gives a more comprehensive view than the value of $\hat{P}_I(\mathbf{d}^*)$ (which is the value of the ECDF at 0). This emphasises the improvement in Plan 5 particularly. However, there is still a probability that the solution could perform worse than the original IP solution, though the tail of the distribution is not too long.

Tables 5.3.1 and 5.3.2 show that different solutions offer different characteristics. Solution 3 has the lowest mean but is quite variable. However, solution 1 has the lowest standard deviation and the lowest 0.95 quantile but requires the largest number of changes to the schedule. Solution 5 offers far fewer schedule changes but is the most variable. Different airlines may have differing priorities on this trade-off, and will also take other considerations into account, such as the impact on crew. The use of a symbiotic simulation in this setting would allow this trade-off to be adapted for different decisions to reflect individual circumstances.

5.4 Repeated Experiment Results

Whilst the solution method for the IP is deterministic, the simulation optimisation is based on the random output of the simulation. Therefore, the final solution obtained and implemented by the airline is determined by the random number generator seed used at the beginning of the algorithm. To account for this in the evaluation of the proposed methodology, macro-replications of the whole process have been performed.

One method used to evaluate STRONG, and other simulation optimisation algorithms, is to test them on analytical functions with added noise, then look at the average optimality gap across macro-replications, i.e., a relative distance from the true optimal value of a problem (Chang et al., 2013). This requires both being able to evaluate the objective function exactly and knowing what the optimal solution is, which is not possible in many real simulation optimisation problems. Firstly, there is no way to guarantee finding the exact optimal solution due to the many challenges of simulation optimisation. For this reason, rather than focussing on the distance from optimality, most of the comparisons reported in this chapter are with either taking no action or the starting solution given by the IP. Thus, the focus is on the possible improvement made using the simulation optimisation approach. Secondly, we can only estimate the performance of any solution via simulation. Each solution resulting from the simulation optimisation is simulated 1000 times using CRN so that, as much as possible, each solution has faced the same conditions when we do the comparison. With each aircraft given its own random number stream, each aircraft's individual circumstances should be well aligned. This achieved correlations across solutions of over 0.8, and normally over 0.95 and removes a substantial part of the uncertainty involved.

In addition we go beyond just reporting the 'expected' value of cost (as in the optimality gap) as the 'expected' cost will almost certainly not occur in reality. The real circumstances under which the schedule is operating may be better or worse than predicted. As this is a one-time-only decision, it is important to see the full range of possible outcomes and demonstrate that our solutions are robust to operating under different revelations of the future, rather than focussing on the mean. For a traffic signalling problem, Osorio and Bierlaire (2013) present results using ECDFs of the key performance measure, comparing them to the ECDF of the starting point (as we have done in Figure 5.3.2). When dealing with 10 distinct solutions from 10 macro-replications, the authors combine all solutions into one ECDF. For larger numbers of macro-replications, this can hide individual results, whilst plotting each solution's ECDF as separate lines on the same plot may be unclear. Instead, we summarise the cost distribution for each solution by reporting histograms of the mean and 0.95 quantile from each solution, based on the CRN replications.

Each of the next three subsections (5.4.1 to 5.4.3) refers to one of the problems introduced in Section 5.2. For each, the IP has been solved in the manner described in Section 3.4 to obtain a set of solutions with different priorities over the objectives. For each solution in each set, either 50 or 100 macro-replications of the simulation optimisation described in Section 4.3 starting at that IP solution have been run, each with a different starting seed for the random number generator to produce a (possibly unique) solution. So for each IP solution, we obtain a set of possible 'improved solutions' that could have resulted from a single run. Each of these has been simulated 1000 times to obtain an empirical distribution of how that solution could perform. We present the histograms of the mean and 0.95 quantile across each of the solutions, and compare this to the 'No Action' solution and the starting solution from the IP.

5.4.1 Problem 1

The results from the IP are discussed in Section 5.3, and are shown in Table 5.3.1. One hundred macro-replications of the simulation optimisation process were performed for each of the starting solutions in Table 5.3.1. Each of the resulting solutions were simulated 1000 times using CRN.

For each of the five starting solutions, Figures 5.4.1 and 5.4.2 show histograms of the means and 0.95 quantiles of the empirical distributions of cost across the set of solutions obtained when using $N^p_{\max} = 50$. The plots show that all of the solutions obtained have a substantially smaller mean cost than the corresponding IP solution, suggesting that the algorithm is achieving its aim. However, the same is not true for the 0.95 quantile of cost, as observed in Figure 5.4.2. Here the simulation optimisation solutions only show improvements for 77% of the cases at best (Plan 1), and in 0% of the cases at worst (Plan 4). Our tentative explanation behind this is that the simulation optimisation tends to try to decrease the planned delay, d^f, as this leads to a chance of an earlier departure time if the aircraft is ready. However, in doing so, two increases in variability occur. The first is that the actual observed delay, D^f, becomes more likely to exceed d^f. This on its own adds to the variance of the cost. This is compounded by a second factor, which is the penalty term $c_p(D^f - d^f)^+$ in

Equation (4.2.3). This part of the delay is therefore scaled by an increased value, $c_d + c_p$, which also increases the variance of the cost.

The negative relationship between the mean and the 0.95 quantile is demonstrated in Figure 5.4.3, which plots each solution's mean against its 0.95 quantile.

5.4.2 Problem 2

The IP assumes that aircraft $A2$ will be ready by 16:10. The whole problem, with a maximum allowable delay of $M = 600$ minutes, and an interval size of $m = 30$ minutes, has 7003 flight delay arcs, 5605 ground arcs, 1,402,411 constraints and 1,576,503 variables. This took 782.25 seconds to solve for the initial solution, when 20 processors were available to the Gurobi Solver. This had a cost of €10,500, a delay of 210 minutes and 30 tail number exchanges. As is described later, a finer resolution leads to better solutions, but this would increase the problem size considerably. For this reason, we considered the problem reduction heuristic mentioned in Section 3.5.2. This reduces the fleet size to 48 aircraft and the flight programme to 198 flights, allowing the integer programming problem to be solved relatively quickly.

Using a maximum allowable delay of $M = 600$ minutes and an interval size $m = 15$ minutes the reduced problem has 5448 flight delay arcs and 3913 ground arcs. The minimum turn time used in the IP is $t_{\text{min}} = 20$ minutes. This is the smallest turn time that appears in the schedule and so does not account for refuelling at hub airports. The solutions from a 900 second run with 20 processors available are shown in Table 5.4.1. It is clear from the table that it takes a number of iterations before the first Pareto optimal solution is found, given by solution 8. Before this, the delay and cost

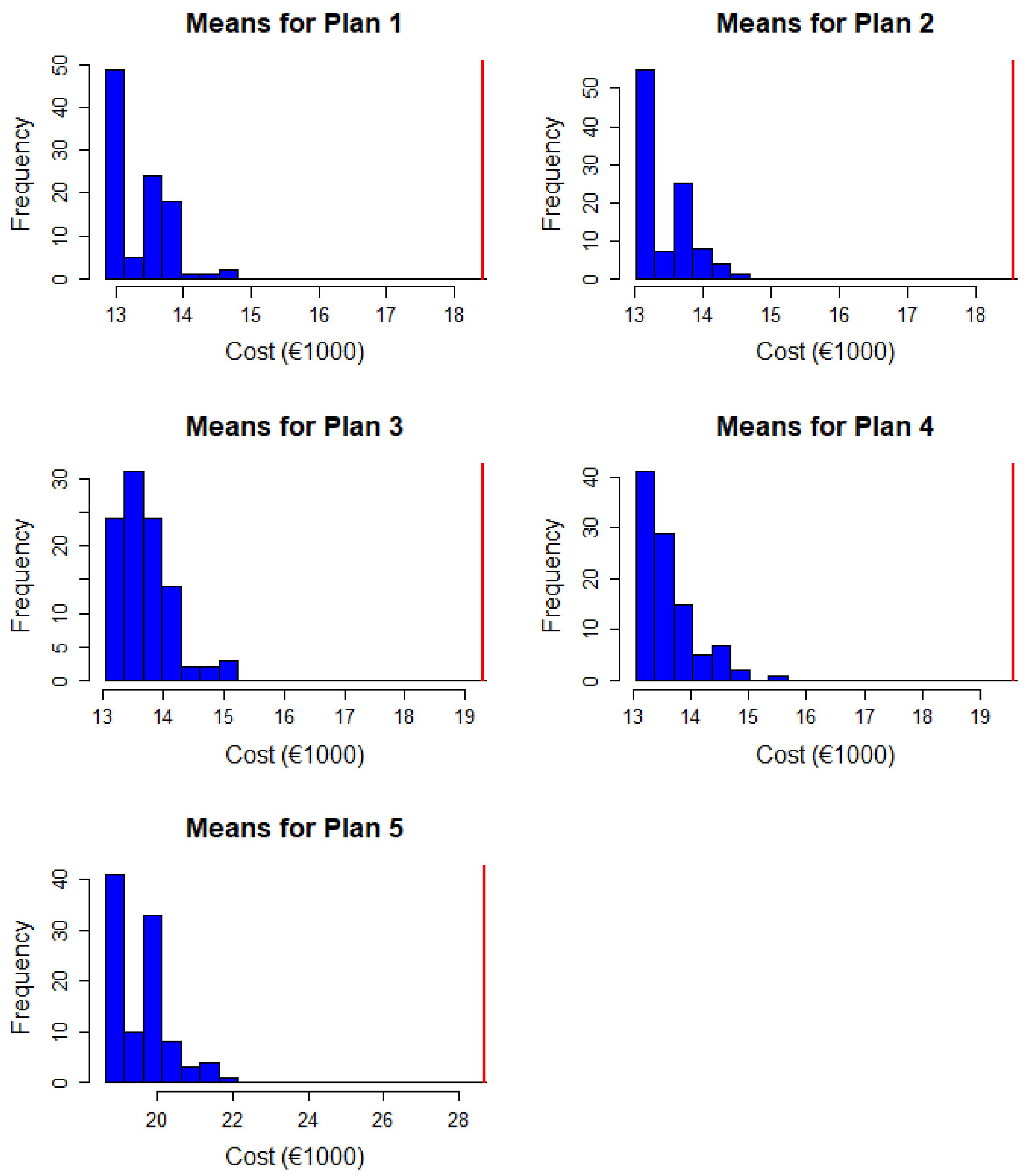

Figure 5.4.1: Problem 1 histograms of the mean of each solution from the simulation optimisation, using $N^p_{\max} = 50$. Each plot corresponds to one of the IP solutions. The red line is the mean of the starting solution from the IP.

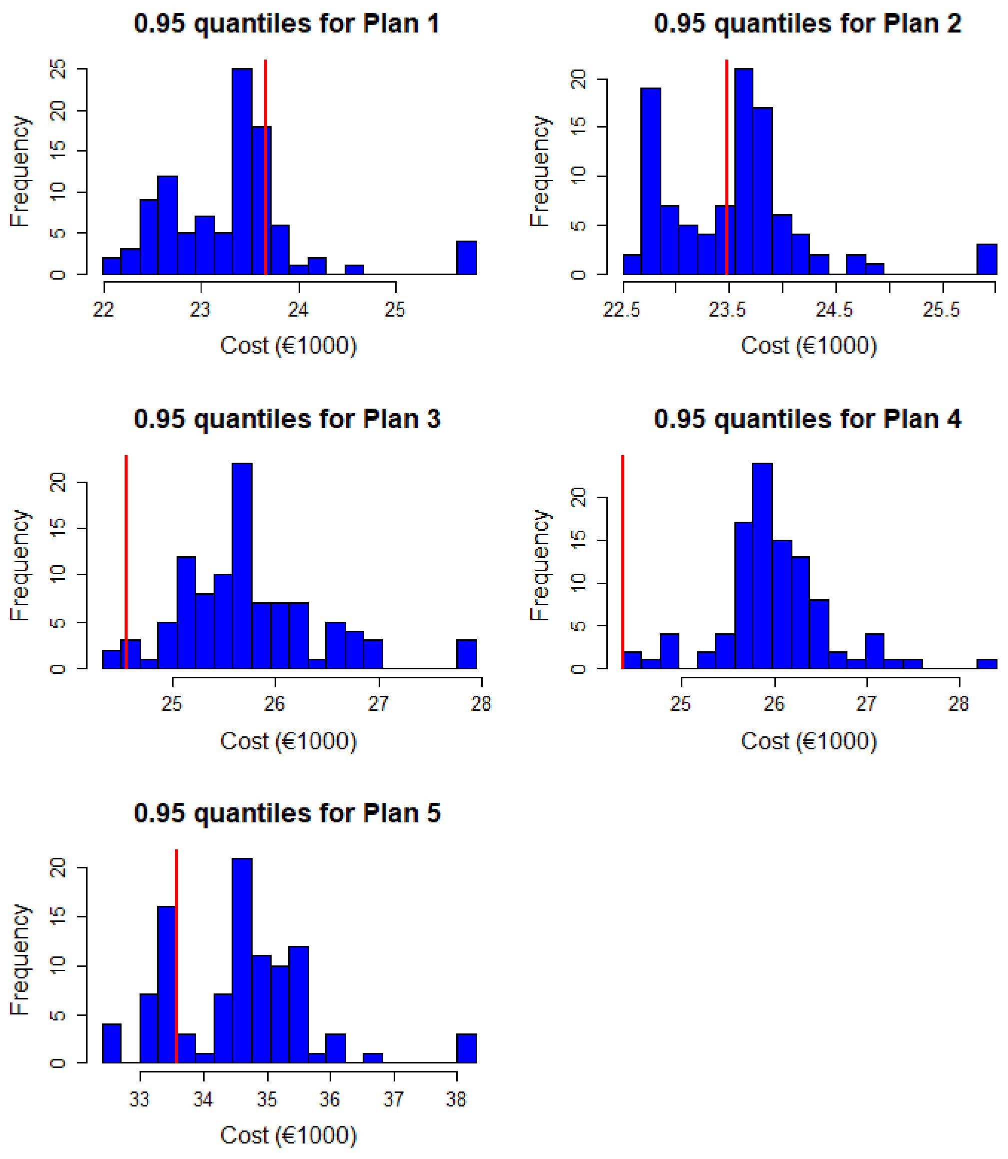

Figure 5.4.2: Problem 1 histograms of the 0.95 quantile of each solution from the simulation optimisation, using $N^p_{\max} = 50$. Each plot corresponds to one of the IP solutions. The red line is the mean of the starting solution from the IP.

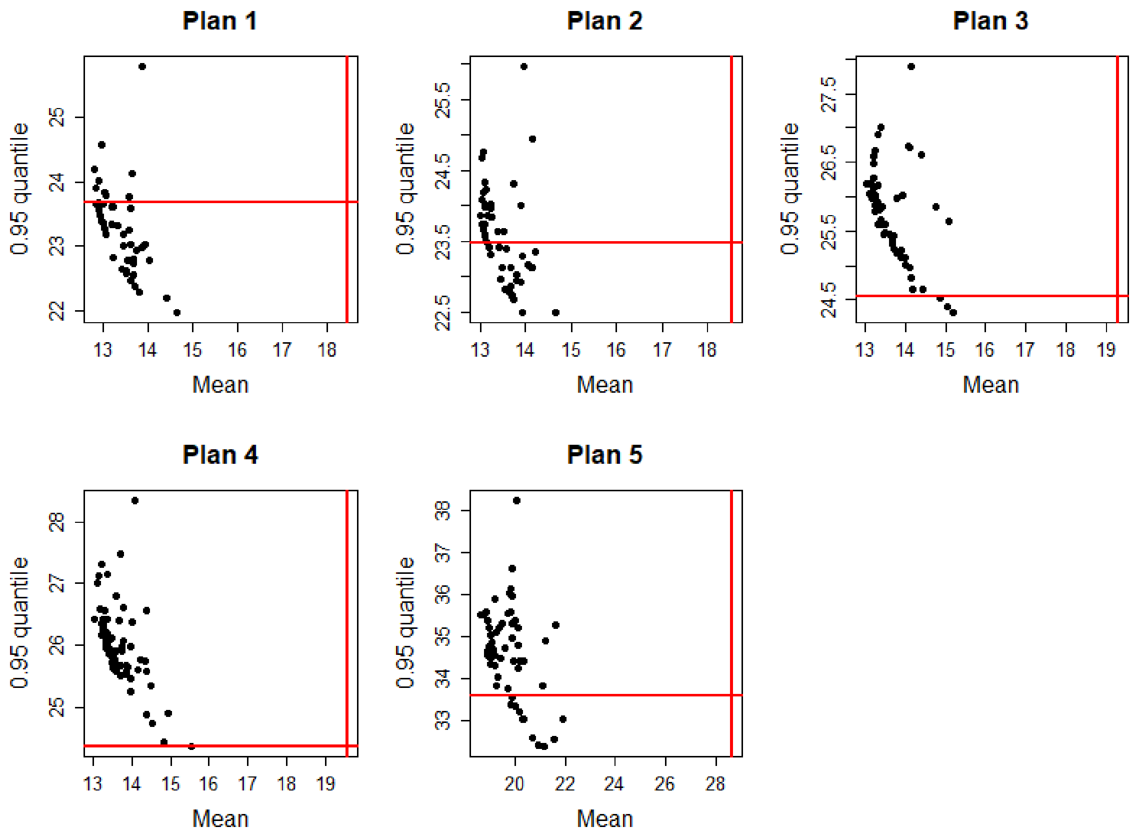

Figure 5.4.3: Mean against 0.95 quantile of solutions found, using $N^p_{\max} = 50$, for each plan from the IP in Problem 1. Red lines indicate the performance of the starting solution from the IP.

objectives remain unchanged from the initial solution whilst the algorithm tries to reduce the number of tail number exchanges by returning more undisrupted aircraft to their original flights. This is a common theme in larger problems, so further work is required to find the Pareto frontier sooner.

The simulation takes into account that refuelling an aircraft leads to higher turn times between some flights. The grounded aircraft's repair time comes from a $\Gamma(111, 3)$ distribution, so that the probability of finishing this by 15:50 is approximately 0.7, giving 20 minutes to be ready for the IP assumed ready time. Turn time distributions are built around the typical times quoted in Airbus (2019).

The simulation optimisation process, starting with solutions 8 to 13, was repeated 50 times, each with $N^p_{\max} = 50$. Each macro-replication of the algorithm could produce a different final solution (due to the dependence of the algorithm on the starting seed). As in 5.4.1, these are then simulated 1000 times using CRN. Histograms of the means and 0.95 quantiles for each solution are shown in Figures 5.4.4 and 5.4.5 respectively. The results show that a significant decrease in the mean is achieved in all macro-replications. However, Figure 5.4.5 gives a different picture regarding the robustness, which is arguably more important. As the simulation optimisation tends to decrease some delays, increasing the probability of exceeding the planned delay, the results become more variable due to the penalty. The results from Plans 9 and 10 are the worst, with only 78% improving on the 0.95 quantile of the IP based solution, whilst the results for plan 12 are all positive. This is due to one flight initially being delayed 2 hours by the IP, leading to some passenger compensation (€1310). The optimisation quickly identifies this as a way to achieve large gains with small effort.

Table 5.4.1: Solutions from the IP for Problem 2 using the heuristic for problem size reduction. Maximum allowable runtime was 900 seconds.

Solution	Cost (€1000)	Tail Number Exchanges	Delay (minutes)	Cancellations	Solution Time (seconds)
1	6.00	32	120	0	56.56
2	6.00	28	120	0	50.24
3	6.00	27	120	0	47.56
4	6.00	24	120	0	60.37
5	6.00	22	120	0	51.77
6	6.00	21	120	0	55.68
7	6.00	20	120	0	57.46
8	6.00	19	120	0	61.87
9	6.25	17	135	0	65.06
10	9.75	15	195	0	90.28
11	9.75	14	195	0	62.00
12	11.06	13	195	0	76.96
13	12.75	12	255	0	104.75

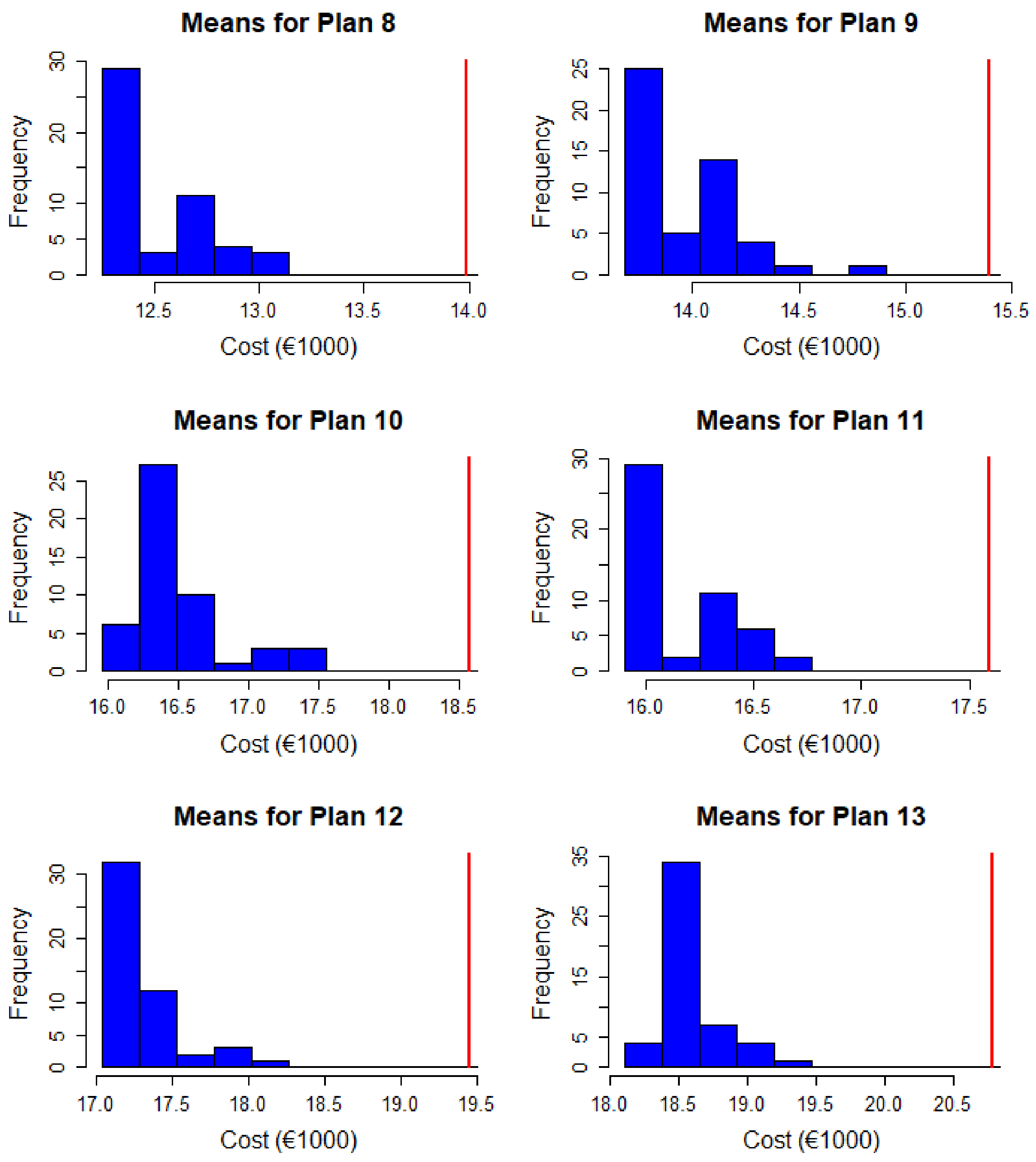

Figure 5.4.4: Problem 2 histograms of the mean of each solution from the simulation optimisation. Each plot corresponds to one of the IP solutions. The red line is the mean of the starting solution from the IP.

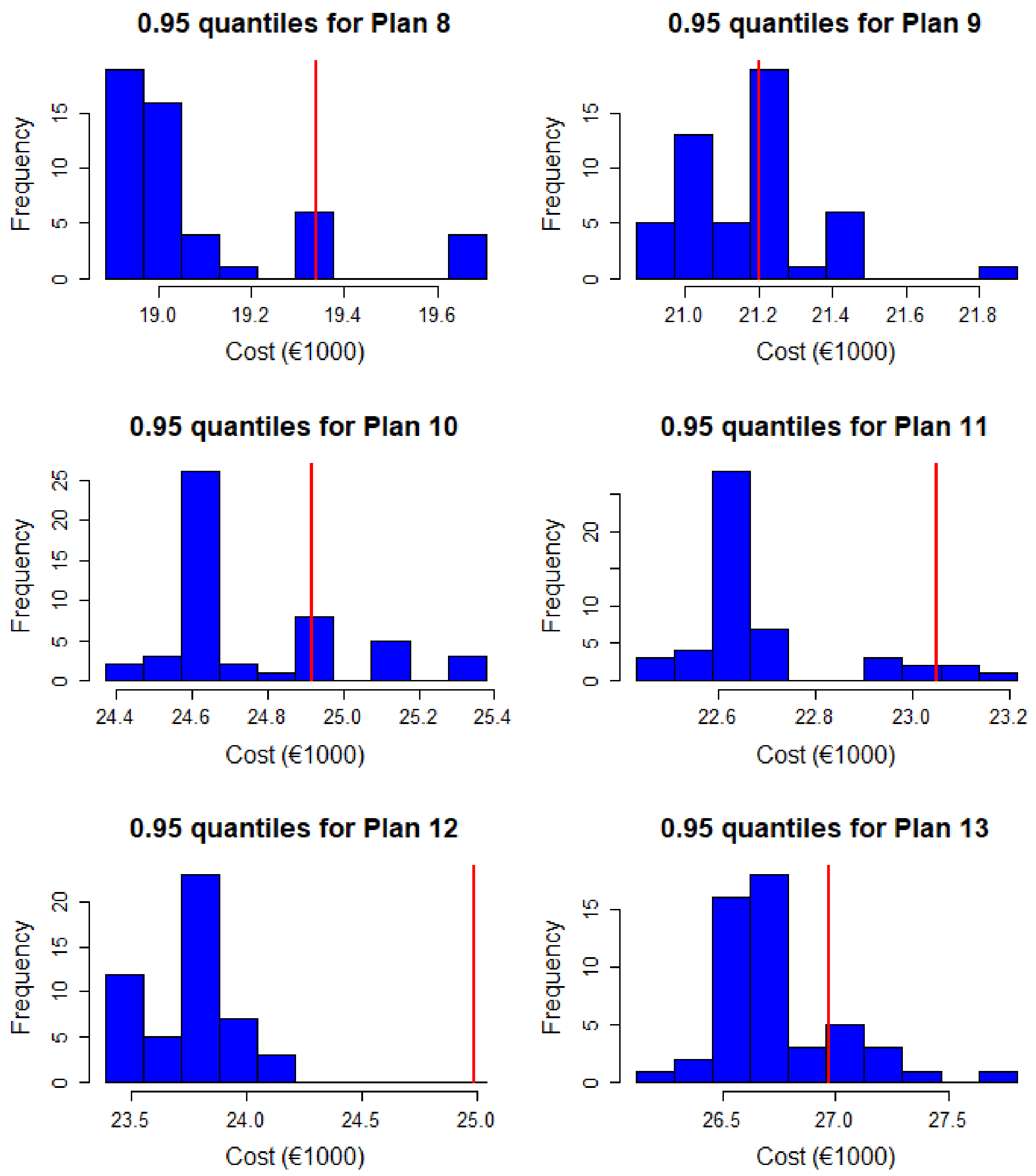

Figure 5.4.5: Problem 2 histograms of the 0.95 quantile of each solution from the simulation optimisation. Each plot corresponds to one of the IP solutions. The red line is the mean of the starting solution from the IP.

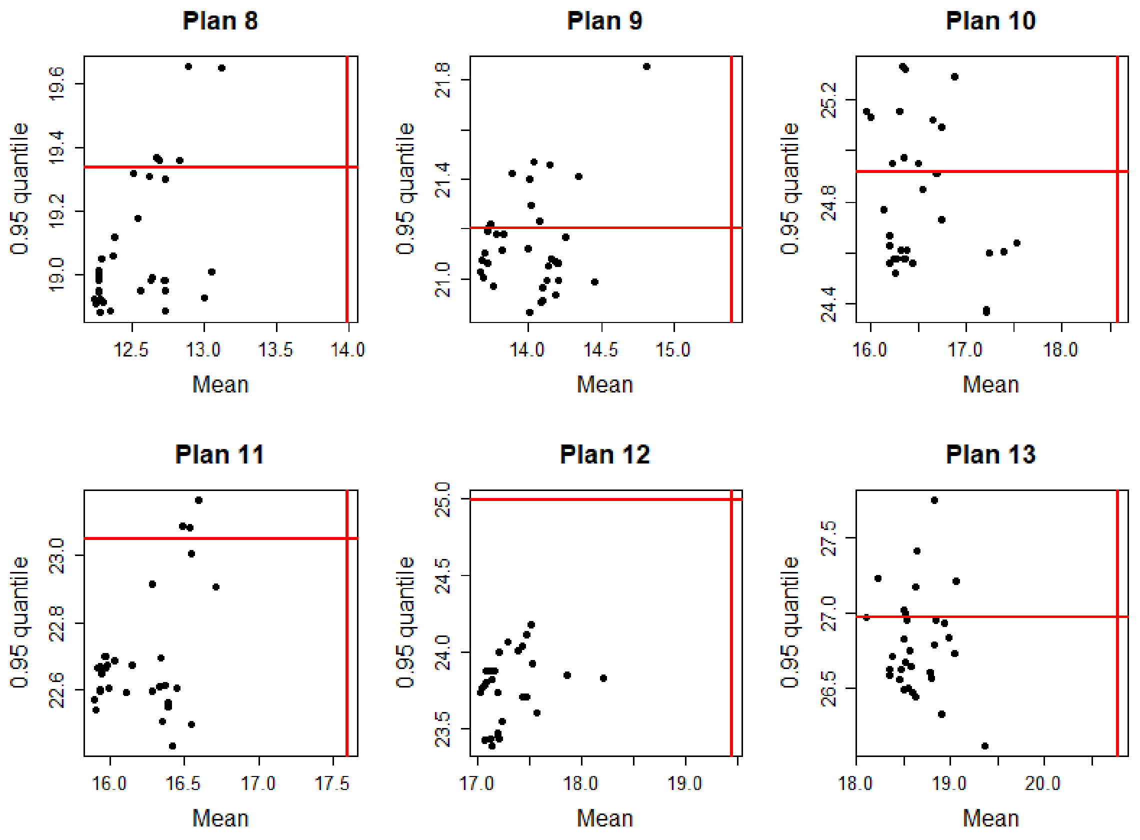

Figure 5.4.6: Mean against 0.95 quantile of solutions found for each plan from the IP in Problem 2. Red lines indicate the performance of the starting solution from the IP.

Interestingly, Figure 5.4.6, which plots the mean against the 0.95 quantile, does not suggest the same behaviour as in Problem 1.

The variances of all solutions are higher than in Problem 1. Under closer scrutiny of the simulated solutions, it appears that flights not associated with the disruption (flights that the IP leaves unchanged) add to the variability, not with large delays but with many small delays. This variation reflects the original schedule's robustness rather than the proposed solution's handling of the disruption. The increased noise makes the situation harder for the optimisation algorithm. Removing unaffected flights from the simulation could both reduce the simulation computation time and improve the results. This was not fully investigated, but may lead to better solutions.

In Section 5.3 (Figure 5.3.3), we showed the ECDF of the difference in cost for replications sharing the same CRN between the IP solutions $(\mathbf{x}, \mathbf{d}_0)$ and the final solution $(\mathbf{x}, \mathbf{d}^*)$ for Problem 1. Figure 5.4.7 shows this for each of the 50 solutions found for each plan for Problem 2. On the whole, the performance is very similar to Problem 1, though the large improvements in cost are not seen, which is reflective of the improvement as a whole being smaller. The estimated probability of improvement, $\hat{P}_I(\mathbf{d}^*)$, (the value on the ECDF at 0) tend to cluster between 0.8 and 0.9, with the exception of Plan 12, which are between 0.9 and 1. This behaviour is shared across the ECDFs, even though the body of the distribution can look quite different. Whilst the probability of being worse is again non-negligible, the upper tails are not too long. In particular, if we consider the conditional expectation given that $(\mathbf{x}, \mathbf{d}^*)$ performs worse:

$$\mathbb{E}\left[G_i(\mathbf{x}, \mathbf{d}^*) - G_i(\mathbf{x}, \mathbf{d}_0) | G_i(\mathbf{x}, \mathbf{d}^*) - G_i(\mathbf{x}, \mathbf{d}_0) > 0\right], \tag{5.4.1}$$

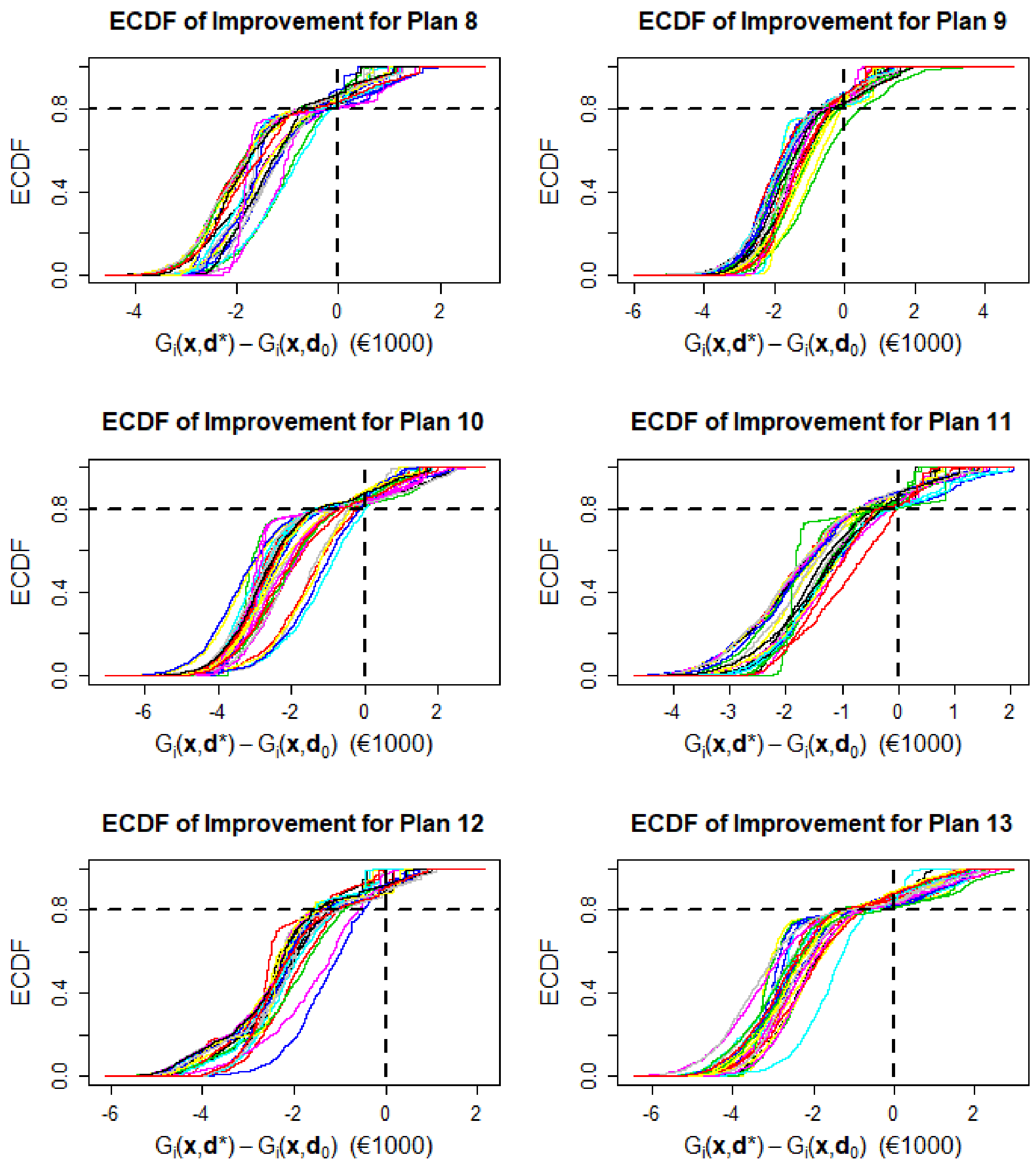

Figure 5.4.7: ECDFs of the cost difference between the solution $(\mathbf{x}, \mathbf{d}^*)$ and $(\mathbf{x}, \mathbf{d}_0)$, for each solution found. Each line corresponds to an individual solution.

the maximum values for each plan are only €980, €1080, €1474, €1029, €614 and €1627, indicating that the losses made are not that large in comparison to the potential improvements.

5.4.3 Problem 3

As the airline is unlikely to be able to control any delay times during the poor weather conditions, there is little the airline can do to affect the departure times of disrupted flights originating at LGW. These flights are therefore left out of the IP problem. However, the airline may wish to make changes to subsequent flights to reduce the impact of delays propagating through the schedule. The simulation was used to estimate the ready times of the disrupted aircraft for these subsequent flights.

The heuristic approach for problem size reduction was used. As is intuitive, this had a smaller impact than in Problem 2 as the number of disrupted aircraft is considerably larger. This means that the set of airports not visited by the set of disrupted aircraft, A^*, is quite small. Only the small number of aircraft in the fleet that operate exclusively between the excluded airports are removed from the problem. The fleet was reduced to 101 aircraft, with 440 flights. The maximum allowable delay was $M = 60$ minutes with a step size of $m = 20$ minutes, producing 1760 flight delay arcs and 2586 ground arcs. This restriction was required to get solutions within a reasonable time, as the problem with $M = 180$ minutes and $m = 20$ minutes took over 2 hours to solve. The results from the IP, with 20 processors available, are shown in Table 5.4.2. Again, the approach takes a while to produce a Pareto optimal solution (the first of which is solution 12), adding further evidence that work is needed to im-

Table 5.4.2: Solutions from the IP for Problem 3 using the heuristic for problem size reduction. Maximum allowable runtime was 7200 seconds.

Solution	Cost (€1000)	Tail Number Exchanges	Delay (minutes)	Cancellations	Solution Time (seconds)
1	27.00	48	540	0	260.22
2	27.00	47	540	0	152.42
3	27.00	46	540	0	155.20
4	27.00	44	540	0	180.76
5	27.00	41	540	0	127.20
6	27.00	35	540	0	549.09
7	27.00	34	540	0	142.94
8	27.00	31	540	0	149.41
9	27.00	23	540	0	162.92
10	27.00	20	540	0	93.69
11	27.00	19	540	0	52.38
12	27.00	17	540	0	105.67
13	29.00	13	580	0	67.41
14	31.00	9	620	0	72.30
15	33.00	8	660	0	61.63
16	35.00	4	700	0	447.72

prove the first stage of the search procedure. Despite the IP having a similar number of variables to the Problem 2 IP (around 440,000), there are approximately 150,000 more constraints. This change in structure seems to make the problem much harder to solve. The solution times are quite long in a number of cases, even in the highly restrictive choices of M and m, highlighting the need for better problem reduction methods and more specialised solution approaches.

It is interesting to note that if m is taken to be 10 minutes instead of 20 minutes, much better solutions are found, with the first Pareto optimal solution having a cost of €15,000 using 28 tail number exchanges. The solution with 17 tail number exchanges (equivalent to the first Pareto optimal solution in Table 5.4.2) had a cost of €17,000. The results highlight the benefit to the solutions of using a smaller time step m. The process did however take much longer to solve, with the initial solution taking 864.7 seconds, though subsequent solution times were similar to those in Table 5.4.2.

In the simulation, the weather scaling variable at LGW is set to 0.2 between 9am and 11am, at which time it returns to 1. The rest of the simulation settings were identical to those in Problem 2.

The simulation optimisation was performed on solutions 12 to 16. The solutions had between 16 and 19 variables, which reduced the number of iterations that STRONG could go through with the budget of $N_{\max} = 2000$ replications, as more design points were required to estimate the models. However, this highlights a benefit of using CEA to build the designs over factorial designs, which would require 2^{19} design points, instantly using up the simulation budget.

There were 50 macro-replications of the simulation optimisation process, each with

$N^p_{\max} = 50$. The solutions were simulated using 1000 CRN replications. The means and 0.95 quantiles are shown in Figures 5.4.8 and 5.4.9, respectively. These plots show substantial improvements in both the mean and the 0.95 quantile for all the macro-replications. The performance also seems to be quite similar amongst the solutions found. The plan, defined predominantly by the aircraft allocation, does not appear to impact the overall solution performance, as all the histograms are very similar.

Figure 5.4.10 shows quite a different relationship between the mean and 0.95 quantiles when compared to the previous problems. The relationship appears to be positive and almost linear, with no trade-off between these objectives.

The plots also highlight a difference in the problems structure. Whilst in the previous problems the IP produced large benefits over the 'No Action' policy, here the benefit is marginal in expectation. The real gain then comes from the simulation optimisation. The simulation optimisation largely reduces the planned delays, perhaps exploiting slack in the schedule, both in terms of turn times and block times. However, the change in aircraft allocation from the IP is what allows this to happen.

5.5 Simulation Optimisation Parameter Settings

This section performs a small investigation of parameter settings in the simulation optimisation algorithm. This is undertaken for Problem 1 in Section 5.4.1, as this is the least computationally intensive. Comparisons are through the ECDF of the estimated optimality gap of the mean cost (relative to the best solution found across all settings), using a principle similar to the performance profile proposed by Dolan

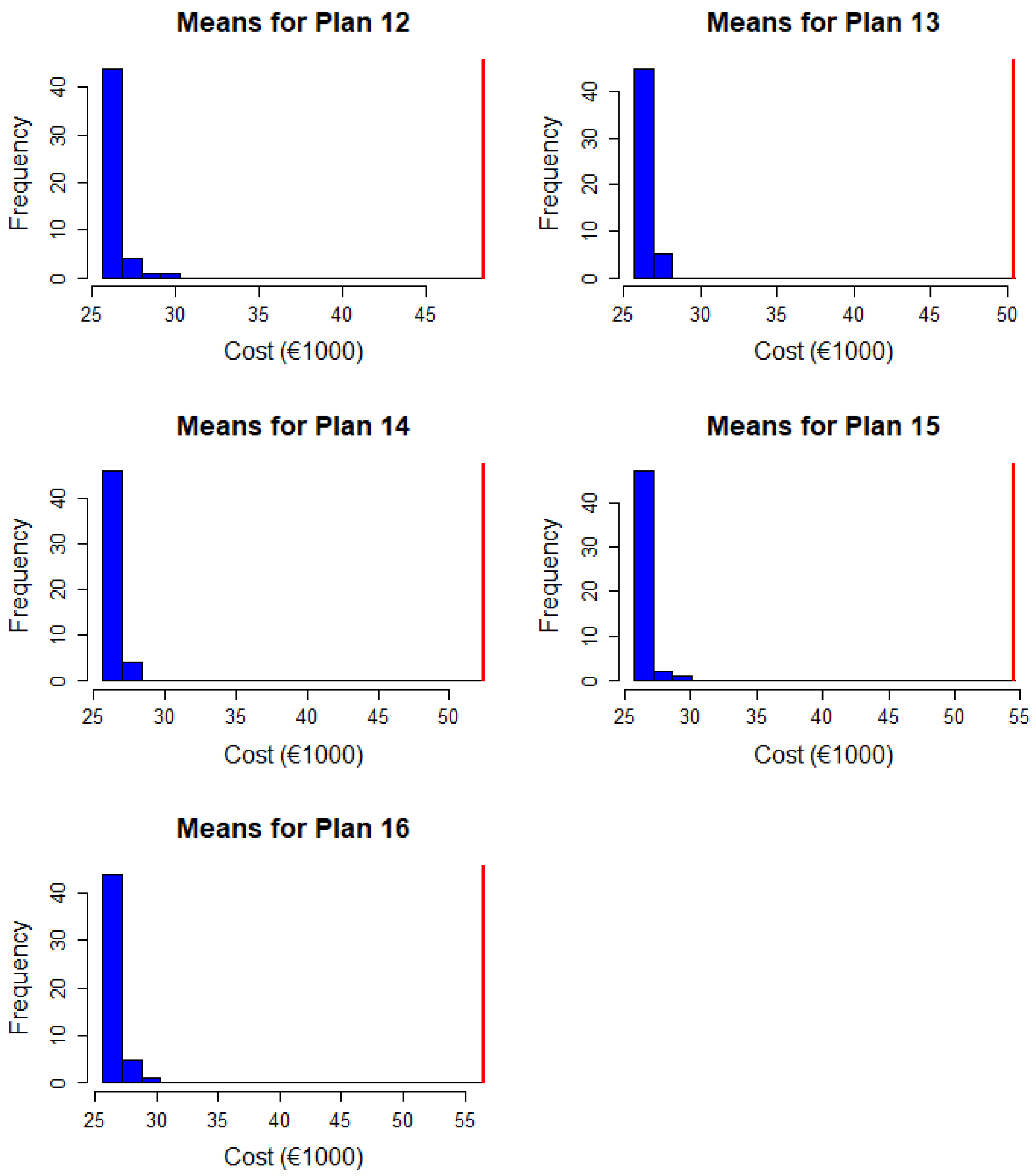

Figure 5.4.8: Problem 3 histograms of the mean of each solution from the simulation optimisation. Each plot corresponds to one of the IP solutions. The red line is the mean of the starting solution from the IP.

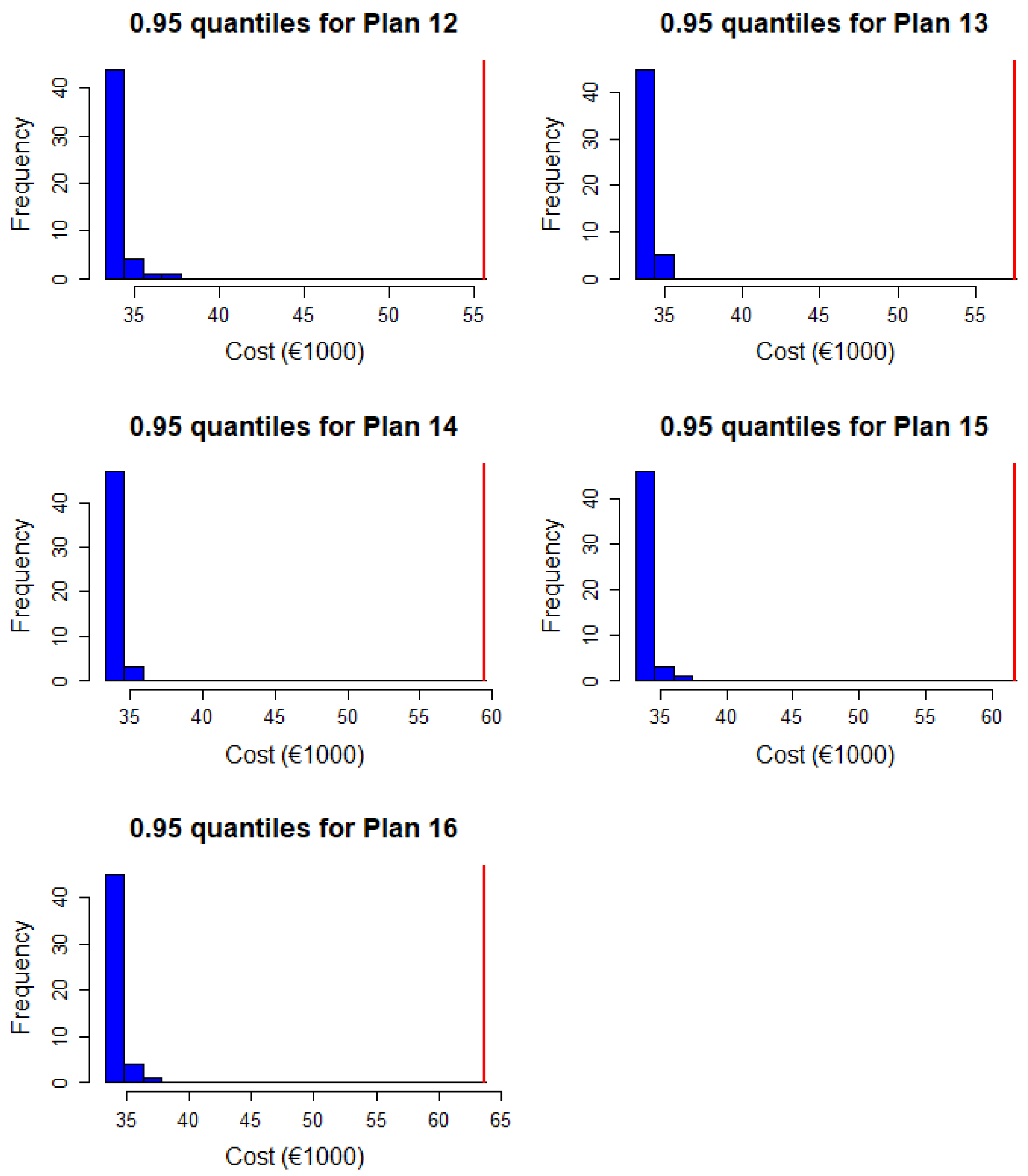

Figure 5.4.9: Problem 3 histograms of the 0.95 quantile of each solution from the simulation optimisation. Each plot corresponds to one of the IP solutions. The red line is the mean of the starting solution from the IP.

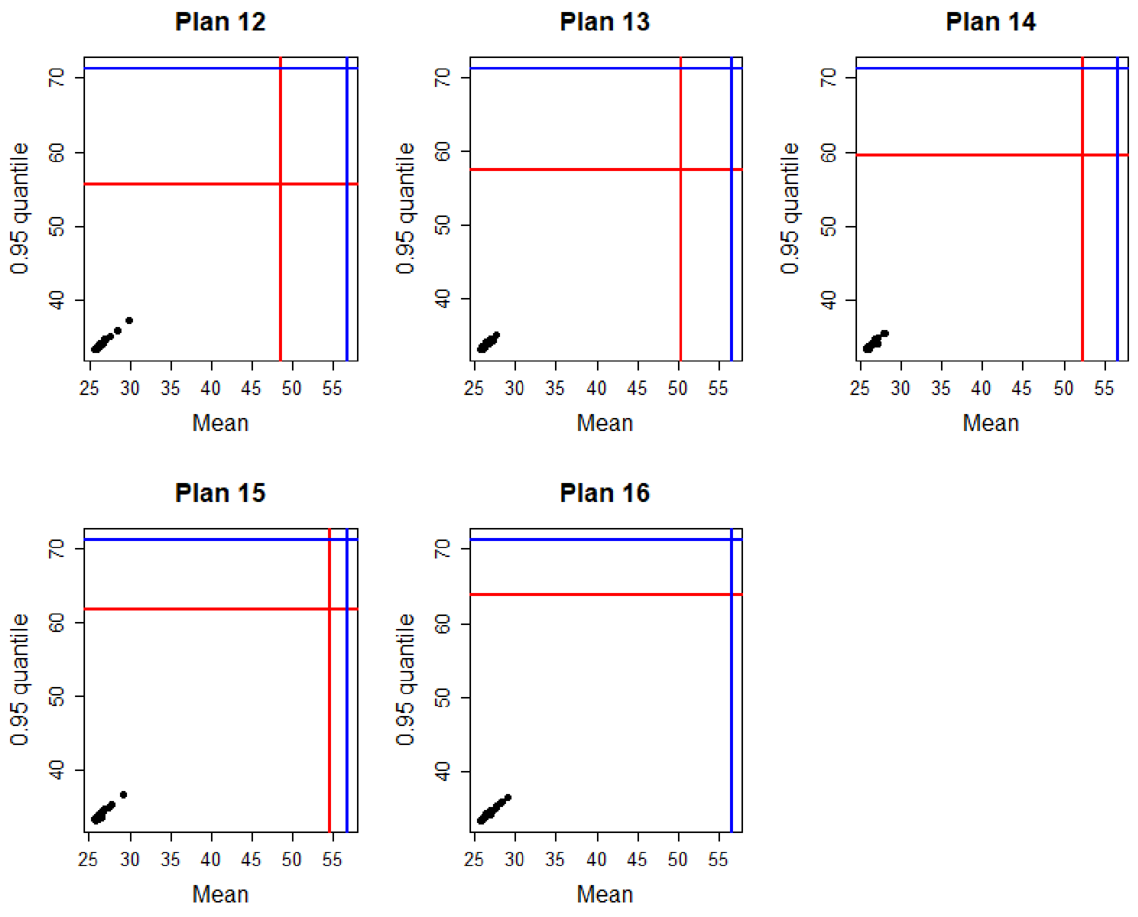

Figure 5.4.10: Mean against 0.95 quantile of solutions found for each plan from the IP in Problem 3. Red lines indicate the performance of the starting solution from the IP. The blue lines show the performance when no action is taken.

and Moré (2002) for comparing deterministic algorithms across multiple problems.

The limit in the design point replications creates a trade-off between using the budget to create better models and making more steps. To consider this trade-off, the optimisation on each plan was repeated using the algorithm with $N^p_{\max} = 25$ instead of 50. The overall performance was quite similar. To quantify the behaviour, we considered the 'optimality gap' of each macro-replication, defined as the proportion above the best mean found for the corresponding plan by any of the algorithm settings. The best found is used as an estimate of the optimal solution, which is unknown. This 'optimality gap' of the solution of the k^{th} macro-replication of Plan p for setting s, $(\mathbf{x}_p, \mathbf{d}_{kps})$, is

$$O(\mathbf{x}_p, \mathbf{d}_{kps}) = \frac{\hat{g}(\mathbf{x}_p, \mathbf{d}_{kps}) - \min_{s,k}\{\hat{g}(\mathbf{x}_p, \mathbf{d}_{kps})\}}{\min_{s,k}\{\hat{g}(\mathbf{x}_p, \mathbf{d}_{kps})\}}.$$

The ECDF of the optimality gaps for each algorithm setting is shown in Figure 5.5.1, combining all five plans into on plot. A point (x, y) on a line shows that in $100y\%$ of the macro-replications, the algorithm was within a factor of x of the best expected cost found. The higher the line, the better the algorithm has performed. The plot shows that across the 500 macro-replications of the simulation optimisation (100 for each plan) there is little to choose between the two settings, suggesting there is no significant difference between them. This is not to say that $N^p_{\max}$ is unimportant, as the effects may be more clear as the computational budget, $N_{\max}$, increases or for greater values of $N^p_{\max}$.

The third line in Figure 5.5.1 is the ECDF of the optimality gap when including all flight delays as decision variables. Chapter 4 mentions that the problem size for

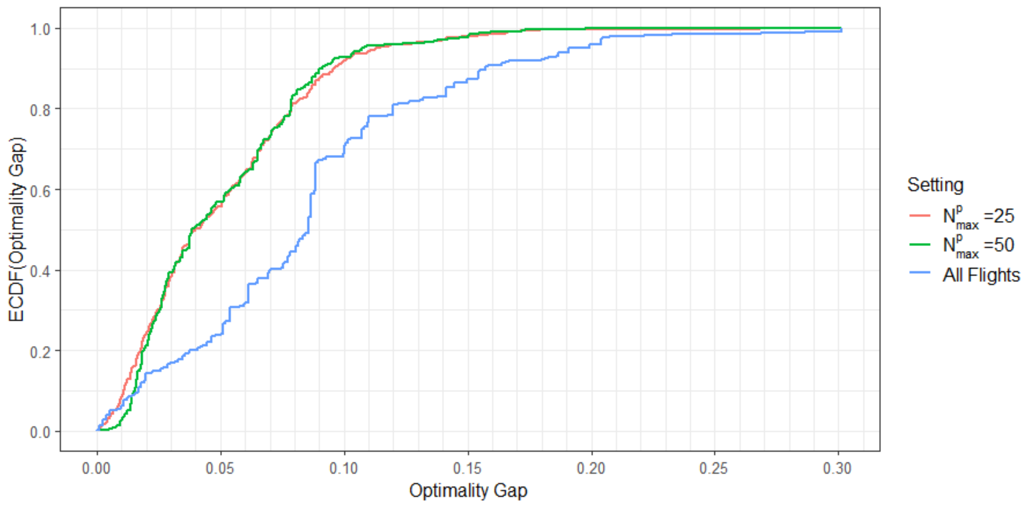

Figure 5.5.1: Empirical optimality gap for the algorithm with $N^p_{\max} = 25$ and $N^p_{\max} = 50$ and All Flights, from all five IP starting solutions.

the simulation optimisation is reduced by only including the flights that the IP gives a non-zero delay as variables. To test whether this led to better solutions being missed, 100 macro-replications for each of the five starting solutions were carried out using all 54 flights as variables. This algorithm took only two or three steps per macro-replication as the designs required more points. The time to run the CEA algorithm also became significant, even for the linear models. The initial performance is comparable with the other algorithms, but this happens less than 10% of the time, after which it is considerably worse. This comparison was consistent across each plan, except Plan 2, see Figure 5.5.2. Here the initial performance is better, up to the 30% point, after which the performance is worse or comparable. It is important to note that the 'All Flights' case overran the budget significantly more, and could take twice as long to run.

We are also interested in the effect the computational budget, $N_{\max}$, has on the

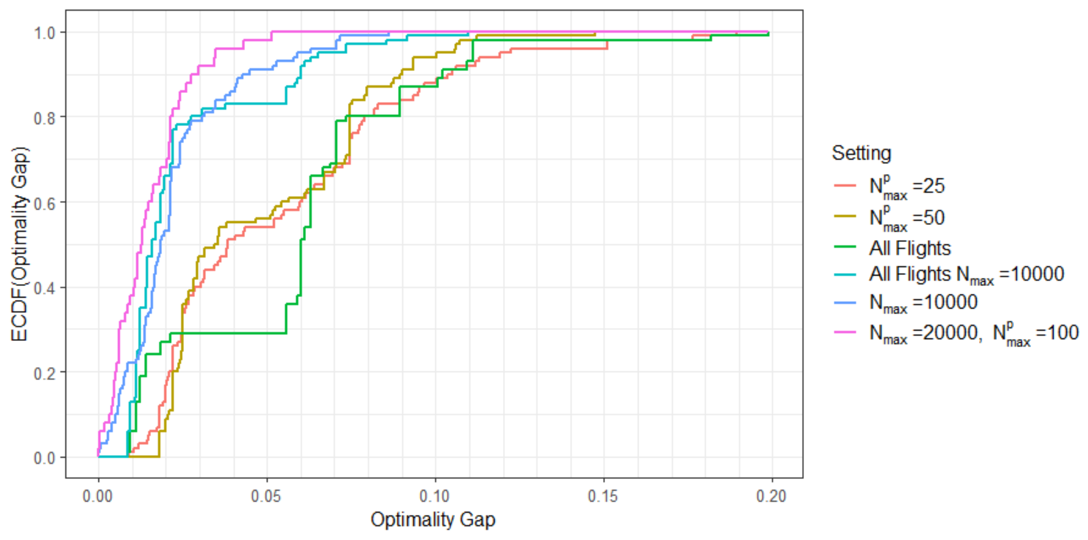

Figure 5.5.2: Empirical optimality gap for Plan 2 for all algorithm settings.

optimisation performance. We ran 100 macro-replications of the algorithm with $N_{\max} = 10{,}000$ (using both the reduced and full problems) and 50 macro-replications of the algorithm with $N_{\max} = 20{,}000$ and $N^p_{\max} = 100$, all for Plan 2. The resulting optimality gaps are shown in Figure 5.5.2. The plot shows that the additional budget allowed the reduced problems to find better solutions, an improvement of between 1% and 2%. The setting with the largest budget dominates the other algorithm settings and is the least variable, obtaining within 5.2% of the best in all 50 cases. The performance of the $N_{\max} = 10{,}000$ is also comparable with this. However, the limited budget cases do perform significantly worse, though they are within 10% most of the time and have a much lower run time. This shows the benefits of more computing time for this problem, and should be noted for implementation.

It is clear from each of the plots that the relative gap can be quite large, and that the algorithm lacks some consistency in achieving near optimality. A 1% optimality gap is not the norm, though this improves for larger budgets.

Another interesting observation, which is also true for Problems 2 and 3, is that the trust-region radius, Δ_j, rarely went below threshold to switch to the quadratic model, $\tilde{\Delta}$, and so the majority of the movement was through the linear model, rather than the quadratic model. This is due to the choice of γ_0, Δ_0, $\tilde{\Delta}$ and $N_{\max}$, which have not been tuned in our case. A smaller value of γ_0 would lead to the quadratic model being used more often. This could lead to fewer unsuccessful iterations, and so more progress. On the other hand, with a fixed budget, it could lead to fewer iterations as the designs would require more of the budget for each iteration.

5.6 Other Examples

In the work described in Chapter 7, several other disruption problems were solved. This section briefly reports the simulation optimisation results of these examples.

These problems are based on a fleet of 116 aircraft. In each case, the recovery window is until the end of the day, and so the problem size decreases as the start time gets later. The IP was set up with strict constraints on the position of aircraft at the end of the day, applying the return node constraint (3.3.10) to the entire fleet.

A summary of the disruptions is shown in Table 5.6.1. The disruptions are caused by either an aircraft being grounded for several hours due to a technical issue (Days 1, 3 and 4) or weather events causing high airport congestion (Days 2 and 6) or closure (Day 5). Days 1, 3 and 4 all involve hub airports of varying size, LGW, Geneva Airport (GVA) and MAN, allowing flexibility in the solution as multiple aircraft come

Table 5.6.1: Description of each problem and simulated cost performance (€1000) of No Action and IP solutions, with 95% confidence interval halfwidths in parentheses.

Problem	Disrupted	Time	No Action		IP Solution	
	Aircraft		$\hat{g}(\mathbf{x}, \mathbf{d})$	$\hat{q}_{0.95}$	$\hat{g}(\mathbf{x}, \mathbf{d}_0)$	$\hat{q}_{0.95}$
Day 1	1	16:00	61.76 (1.01)	75.61	13.96 (0.12)	17.47
Day 2	2	9:40	41.09 (0.66)	53.77	32.84 (0.48)	38.27
Day 3	1	8:45	154.64 (1.67)	176.21	6.75 (0.83)	16.00
Day 4	1	13:20	165.3 (2.65)	210.75	26.3 (0.39)	30.45
Day 5	2	10:40	28.77 (0.58)	34.97	22.56 (0.49)	27.87
Day 6	5	10:10	22.61 (0.78)	29.81	23.51 (0.76)	28.55

through the hub airports. The repair time distribution for each is a $\Gamma(100, 3)$, and a delay of 370 minutes is applied in the IP. For the weather affected days, busy airports or periods where weather was more likely to cause congestion were chosen. Day 2 occurs at AMS, one of Europe's busiest airports. Day 5 occurs at a small hub airport, Nantes Atlantique Airport (NTE). Even though this is not a busy airport, the period of closure leads to a backlog when the airport reopens. The Day 6 disruption occurs at Paris Orly Airport (ORY). It is expected to affect five aircraft directly, some of which have not yet arrived at ORY. As in Problem 3, the flights directly disrupted by the weather are not changed and only subsequent flights are included in the IP. In all problems, the turn time distributions were identical to those in Problems 2 and 3.

Table 5.6.1 also shows the simulation-based cost comparison between not taking any action and the IP solution. From these we can see that altering the aircraft allocation through the IP has the largest effect in the cases where an aircraft is grounded. This is not surprising as not making any changes to the schedule means that every subsequent flight of the disrupted aircraft will be delayed by several hours, each incurring high compensation costs. In reality, this would not happen and so the 'No Action' benchmark is less relevant. As the Day 1 disruption occurs in the afternoon, there are fewer future flights to be impacted, resulting in a smaller overall cost. In the weather related cases (Days 2, 5 and 6), where more flights are delayed but by a smaller amount, there seems more scope for recovery by using the buffer built into the schedule without schedule change. Whilst the IP can make positive aircraft exchanges, the impact is smaller. In the case of Day 6, this even leads to a negative impact. This could be due to the estimated ready times for disrupted aircraft being too conservative, leading to unnecessary delays.

Due to limited computational resources, we have only selected one IP solution from each problem to perform the simulation optimisation on, with 50 macro-replications for Day 1 and 25 for the others. As in Section 5.4, the results were simulated 1000 times using CRN to produce estimates of their mean and 0.95 quantile performance in cost. The less extensive results from these problems are shown in Figures 5.6.1, 5.6.2 and 5.6.3. The overall picture is consistent with the results of Section 5.4: the simulation optimisation almost always reduces the mean costs from the IP based solutions but is not always successful in reducing the 0.95 quantile. In all but Day 5, the STRONG algorithm was able to reduce both the mean and 0.95 quantile of the solution for all starting seeds.

However, on Day 5 there are three macro-replications that fail to find a noticeable improvement. Figure 5.6.3 shows that these solutions do not get worse (as perhaps suggested by the histogram format in Figure 5.6.1). On two occasions the algorithm finds a solution with only a slight improvement in mean and a larger 0.95 quantile. Perhaps more concerning is that in one case of Day 5 the algorithm repeatedly rejects proposed solutions and so ends up where it began (seen as a dot at the intersection of the red lines in Figure 5.6.3). It may be that the objective function in this case is dominated by the variance of the costs or that the trust-region size does not shrink quickly enough to make use of a quadratic model, i.e., γ_0 is too large and $\tilde{\Delta}$ too small, though this has not been fully investigated.

Note that on Day 3, the IP rearranged the aircraft allocation in such a way that no delays were required. Therefore, the simulation optimisation was not needed. We believe that there will be disruptions of this kind, particularly at hub airports where there are often many options for rearranging aircraft allocations.

Figure 5.6.3 also shows the relationship between the mean and 0.95 quantile. None of these problems appear to have the conflicting relationship between quantile and expected value that was observed in Problem 1 (Section 5.4.1). The relationship looks stronger for the weather related problems, Days 2, 5 and 6. The reason behind this has not been explored.

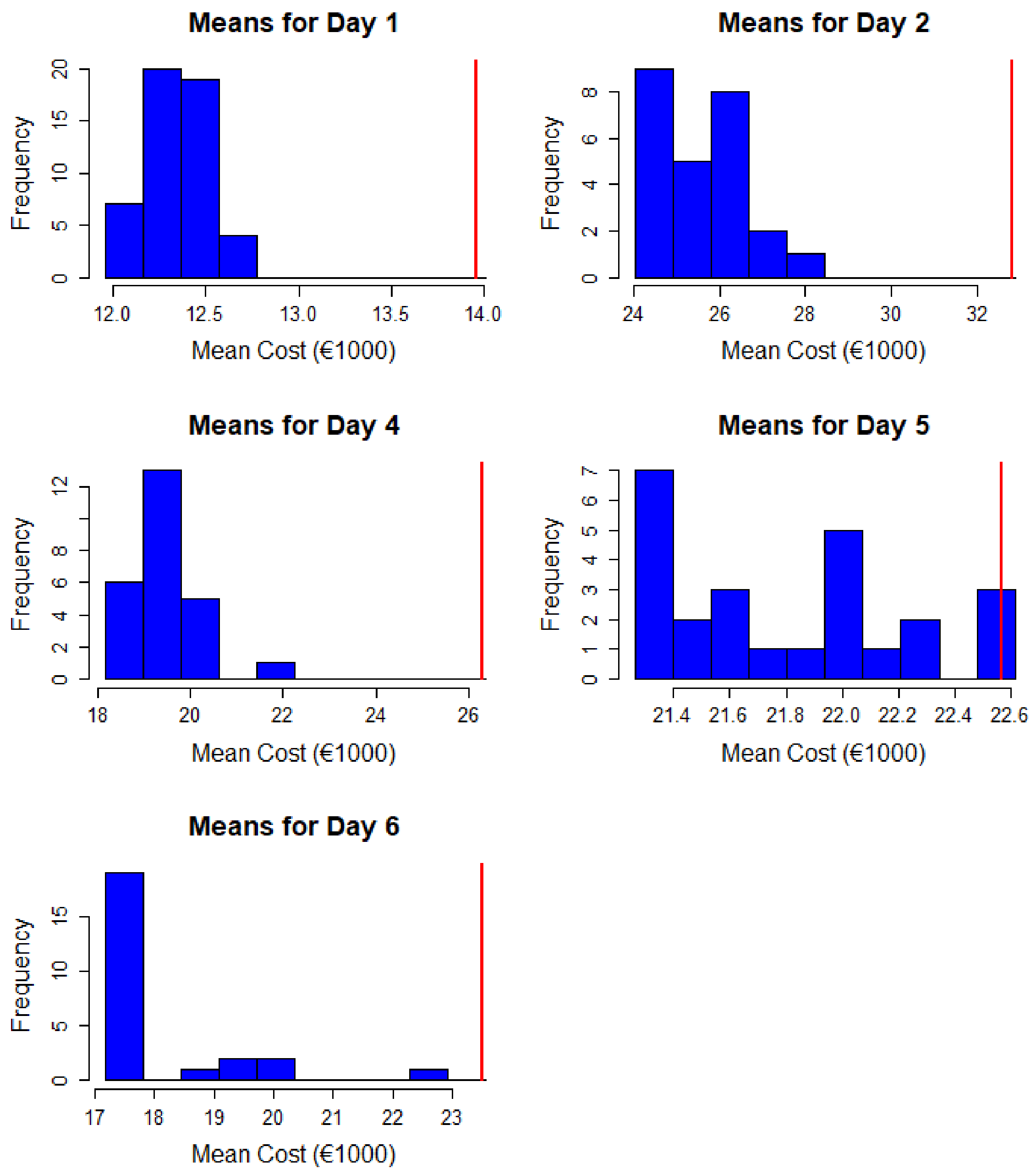

Figure 5.6.1: Histograms of the mean cost for the simulation optimisation solution in the different problems. The red line is the mean of the starting solution from the IP.

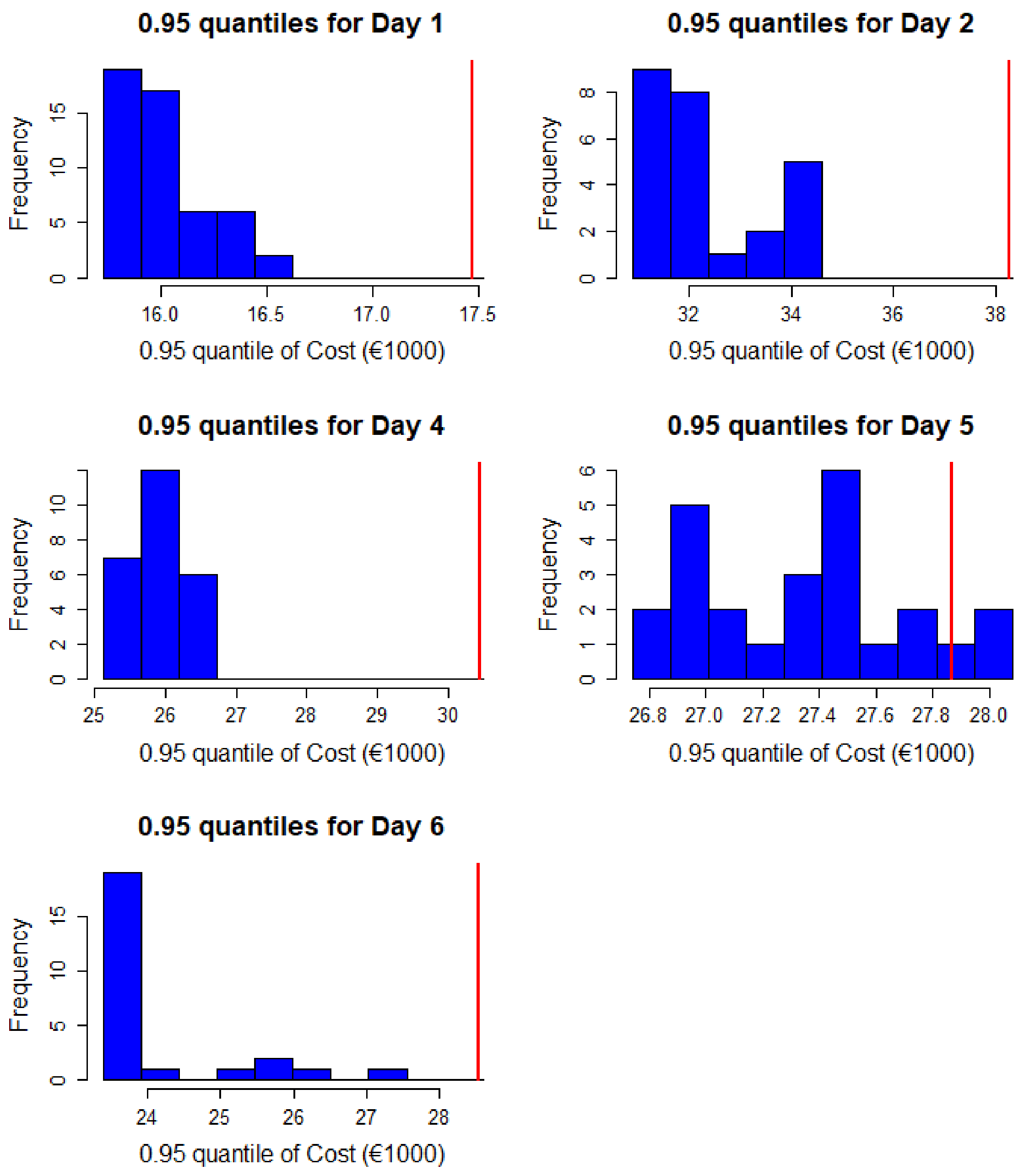

Figure 5.6.2: Histograms of the cost 0.95 quantile for the simulation optimisation solution in the different problems. The red line is the 0.95 quantile of the starting solution from the IP.

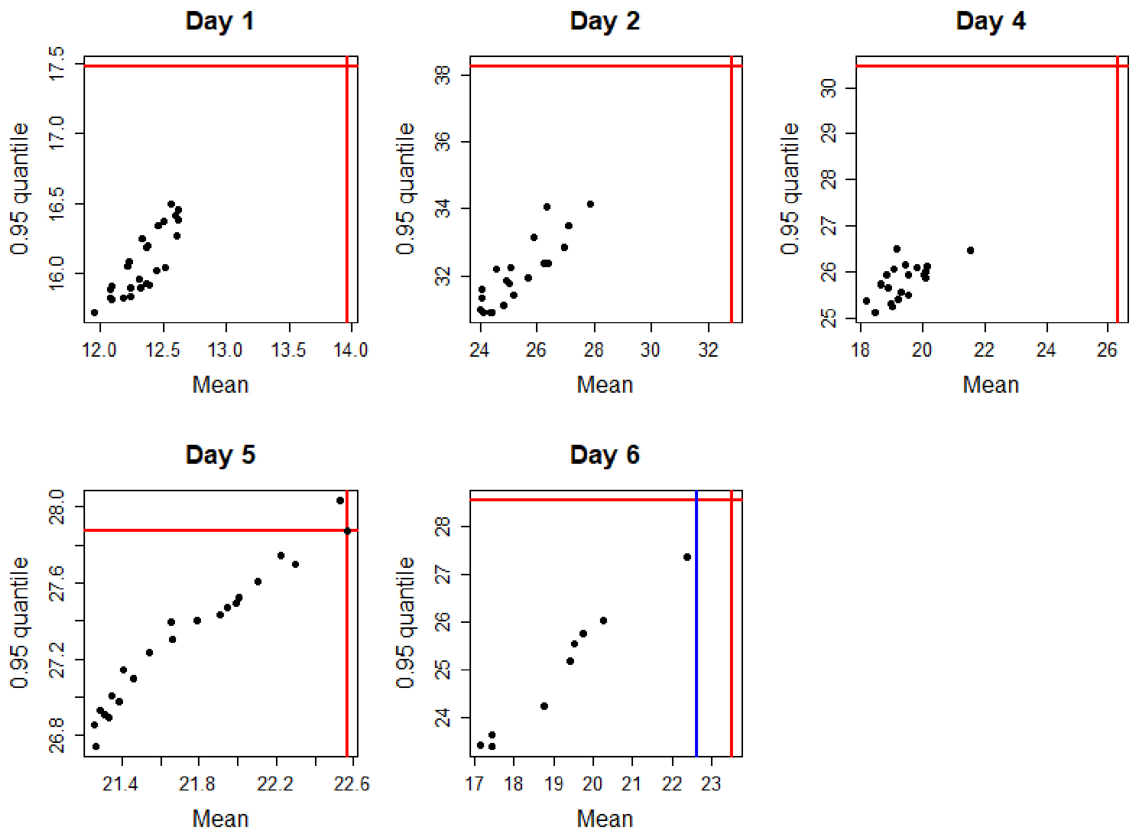

Figure 5.6.3: Mean against 0.95 quantile for each problem. Red lines indicate the performance of the starting solution from the IP. The blue lines show the performance when no action is taken.

5.7 Discussion and Conclusions

This chapter has presented and discussed the empirical performance of the multi-fidelity modelling approach to the ARP described in Chapters 3 and 4. Three problems of differing size and complexity have been described in detail, and the results from both the Integer Program and simulation optimisation have been examined. Six other problems have also been discussed. These suggest that combining deterministic models with simulation optimisation can provide good solutions to the ARP.

Overall, the results point to promising performance, showing that the IP solutions often offer an improvement over the 'No Action' response, largely by altering the aircraft allocation. However, the results also demonstrate that, in ignoring stochastic system elements, the deterministic model underestimates the costs involved. The similar rankings between the cost estimates of the IP and the simulation justifies the use of multi-fidelity modelling.

The simulation optimisation process, using an extension of STRONG to handle bound constraints, also appears to succeed in improving the initial solutions from the IP. The simulation optimisation aims to reduce the expected costs of the solution, which it achieves in almost all of the cases considered. This demonstrates the benefits of using a high-fidelity model for further improvement, rather than just checking robustness. However, whilst the improvement is fairly consistent, the optimality gap can be quite large, especially for small computational budgets.

There are, in addition, a number of areas for improvement and further work that have been identified. The solution times involved in solving the deterministic ARP for the larger problems are too long for practical use. This highlights the need for better problem reduction methods and more specialised solution methods. As was mentioned in Section 5.4.3, the time discretisation used in the IP, m, can have a large effect on the solution quality. The trade-off between the solution quality and solution time is clear, with regards to m. This must be considered by the user during implementation. The results from the IP also suggest that using a pure version of the ϵ-constraint approach to multi-objective optimisation produces a number of cost optimal solutions before a Pareto optimal solution is found. To avoid this, either a more restrictive constraint on the number of exchanges could be added to the problem, or a hybrid solution method (for example, see Section 4.2.2 of Ansari et al. (2018)) that includes weighted versions of the exchange and delay objectives in the objective function. For example, the objective (3.3.2) could be replaced with

$$\min \qquad \sum_{a\in A}\sum_{f_\delta\in L} c^{f_\delta} x_a^{f_\delta} + \sum_{f\in F} C^f y^f + \varepsilon \sum_{a\in A}\sum_{f_\delta\in L}(1-o_a^f)x_a^{f_\delta}, \tag{5.7.1}$$

where $\varepsilon \ll c_d$. If ε is chosen to be sufficiently small, this addition would not hinder the optimal solution for disrupted flights, but would encourage movement towards the Pareto frontier by penalising aircraft exchanges that are not part of the disruption. Initial results applied to the three problems were promising, and are detailed in Appendix B. Passing a partial initial solution to the Gurobi Optimizer 7.0.2 with all undisrupted aircraft assigned to their original flights was also attempted, but this did not seem to improve the results. There are many other deterministic models and solution approaches available in the literature (see Chapter 2). There is no reason to believe that these could not replace the IP model here if they provided better starting solutions for the simulation optimisation or could be solved more quickly.

The simulation optimisation process currently takes a long time. This is due to a sequential approach of running the simulations. However, as described in Section 4.4, the algorithm could be parallelised in a natural way. This would reduce computation time considerably.

The STRONG algorithm performance is influenced by its parameters. Whilst we have briefly explored the effect of the computational budgets allocated to both the whole algorithm and individual designs, most of the parameters have not had any tuning. Some have been selected to reflect practical aspects, whereas others match the choices made in Chang et al. (2013). It would be advisable to perform formal parameter tuning across a range of offline disruptions.

In summary, this chapter has presented an empirical evaluation of the method proposed in Chapters 3 and 4. The performance suggests that the combination of deterministic and simulation optimisation can offer good solutions when measured using the simulation evaluation. We have also identified potential areas for improvement in the algorithm and further work.

Chapter 6

Extension of STRONG to Bound Constraints

6.1 Introduction

The evaluation of the multi-fidelity modelling approach in Chapter 5 is based on one particular simulation model and a particular choice of objective function. However, there are other aspects of the Aircraft Recovery Problem an airline might wish to include in the model, such as costs for passengers missing a connection. To show the potential of the proposed approach to find improved solutions when a different simulation model is used, this chapter presents a deeper consideration of the simulation optimisation algorithm presented in Chapter 4. A limitation that remains is that the feasible region must be a hyper-box, constrained only by bounds on the decision variables.

A number of extensions have been made to the base algorithm, STRONG (Chang et al., 2013), to enable it to handle bound constraints. More details on some of these aspects are given: Section 6.2 discusses the change of criticality measure; Section 6.3 presents the details of the Coordinate-Exchange Algorithm (CEA) to produce a good design for estimating the meta-model; and Section 6.4 describes the algorithm for producing the proposed step.

Throughout this chapter, we consider a general simulation optimisation problem with continuous decision variables and bound constraints. For this reason, we drop the explicit dependence on the aircraft allocation $\mathbf{x}$ and the delays $\mathbf{d}$ in the differential operator from the notation.

Let $g\colon \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable real-valued function on $\mathcal{D} \subset \mathbb{R}^n$. We seek a minimum for this function within the hyper-box $\mathcal{D}$:

$$\begin{aligned} &\min_{\mathbf{d}} \;\; g(\mathbf{d}) \\ \text{subject to} \qquad &\mathbf{d} \in \mathcal{D} = \{\mathbf{d} \in \mathbb{R}^n : l^k \leq d^k \leq u^k, \;\; \forall k = 1, ..., n\} \end{aligned} \tag{6.1.1}$$

where

$$\mathbf{d} = (d^1, ..., d^n)^T.$$

The approach for problem (6.1.1) is the algorithm described in Chapter 4, although excluding the final step of testing against $\mathbf{d} = \mathbf{0}$, step 11 of Algorithm 4.1.

The most basic convergence statement to be made for (unconstrained) trust-region optimisation algorithms is that

$$\liminf_{j \to \infty} ||\nabla g(\mathbf{d}_j)||_2 = 0, \tag{6.1.2}$$

such as Theorem 4.5 of Nocedal and Wright (2006) and Theorem 6.4.5 of Conn et al. (2000). This statement says that the algorithm will find a stationary point and will always return to a stationary point in the future, or that there is at least one subsequence of solutions $\{\mathbf{d}_{j_i}\}_{i=0}^{\infty}$ such that the gradient converges to $\mathbf{0}$. If the threshold for the RC test, η_0, is greater than 0, the stronger result of $\lim_{j\to\infty} ||\nabla g(\mathbf{d}_j)||_2 = 0$ can also be achieved (see for example Theorem 6.4.6 of Conn et al. (2000)) but requires Equation (6.1.2) to be established as a first step. For second-order convergence, stronger assumptions about the Hessian of the meta-model are required (see Section 6.5 of Conn et al. (2000)).

In the constrained setting, the criticality measure is generalised to some function, $\pi\colon \mathcal{D} \to \mathbb{R}^+$, and so condition (6.1.2) becomes

$$\liminf_{j\to\infty} \pi(\mathbf{d}_j) = 0. \tag{6.1.3}$$

If one is to prove convergence of the extended version of STRONG presented in Chapter 4 for stochastic problems, Equation (6.1.3) with a probability statement, that is,

$$\liminf_{j\to\infty} \pi(\mathbf{d}_j) = 0 \qquad \text{w.p. } 1, \tag{6.1.4}$$

would be the initial target. Following the description of the algorithm details in Sections 6.2, 6.3 and 6.4, this chapter sets out a pathway to achieve this result. Section 6.5 discusses some of the assumptions needed and additional choices for algorithm quantities and procedures that we believe are necessary for this proof. Section 6.6 discusses the major steps of a proof of convergence, presenting some proofs of lemmas and highlighting where further work is required to complete the proof. Conclusions, further work and ideas for algorithm improvement are given in Section 6.7.

6.2 Criticality Measure

In constrained optimisation, the notion of optimality must be extended beyond that of a stationary point, as an optimal solution may lie on the boundary of $\mathcal{D}$ without being a stationary point of g. One generalisation are the Karush-Kuhn-Tucker (KKT) conditions, which exploit Lagrangian multipliers. An alternative formulation is to consider the normal cone (see Section 3.2.3 of Conn et al. (2000)).

Definition 6.2.1. *Let $\mathcal{D}$ be a closed, convex subset of $\mathbb{R}^n$. The normal cone of $\mathcal{D}$ at $\mathbf{d} \in \mathcal{D}$ is defined to be the set*

$$\mathcal{N}(\mathbf{d}) := \left\{\mathbf{y} \in \mathbb{R}^n \;:\; \mathbf{y}^T(\mathbf{u} - \mathbf{d}) \leq 0 \; \forall \mathbf{u} \in \mathcal{D}\right\}.$$

The following is Theorem 3.2.9 of Conn et al. (2000), which discusses the link between a first-order critical point and the normal cone.

Theorem 6.2.2. *Suppose that $\mathcal{D}$ is non-empty, closed and convex, that g is continuously differentiable in $\mathcal{D}$, and that $\mathbf{d}^*$ is a first-order critical point for the minimisation of g over $\mathcal{D}$. Then, provided that a first-order constraint qualification holds at $\mathbf{d}^*$,*

$$-\nabla g(\mathbf{d}^*) \in \mathcal{N}(\mathbf{d}^*). \tag{6.2.1}$$

Constraint qualifications are defined in Appendix C.1.

To measure the distance from the normal cone at $\mathbf{d}$, Conn et al. (1993) introduced the following quantity.

Definition 6.2.3. *Let* $\mathbf{d} \in \mathcal{D}$ *and* $g \colon \mathcal{D} \to \mathbb{R}$ *be differentiable at* $\mathbf{d}$. *Then* $\chi \colon \mathcal{D} \times \mathbb{R}^+ \to \mathbb{R}$*, defined as*

$$\chi(\mathbf{d}, \theta) := \left| \min_{\mathbf{s}} \left\{ \nabla g(\mathbf{d})^T \mathbf{s} \colon \mathbf{d} + \mathbf{s} \in \mathcal{D} \ , \ ||\mathbf{s}||_2 \leq \theta \right\} \right| , \tag{6.2.2}$$

is the best possible and feasible decrease in a linear model at $\mathbf{d}$ *on a ball of radius* θ.

This quantity is thoroughly explored in Chapter 12 of Conn et al. (2000), where it is proven that this is a valid critically measure (Theorem 12.1.4). It has other useful properties, such as monotonicity in θ. In the unconstrained case, $\mathcal{D} = \mathbb{R}^n$, $\chi(\mathbf{d}, 1)$ reduces to the unconstrained criticality measure, $||\nabla g(\mathbf{d})||_2$.

In a stochastic setting, $\chi(\mathbf{d}, \theta)$ cannot be evaluated exactly, as the gradient can only be estimated by $\widehat{\nabla} g(\mathbf{d})$. We therefore define $\hat{\chi}(\mathbf{d}, \theta)$ as the estimate of $\chi(\mathbf{d}, \theta)$ using the gradient estimator:

$$\hat{\chi}(\mathbf{d}, \theta) := \left| \min_{\mathbf{s}} \left\{ \widehat{\nabla} g(\mathbf{d})^T \mathbf{s} \colon \mathbf{d} + \mathbf{s} \in \mathcal{D} \ , \ ||\mathbf{s}||_2 \leq \theta \right\} \right| \tag{6.2.3}$$

For convenience, we work with the following adaptation of χ.

Definition 6.2.4. *Let* $\mathbf{d} \in \mathcal{D}$ *and* $g \colon \mathcal{D} \to \mathbb{R}$ *be differentiable at* $\mathbf{d}$. *Then* $\pi \colon \mathcal{D} \to \mathbb{R}$*, defined as*

$$\pi(\mathbf{d}) := \min\{\chi(\mathbf{d}, 1), 1\}. \tag{6.2.4}$$

As with χ, we cannot calculate this exactly, and therefore estimate it with

$$\hat{\pi}(\mathbf{d}) := \min\{\hat{\chi}(\mathbf{d}, 1), 1\}. \tag{6.2.5}$$

6.3 Coordinate-Exchange Algorithm

Section 4.3.1(i) discussed the issue of building an experimental design, defined by a design matrix $\mathbf{D}_j$, to estimate the meta-model $\hat{r}_j$ when the current solution $\mathbf{d}_j$ is near the boundary. To tackle this problem, we use a Coordinate-Exchange Algorithm (CEA), proposed by Meyer and Nachtsheim (1995). The purpose of this algorithm is to reduce the problem of finding a D-optimal design (maximising $\det(\mathbf{D}_j^T\mathbf{D}_j)$) with a fixed budget of P design points into a sequence of smaller problems of moving the individual coordinates of the design point $\mathbf{d}_{jp}$ within the region of interest to maximise $\det(\mathbf{D}_j^T\mathbf{D}_j)$. This can be done as our region of interest $\mathcal{D}\cap\mathcal{B}_j$ is the Cartesian product of n intervals

$$\mathcal{D}\cap\mathcal{B}_j = \prod_{k=1}^{n}\left[\max\{l^k, d_j^k-\Delta_j\}, \min\{u^k, d_j^k+\Delta_j\}\right],$$

so the values that each component can take do not depend on the other coordinates. For simplicity, let $L_j^k = \max\{l^k, d_j^k-\Delta_j\}$ and $U_j^k = \min\{u^k, d_j^k+\Delta_j\}$. The design will be embedded in the feasible trust region $\prod_{k=1}^{n}\left[L_j^k, U_j^k\right]$. This section gives a more detailed description of the procedure.

Note that the p^{th} row of $\mathbf{D}_j$ refers to the design point $\mathbf{d}_{jp}$ but is represented by some row function:

$$\mathbf{D}_j^{p\cdot} = \phi(\mathbf{d}_{jp})^T.$$

As the design is centred at the current solution, $\mathbf{d}_j$, the row function accounts for the position of the design point relative to $\mathbf{d}_j$. In the linear case

$$\phi(\mathbf{d}_{jp})^T = \begin{bmatrix}1 & (d_{jp}^1-d_j^1) & \cdots & (d_{jp}^n-d_j^n)\end{bmatrix},$$

whilst in the quadratic case, interaction and squared terms are also included:

$$\phi(\mathbf{d}_{jp})^T =$$

$$\begin{bmatrix}1 & (d^1_{jp}-d^1_j) & \cdots & (d^n_{jp}-d^n_j) & (d^1_{jp}-d^1_j)(d^2_{jp}-d^2_j) & \cdots & (d^1_{jp}-d^1_j)^2 & \cdots & (d^n_{jp}-d^n_j)^2\end{bmatrix}.$$

The use of ϕ as a generic row function enables us to treat the linear and quadratic cases in a similar way. The covariance matrix of the model parameters is $\mathbf{C}_j = (\mathbf{D}_j^T\mathbf{D}_j)^{-1}$. The covariance of the predicted responses at points $\mathbf{d}$ and $\mathbf{d}'$ is given by

$$v(\mathbf{d},\mathbf{d}') = \phi(\mathbf{d})^T\mathbf{C}_j\phi(\mathbf{d}'). \tag{6.3.1}$$

The analysis of Lemma 3.2.1 and Theorem 3.2.1 of Fedorov (1972) (pages 161-163) shows that replacing a design point $\mathbf{d}_{jp}$ by a point $\mathbf{d}'$ changes $\det(\mathbf{D}_j^T\mathbf{D}_j)$ by a factor of

$$\delta(\mathbf{d}',\mathbf{d}_{jp}) = 1 + [v(\mathbf{d}',\mathbf{d}') - v(\mathbf{d}_{jp},\mathbf{d}_{jp})] + \left[v(\mathbf{d}',\mathbf{d}_{jp})^2 - v(\mathbf{d}',\mathbf{d}')v(\mathbf{d}_{jp},\mathbf{d}_{jp})\right]. \tag{6.3.2}$$

This becomes our objective function when moving $\mathbf{d}_{jp}$ to a new position $\mathbf{d}^*_p$:

$$\mathbf{d}^*_p = \arg\max_{\mathbf{d}'\in\mathcal{D}\cap\mathcal{B}_j} \delta(\mathbf{d}',\mathbf{d}_{jp}) \tag{6.3.3}$$

One can then repeatedly cycle through each design point $\mathbf{d}_{jp}$, moving each to improve the design and updating the design before moving onto the next. This is known as the modified Fedorov algorithm (Cook and Nachtsheim, 1980). Alternatively, Equation (6.3.2) suggests that prioritising points with a low predictive variance $v(\mathbf{d}_{jp},\mathbf{d}_{jp})$ gives the largest scope for improvement. Therefore, one can produce the order $\Pi = (p_1, ..., p_P)$ such that $v(\mathbf{d}_{jp_1},\mathbf{d}_{jp_1}) \leq v(\mathbf{d}_{jp_2},\mathbf{d}_{jp_2}) \leq \ldots \leq v(\mathbf{d}_{jp_P},\mathbf{d}_{jp_P})$. The reordering is done for the quadratic case, but not the linear. To maximise the improvement we move the point contributing least to the design (as it has the smallest predictive variance) and move it to a place of high predictive variance.

Problem (6.3.3) is still an n-dimensional polynomial of degree at least 4. Instead of solving this directly, the CEA simplifies it to a sequence of 1-dimensional problems when we move one coordinate at a time; i.e., move d_{jp}^k to

$$d_p^k = \arg\max\left\{\delta_p^k(d')\colon d' \in \left[L_j^k, U_j^k\right]\right\} \tag{6.3.4}$$

where

$$\delta_p^k(d') = \delta((d_{jp}^1, ..., d_{jp}^{k-1}, d', d_{jp}^{k+1}, ..., d_{jp}^n)^T, \mathbf{d}_{jp}). \tag{6.3.5}$$

This can be solved using an optimisation process or by a grid search (as suggested by Meyer and Nachtsheim (1995)). In our case the Brent optimiser (Brent (1973), page 79) is used in the linear case, starting at whichever interval end gives a higher δ_p^k. For the quadratic case, we use a 1-dimensional grid search with 100 points spanning the interval.

6.3.1 Initial Design Matrix

Creating the initial design matrix can be done randomly or using principles of iteratively reducing predictive variance. The key is to ensure that the design is non-singular. Galil and Kiefer (1980) propose selecting the first design point $\mathbf{d}_{j1}$ by maximising $\phi(\mathbf{d}_{j1})^T\phi(\mathbf{d}_{j1})$. Following that, if $\mathbf{D}_{jp}$ is the design matrix including the first p points, Galil and Kiefer (1980) propose selecting $\mathbf{d}_{jp}$ to maximise $\det(\mathbf{D}_{jp}^T\mathbf{D}_{jp})$. The objective for the augmentation can be written as

$$a_p(\mathbf{d}_{jp}) = \phi(\mathbf{d}_{jp})^T \mathbf{W}_p \phi(\mathbf{d}_{jp}) \tag{6.3.6}$$

where $\mathbf{W}_p$ changes depending on whether the design is yet saturated (i.e., has more design points than model parameters b)

$$\mathbf{W}_p = \begin{cases} I_b & \text{if } p = 1; \\ I_b - \mathbf{D}_{j(p-1)}^T(\mathbf{D}_{j(p-1)}\mathbf{D}_{j(p-1)}^T)^{-1}\mathbf{D}_{j(p-1)} & \text{if } 1 < p \leq b; \\ (\mathbf{D}_{j(p-1)}\mathbf{D}_{j(p-1)}^T)^{-1} & \text{if } b < p \leq P, \end{cases} \tag{6.3.7}$$

where I_b is the $b \times b$ identity matrix. Meyer and Nachtsheim (1995) again suggest that this optimisation occurs on a component-wise basis. We use the Brent optimiser for this. The algorithm for finding the initial matrix is given in Algorithm 6.1.

6.3.2 Overall Algorithm

Once the initial matrix has been created, the algorithm iteratively orders the design points by predictive variance, for each optimising coordinate-wise, and updating the design and covariance matrix $\mathbf{C}_j$ between each. Once three successive cycles through all points have produced an increase of less than ε_D (nominally 1%), the algorithm terminates. The full algorithm is shown in Algorithm 6.2.

Updating $\mathbf{C}_j$ (step 13 of Algorithm 6.2) can either be done by a direct recalculation or using the update formula in Equation (6) of Meyer and Nachtsheim (1995), which follows from Lemma 3.3.1 of Fedorov (1972) (page 167). Suppose we replace design point $\mathbf{d}_{jp}$ by $\mathbf{d}'$. Let $F_1 = [\phi(\mathbf{d}'), -\phi(\mathbf{d}_{jp})]$ and $F_2 = [\phi(\mathbf{d}'), \phi(\mathbf{d}_{jp})]$. Then the update

Algorithm 6.1 Creating a Non-singular Design Matrix

```
 1: Set number of design points $P$, dimension of space $n$, number of model parameters
    $b$ and hyper-box bounds $L_j^k$ and $U_j^k$
 2: Find $\mathbf{d}_{j1} \leftarrow \arg\max_{\mathbf{d}' \in \mathcal{D} \cap \mathcal{B}_j} \phi(\mathbf{d}')^T \phi(\mathbf{d}')$ using component-wise Brent optimiser
 3: $\mathbf{D}_{j1} \leftarrow \phi(\mathbf{d}_{j1})^T$
 4: for $p = 2, ..., P$ do
 5:     Set $\mathbf{d}_{jp} \leftarrow \mathbf{d}_j$ and $\mathbf{W}_p$ by Equation (6.3.7)
 6:     for $k = 1, ..., n$ do
 7:         Optimise $a_p(\mathbf{d}_{jp})$ over component $d_{jp}^k$ using Brent optimiser
 8:     end for
 9:     Augment design
            $\mathbf{D}_{jp} = \begin{bmatrix} \mathbf{D}_{j(p-1)} \\ \phi(\mathbf{d}_{jp})^T \end{bmatrix}$
10: end for
11: return $\mathbf{D}_{jP}$
```

Algorithm 6.2 Coordinate-Exchange Algorithm

```
1: Set number of design points $P$, dimension of space $n$, number of model parameters
   $b$, tolerance $\varepsilon_D$, trust-region size $\Delta_j$ and little progress counter $N \leftarrow 0$
2: Calculate limits of $\mathcal{D} \cap \mathcal{B}_j$: $L_j^k \leftarrow \max\{l^k, d_j^k - \Delta_j\}$ and $U_j^k \leftarrow \min\{u^k, d_j^k + \Delta_j\}$
3: Generate initial $\mathbf{D}_j$ using Algorithm 6.1
4: repeat
5:     $\xi \leftarrow \det(\mathbf{D}_j^T \mathbf{D}_j)$, $\Pi \leftarrow (1, 2, ..., P)$
6:     if Quadratic then
7:         $\Pi \leftarrow (p_1, ..., p_P)$: points increasing in $v(\mathbf{d}_{jp}, \mathbf{d}_{jp})$
8:     end if
9:     for $p \in \Pi$ do
10:        for $k = 1, ..., n$ do
11:            $d_{jp}^k \leftarrow \arg\max_{d' \in [L_j^k, U_j^k]} \delta_p^k(d')$ (using Brent optimiser or grid search)
12:            $\mathbf{D}_j^{p\cdot} \leftarrow \phi(\mathbf{d}_{jp})^T$
13:            Update $\mathbf{C}_j$ using Equation (6.3.8)
14:        end for
15:    end for
16:    if $\det(\mathbf{D}_j^T \mathbf{D}_j) < (1 + \varepsilon_D)\xi$ then
17:        $N \leftarrow N + 1$
18:    else
19:        $N \leftarrow 0$
20:    end if
21: until $N = 3$
22: return $\mathbf{D}_j$
```

of $\mathbf{C}_j$ is

$$\mathbf{C}_j \leftarrow \mathbf{C}_j - \mathbf{C}_j F_1[I_2 + F_2^T \mathbf{C}_j F_1]^{-1} F_2^T \mathbf{C}_j, \tag{6.3.8}$$

which only involves inverting a 2×2 matrix and some matrix multiplication. The savings become significant when n gets large.

6.4 Calculation of the Generalised Cauchy Point

STRONG produces an approximate solution to the trust-region sub-problem using the Cauchy point of the meta-model. With constraints, this is not guaranteed to be feasible. For bound constrained problems, Conn et al. (1988) propose an alternative step, known as the Generalised Cauchy Step. This minimises the meta-model along the *projected-gradient path*. Section 4.3.1 discussed the linear case, where there is an analytical expression defined component-wise by

$$\mathbf{s}_j(\tau^*)^k = \begin{cases} \max\{-\Delta_j, l^k - d_j^k\} & \text{if } \widehat{\nabla} g_j(\mathbf{d}_j)^k > 0; \\ \min\{\Delta_j, u^k - d_j^k\} & \text{if } \widehat{\nabla} g_j(\mathbf{d}_j)^k < 0; \\ 0 & \text{if } \widehat{\nabla} g_j(\mathbf{d}_j)^k = 0. \end{cases} \tag{6.4.1}$$

A brief description of the process for the quadratic case was also given. Both cases are about finding the step size along the projected-gradient path, τ^*, that minimises $\hat{r}_j$. This section gives a more detailed description of the algorithm for the quadratic case, and is based on Algorithm 17.3.1 of Conn et al. (2000, page 791).

The projected-gradient path consists of a sequence of straight lines, broken by points where a variable hits its boundary of the feasible trust region. Let τ_B^k be the 'break point' of variable k:

$$\tau_B^k = \begin{cases} \min\{u^k - d_j^k, \Delta_j\}/(-\widehat{\nabla} g_j(\mathbf{d}_j)^k) & \text{if } \widehat{\nabla} g_j(\mathbf{d}_j)^k < 0; \\ \max\{l^k - d_j^k, -\Delta_j\}/(-\widehat{\nabla} g_j(\mathbf{d}_j)^k) & \text{if } \widehat{\nabla} g_j(\mathbf{d}_j)^k > 0; \\ 0 & \text{if } \widehat{\nabla} g_j(\mathbf{d}_j)^k = 0. \end{cases} \tag{6.4.2}$$

This is the value of $\tau \geq 0$ such that the vector $-\tau\widehat{\nabla} g_j(\mathbf{d}_j)$ brings you to either the boundary in variable k of the trust region or the boundary in variable k of the feasible region, whichever is nearest.

The i^{th} straight line segment is defined by $p(\tau, \mathbf{d}_j)$ for τ in an interval between two break points, $\tau_B^{k_1}$ and $\tau_B^{k_2}$. For convenience of notation, we will label the smaller of these break points as τ_i and the larger as τ_{i+1}. Thus, $\tau_0 = 0 < \tau_1 < \tau_2 < ... < \tau_m$ (where $m \leq n$) is the sequence of ordered break points. If two break points coincide, they are labelled as the same in the sequence. Let $\mathcal{K}_i = \{k\colon \tau_B^k \leq \tau_i\}$ be the set of all variable indices that are constrained on the i^{th} line segment. This is used to calculate the negative projected-gradient (and therefore the direction of travel), $\mathbf{z}_i$, in this segment. This is $-\widehat{\nabla} g_j(\mathbf{d}_j)$ with constrained variable components set to 0:

$$\mathbf{z}_i = -\widehat{\nabla} g_j(\mathbf{d}_j) + \sum_{k \in \mathcal{K}_i} \widehat{\nabla} g_j(\mathbf{d}_j)^k \mathbf{e}^k, \tag{6.4.3}$$

where $\mathbf{e}^k$ is the unit vector in variable k. At the start of the algorithm, there may be no constrained variables, and so we define $\mathbf{z}_0$ as

$$\mathbf{z}_0 = \begin{cases} -\widehat{\nabla} g_j(\mathbf{d}_j) & \text{if } \mathcal{K}_0 = \emptyset; \\ -\widehat{\nabla} g_j(\mathbf{d}_j) + \sum_{k \in \mathcal{K}_0} \widehat{\nabla} g_j(\mathbf{d}_j)^k \mathbf{e}^k & \text{if } \mathcal{K}_0 \neq \emptyset. \end{cases} \tag{6.4.4}$$

Also let $\mathbf{s}_i = p(\tau_i, \mathbf{d}_j) - \mathbf{d}_j$ be the step to the start of the i^{th} line segment and $\mathbf{s}^{GC}$ be the best solution found so far.

The projected-gradient path creates a one-dimensional piecewise quadratic function $q(\tau)$. Note that as $p(\tau, \mathbf{d}_j)$ is a continuous path in $\mathcal{D}$ and $\hat{r}_j$ is a quadratic function, $q(\tau)$ is continuous and well-defined at the break points, with $\lim_{\tau\uparrow\tau_i} q(\tau) = \lim_{\tau\downarrow\tau_i} q(\tau)$ for each i. It is not however, differentiable at these points. Let $q_i(\tau)$ be the quadratic function in τ along the i^{th} line segment:

$$\begin{aligned}
q_i(\tau) &:= \hat{r}_j(\mathbf{d}_j + \mathbf{s}_i + (\tau - \tau_i)\mathbf{z}_i) \\
&= \hat{g}_j(\mathbf{d}_j) + \widehat{\nabla} g_j(\mathbf{d}_j)^T(\mathbf{s}_i + (\tau - \tau_i)\mathbf{z}_i) \\
&\qquad + \frac{1}{2}\left(\mathbf{s}_i + (\tau - \tau_i)\mathbf{z}_i\right)^T \widehat{H}_j(\mathbf{d}_j)\left(\mathbf{s}_i + (\tau - \tau_i)\mathbf{z}_i\right) \\
&= \hat{g}_j(\mathbf{d}_j) + \widehat{\nabla} g_j(\mathbf{d}_j)^T\mathbf{s}_i + \frac{1}{2}\mathbf{s}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{s}_i \\
&\qquad + \left[\widehat{\nabla} g_j(\mathbf{d}_j)^T\mathbf{z}_i + \mathbf{s}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i\right](\tau - \tau_i) + \frac{1}{2}\mathbf{z}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i(\tau - \tau_i)^2,
\end{aligned}$$

for $\tau \in [\tau_i, \tau_{i+1}]$. This has first and second derivatives with respect to τ:

$$\begin{aligned}
& q_i'(\tau) = \widehat{\nabla} g_j(\mathbf{d}_j)^T\mathbf{z}_i + \mathbf{s}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i + \mathbf{z}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i(\tau - \tau_i) \\
\Rightarrow \quad & q_i'(\tau_i^+) := \lim_{\tau\downarrow\tau_i} q_i'(\tau) = \widehat{\nabla} g_j(\mathbf{d}_j)^T\mathbf{z}_i + \mathbf{s}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i; \qquad (6.4.5) \\
& q_i''(\tau) = \mathbf{z}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i \\
\Rightarrow \quad & q_i''(\tau_i^+) := \lim_{\tau\downarrow\tau_i} q_i''(\tau) = \mathbf{z}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i. \qquad (6.4.6)
\end{aligned}$$

These properties allow us to quickly identify whether there is a local minimum on the i^{th} line segment. Let τ_i^* denote the local minimum of $q_i(\tau)$ (if it exists). This will be a local minimum of $q(\tau)$ when $q_i'(\tau_i^+) < 0$, $q_i''(\tau_i^+) > 0$ and $\tau_i^* < \tau_{i+1}$. That is, the gradient at the start of the interval is negative, there is positive curvature, and the minimum occurs before the i^{th} line segment ends. Then

$$\begin{aligned}
0 &= q_i'(\tau_i^*) \\
&= \widehat{\nabla} g_j(\mathbf{d}_j)^T \mathbf{z}_i + s_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i + \mathbf{z}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i(\tau_i^* - \tau_i) \\
\Rightarrow \qquad \tau_i^* &= \tau_i - \frac{\widehat{\nabla} g_j(\mathbf{d}_j)^T \mathbf{z}_i + \mathbf{s}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i}{\mathbf{z}_i^T \widehat{H}_j(\mathbf{d}_j)\mathbf{z}_i} \\
&= \tau_i - \frac{q_i'(\tau_i^+)}{q_i''(\tau_i^+)}.
\end{aligned}$$

The full algorithm for finding the Generalised Cauchy Point checks each straight line segment in turn for a local minimum, storing the best, and is shown in Algorithm 6.3. In line 6, $q_i'(\tau_i^+) \geq 0$ indicates a positive gradient, meaning there will be no local minimum on the interval. Therefore, we check the start of the interval against the current best. In line 10, the negative gradient, positive curvature and $\tau_i^* < \tau_{i+1}$ indicates that a local minimum does lie in the interval. We check this against the current best.

6.5 Assumptions

This section provides a list of the assumptions about the problems for which this method is most suitable, particularly for asymptotic performance. Some algorithm choices are described and notation is introduced.

6.5.1 Objective Function Assumptions

To make progress in studying the asymptotic properties, we must make some assumptions limiting the behaviour of the objective function over the feasible region.

Algorithm 6.3 Algorithm for Finding the Generalised Cauchy Point

```
1: Compute the break points $\tau_B^k$ for $1 \leq k \leq n$ using (6.4.2) and set $\tau_0 \leftarrow 0$.
2: Set $\mathcal{K}_0 \leftarrow \{k\colon \tau_B^k = 0\}$, $\mathbf{s}_0 \leftarrow 0$, $\mathbf{s}^{GC} \leftarrow 0$, $i \leftarrow 0$ and $\mathbf{z}_0$ using (6.4.4).
3: repeat
4:     Compute the first and second derivative of $q$ with respect to $\tau$ at $\tau_i$, $q_i'(\tau_i^+)$
       and $q_i''(\tau_i^+)$ from (6.4.5) and (6.4.6).
5:     Find the next break point, $\tau_{i+1}$, the first break beyond $\tau_i$.
6:     if $q_i'(\tau_i^+) \geq 0$ then
7:         if $\hat{r}_j(\mathbf{d}_j + \mathbf{s}_i) < \hat{r}_j(\mathbf{d}_j + \mathbf{s}^{GC})$ then
8:             $\mathbf{s}^{GC} \leftarrow \mathbf{s}_i$
9:         end if
10:    else if $q_i''(\tau_i^+) > 0$ and $\tau_i^* = \tau_i - q_i'(\tau_i^+)/q_i''(\tau_i^+) < \tau_{i+1}$ then
11:        if $\hat{r}_j(\mathbf{d}_j + \mathbf{s}_i + (\tau_i^* - \tau_i)\mathbf{z}_i) < \hat{r}_j(\mathbf{d}_j + \mathbf{s}^{GC})$ then
12:            $\mathbf{s}^{GC} \leftarrow \mathbf{s}_i + (\tau_i^* - \tau_i)\mathbf{z}_i$.
13:        end if
14:    end if
15:    Update $\mathcal{K}_{i+1} \leftarrow \{k\colon \tau_B^k \leq \tau_{i+1}\}$, $\mathbf{s}_{i+1} \leftarrow \mathbf{s}_i + (\tau_{i+1} - \tau_i)\mathbf{z}_i$ and $\mathbf{z}_{i+1}$ by (6.4.3).
16:    $i \leftarrow i + 1$
17: until $\tau_i = \max_k \tau_B^k$
18: if $\hat{r}_j(\mathbf{d}_j + \mathbf{s}_i) < \hat{r}_j(\mathbf{d}_j + \mathbf{s}^{GC})$ then
19:     $\mathbf{s}^{GC} \leftarrow \mathbf{s}_i$
20: end if
21: Return $\mathbf{s}^{GC}$ as the step.
```

Assumption AF.1: The objective function $g(\mathbf{d})$ is twice differentiable on $\mathcal{D}$.

Assumption AF.2: The objective function $g(\mathbf{d})$ is bounded below on $\mathcal{D}$.

Assumption AF.3: There exists a constant $0 < \beta_H < \infty$ such that the Hessian of the objective function $H(\mathbf{d})$ satisfies

$$||H(\mathbf{d})|| := \sup_{\mathbf{d}' \neq 0} \frac{||H(\mathbf{d})\mathbf{d}'||}{||\mathbf{d}'||_2} \leq \beta_H, \qquad \forall \mathbf{d} \in \mathcal{D}.$$

That is, the operator norm of the Hessian is bounded on $\mathcal{D}$.

As $\mathcal{D}$ is compact, Assumption AF.3 also implies that $H(\mathbf{d})$ is Lipschitz continuous, that is, there exists $L > 0$ such that

$$||H(\mathbf{d}) - H(\mathbf{d}')|| \leq L||\mathbf{d} - \mathbf{d}'||_2, \qquad \forall \mathbf{d}, \mathbf{d}' \in \mathcal{D}.$$

It is also clear that $L \leq 2\beta_H$. This assumption is required to ensure that the bias in the gradient estimator will become negligible with sufficiently many observations and decreasing trust-region size.

STRONG makes a further assumption on the objective function.

Assumption AF.4: The objective function is the expectation of some random variable

$$g(\mathbf{d}) = \mathbb{E}\left[G(\mathbf{d})\right]$$

where $G(\mathbf{d}) \sim N(g(\mathbf{d}), \sigma^2(\mathbf{d}))$, and $\sup_{\mathbf{d} \in \mathcal{D}} \sigma^2(\mathbf{d}) < \infty$.

6.5.2 Estimator Assumptions

As in STRONG, the following assumptions are made about the objective function and gradient estimators.

Assumption AE.1: Let $\hat{g}(\mathbf{d})$ be the estimator of $g(\mathbf{d})$ and let N be the number of replications used for this estimation. Then the estimator is uniformly convergent on $\mathcal{D}$:

$$\sup_{\mathbf{d}\in\mathcal{D}} |\hat{g}(\mathbf{d}) - g(\mathbf{d})| \to 0 \qquad \text{as } N \to \infty,$$

with probability 1, and so obeys the Uniform Law of Large Numbers (ULLN).

Assumption AE.2: Let $\widehat{\nabla} g(\mathbf{d})$ be the estimator of $\nabla g(\mathbf{d})$, and let M be the number of replications used within the experimental design (assuming the design has enough design points to support the model). Suppose further that M grows not only by increasing the replications at design points, but also by including additional design points closer to $\mathbf{d}$, and that these design points have more replications than those further from $\mathbf{d}$. Then the estimator is uniformly convergent on $\mathcal{D}$:

$$\sup_{\mathbf{d}\in\mathcal{D}} ||\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d})||_2 \to 0 \qquad \text{as } M \to \infty,$$

with probability 1, and so obeys the ULLN as the trust region shrinks to 0.

These two assumptions are important to ensure that, given sufficient simulation effort, one can get a good approximation of the objective function and its gradient. The specification about how M grows is to ensure that bias incurred from the meta-model assumption decreases with effort. To enable this, we force the number of observations used to estimate $\widehat{\nabla} g_j(\mathbf{d}_j)$ in iteration j (before the inner loop begins), M_j, to gradually increase; $M_j \to \infty$ as $j \to \infty$. The rate does not change the result, but a larger M_j will give a better gradient estimator at the price of more effort at each iteration.

6.5.3 Use of the ℓ_∞ Norm

The ℓ_∞ norm is uniformly equivalent to the ℓ_2 norm, giving two important bounds:

$$||\mathbf{d}||_\infty^2 = \max_{k=1,\dots,n} (d^k)^2 \leq \sum_{k=1}^{n} (d^k)^2 = ||\mathbf{d}||_2^2$$

$$\Rightarrow \qquad ||\mathbf{d}||_\infty \leq ||\mathbf{d}||_2 \tag{6.5.1}$$

$$||\mathbf{d}||_2^2 = \sum_{k=1}^{n} (d^k)^2 \leq \sum_{k=1}^{n} \max_{i=1,\dots,n} (d^i)^2 = n \max_{i=1,\dots,n} (d^i)^2 = n||\mathbf{d}||_\infty^2$$

$$\Rightarrow \qquad ||\mathbf{d}||_2 \leq \sqrt{n}||\mathbf{d}||_\infty \tag{6.5.2}$$

6.5.4 Meta-model

The meta-model used for the trust-region sub-problem takes either a linear or a quadratic form:

$$\hat{r}_j(\mathbf{d}_j + \mathbf{s}) := \begin{cases} \hat{g}_j(\mathbf{d}_j) + \widehat{\nabla} g_j(\mathbf{d}_j)^T \mathbf{s} & \text{if } \Delta_j > \tilde{\Delta}; \\ \hat{g}_j(\mathbf{d}_j) + \widehat{\nabla} g_j(\mathbf{d}_j)^T \mathbf{s} + \frac{1}{2} \mathbf{s}^T \widehat{H}_j(\mathbf{d}_j) \mathbf{s} & \text{if } \Delta_j \leq \tilde{\Delta}. \end{cases} \tag{6.5.3}$$

This ensures that the meta-model is twice continuously differentiable on the trust region for all iterations. We also impose an upper bound on the operator norm of the Hessian estimator $\widehat{H}_j(\mathbf{d}_j)$, $\kappa \geq 1$, such that

$$||\widehat{H}_j(\mathbf{d}_j)|| \leq \kappa - 1, \qquad \forall j. \tag{6.5.4}$$

If the fitted Hessian has $||\widehat{H}_j(\mathbf{d}_j)|| > \kappa - 1$, we replace it with $\widehat{H}_j(\mathbf{d}_j) = (\kappa - 1) \times \widehat{H}_j(\mathbf{d}_j)/||\widehat{H}_j(\mathbf{d}_j)||$. We also introduce the quantity

$$\kappa_j := 1 + ||\widehat{H}_j(\mathbf{d}_j)|| \leq \kappa \qquad \forall j. \tag{6.5.5}$$

6.5.5 Significance Level for the Sufficient Reduction Test

The Sufficient Reduction (SR) test at each iteration has a Type I error rate of α_j. To aid convergence α_j is chosen such that

$$\sum_{j=0}^{\infty} \alpha_j < \infty. \tag{6.5.6}$$

A simple way to ensure this condition is to let α_j decrease geometrically, such as $\alpha_j = \alpha_0 \alpha^j$.

6.6 Proof of Convergence

This section presents a series of results that are needed to prove convergence. For some, it is unclear whether or not they are true, but they would be required for convergence with probability 1 in the sense of Equation (6.1.4) to hold. Throughout, we make reference to results in Conn et al. (2000) which are quoted in Appendix Section C.2 for reference.

6.6.1 The Criticality Measure Estimate is Consistent

It is important to show that the estimator of the criticality measure in Equation (6.2.5) is adequate. We show that, under Assumption AE.2, $\hat{\pi}$ also follows the ULLN.

Lemma 6.6.1. *Suppose that Assumption AE.2 holds. Then*

$$\sup_{\mathbf{d} \in \mathcal{D}} |\hat{\pi}(\mathbf{d}) - \pi(\mathbf{d})| \to 0$$

with probability 1 as the number of observations used tends to infinity.

Proof. Take $\mathbf{d} \in \mathcal{D}$. Let $\mathcal{A} = \{\mathbf{y} \colon \mathbf{d} + \mathbf{y} \in \mathcal{D}, ||\mathbf{y}||_2 \leq 1\}$. Note that $\mathbf{0} \in \mathcal{A}$, and so

$$\min_{\mathbf{y} \in \mathcal{A}} \left(\widehat{\nabla} g(\mathbf{d})^T \mathbf{y} \right) \leq \widehat{\nabla} g(\mathbf{d})^T \mathbf{0} = 0.$$

Therefore,

$$\begin{aligned} 0 \geq \min_{\mathbf{y} \in \mathcal{A}} \left(\widehat{\nabla} g(\mathbf{d})^T \mathbf{y} \right) &= \min_{\mathbf{y} \in \mathcal{A}} \left(\left(\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d}) \right)^T \mathbf{y} + \nabla g(\mathbf{d})^T \mathbf{y} \right) \\ &\geq \min_{\mathbf{y} \in \mathcal{A}} \left[\left(\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d}) \right)^T \mathbf{y} \right] + \min_{\mathbf{y} \in \mathcal{A}} \left(\nabla g(\mathbf{d})^T \mathbf{y} \right). \end{aligned}$$

Using the definitions of $\chi(\mathbf{d}, 1)$ and $\hat{\chi}(\mathbf{d}, 1)$ and the triangle inequality we get:

$$\begin{aligned} \hat{\chi}(\mathbf{d}, 1) = \left| \min_{\mathbf{y} \in \mathcal{A}} \left(\widehat{\nabla} g(\mathbf{d})^T \mathbf{y} \right) \right| &\leq \left| \min_{\mathbf{y} \in \mathcal{A}} \left(\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d}) \right)^T \mathbf{y} \right| + \left| \min_{\mathbf{y} \in \mathcal{A}} \left(\nabla g(\mathbf{d})^T \mathbf{y} \right) \right| \\ &= \left| \min_{\mathbf{y} \in \mathcal{A}} \left(\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d}) \right)^T \mathbf{y} \right| + \chi(\mathbf{d}, 1) \\ \Rightarrow \qquad \hat{\chi}(\mathbf{d}, 1) - \chi(\mathbf{d}, 1) &\leq \left| \min_{\mathbf{y} \in \mathcal{A}} \left(\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d}) \right)^T \mathbf{y} \right|. \end{aligned} \tag{6.6.1}$$

Using a similar argument, one can show that

$$\chi(\mathbf{d}, 1) - \hat{\chi}(\mathbf{d}, 1) \leq \left| \min_{\mathbf{y} \in \mathcal{A}} \left(\nabla g(\mathbf{d}) - \widehat{\nabla} g(\mathbf{d}) \right)^T \mathbf{y} \right|. \tag{6.6.2}$$

Let $\mathbf{y}^*, \mathbf{y}' \in \mathcal{A}$ be the solutions of the optimisation problems in the right-hand sides of (6.6.1) and (6.6.2) respectively. Then, combining (6.6.1) and (6.6.2) gives us

$$\begin{aligned} |\hat{\chi}(\mathbf{d}, 1) - \chi(\mathbf{d}, 1)| &\leq \max \left\{ \left| \min_{\mathbf{y} \in \mathcal{A}} \left(\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d}) \right)^T \mathbf{y} \right|, \left| \min_{\mathbf{y} \in \mathcal{A}} \left(\nabla g(\mathbf{d}) - \widehat{\nabla} g(\mathbf{d}) \right)^T \mathbf{y} \right| \right\} \\ &= \max \left\{ \left| \left(\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d}) \right)^T \mathbf{y}^* \right|, \left| \left(\nabla g(\mathbf{d}) - \widehat{\nabla} g(\mathbf{d}) \right)^T \mathbf{y}' \right| \right\} \\ &\leq \max \left\{ ||\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d})||_2 \cdot ||\mathbf{y}^*||_2 , ||\nabla g(\mathbf{d}) - \widehat{\nabla} g(\mathbf{d})||_2 \cdot ||\mathbf{y}'||_2 \right\}, \end{aligned}$$

where the last inequality holds using the Cauchy-Schwarz inequality. Furthermore, as

$\mathbf{y}^*, \mathbf{y}' \in \mathcal{A}$, $||\mathbf{y}^*||_2, ||\mathbf{y}'||_2 \leq 1$. Thus

$$\begin{aligned}|\hat{\chi}(\mathbf{d},1) - \chi(\mathbf{d},1)| &\leq ||\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d})||_2 \max\{||\mathbf{y}^*||_2, ||\mathbf{y}'||_2\} \\ &\leq ||\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d})||_2. \end{aligned} \tag{6.6.3}$$

Now consider the error of $\hat{\pi}(\mathbf{d})$:

$$\begin{aligned}|\hat{\pi}(\mathbf{d}) - \pi(\mathbf{d})| &= |\min\{1, \hat{\chi}(\mathbf{d},1)\} - \min\{1, \chi(\mathbf{d},1)\}| \\ &= \begin{cases} 0 & \text{if } \hat{\chi}(\mathbf{d},1) > 1, \chi(\mathbf{d},1) > 1 \\ 1 - \chi(\mathbf{d},1) & \text{if } \hat{\chi}(\mathbf{d},1) > 1, \chi(\mathbf{d},1) \leq 1 \\ 1 - \hat{\chi}(\mathbf{d},1) & \text{if } \hat{\chi}(\mathbf{d},1) \leq 1, \chi(\mathbf{d},1) > 1 \\ |\hat{\chi}(\mathbf{d},1) - \chi(\mathbf{d},1)| & \text{if } \hat{\chi}(\mathbf{d},1) \leq 1, \chi(\mathbf{d},1) \leq 1 \end{cases} \\ &\leq |\hat{\chi}(\mathbf{d},1) - \chi(\mathbf{d},1)| \\ &\leq ||\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d})||_2, \end{aligned}$$

where we have used (6.6.3). As this applies for an arbitrary $\mathbf{d} \in \mathcal{D}$, we have that

$$\sup_{\mathbf{d} \in \mathcal{D}} |\hat{\pi}(\mathbf{d}) - \pi(\mathbf{d})| \leq \sup_{\mathbf{d} \in \mathcal{D}} ||\widehat{\nabla} g(\mathbf{d}) - \nabla g(\mathbf{d})||_2 \to 0$$

with probability 1, by AE.2, as the number of observations tends to infinity. Thus, the result holds. □

6.6.2 Meta-model Reduction

The first step of the proof of convergence of a trust-region procedure is to show that the choice of step gives a reduction in the meta-model that is bounded away from 0 as long as an improvement is possible. We separate this into the linear (Stage I) and quadratic (Stage II) cases.

Lemma 6.6.2. *Suppose that the Algorithm is in Stage I during iteration j. Then the proposed solution $\mathbf{d}_j^*$ produces the reduction:*

$$\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j^*) \geq \zeta_j := \hat{\pi}_j(\mathbf{d}_j) \min\{\Delta_j, 1\}. \tag{6.6.4}$$

Proof. Using the linear condition of Equation (6.5.3) we can see that

$$\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j + \mathbf{s}_j(\tau)) = -\widehat{\nabla} g_j(\mathbf{d}_j)^T \mathbf{s}_j(\tau) = |\widehat{\nabla} g_j(\mathbf{d}_j)^T \mathbf{s}_j(\tau)|.$$

As $\mathbf{s}_j(\tau)$ lies on the projected-gradient path, Theorem 12.1.4 of Conn et al. (2000) states that $\mathbf{s}_j(\tau)$ is the solution to the optimisation problem in Equation (6.2.3), so

$$\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j + \mathbf{s}_j(\tau)) = \hat{\chi}_j(\mathbf{d}_j, ||\mathbf{s}_j(\tau)||_2). \tag{6.6.5}$$

By the first part of Theorem 12.1.3 of Conn et al. (2000), $||\mathbf{s}_j(\tau)||_2$ is a non-decreasing function of $\tau \geq 0$. By Theorem 12.1.5(i) of Conn et al. (2000), $\hat{\chi}(\mathbf{d}_j, \theta)$ is a non-decreasing function of θ. Therefore, $\hat{\chi}_j(\mathbf{d}_j, ||\mathbf{s}_j(\tau)||_2)$ is non-decreasing in τ.

Thus, to maximise the model decrease, we wish to choose the largest τ possible, subject to

$$||\mathbf{s}_j(\tau)||_\infty \leq \Delta_j \tag{6.6.6}$$

$$\mathbf{d}_j + \mathbf{s}_j(\tau) \in \mathcal{D}. \tag{6.6.7}$$

Let τ' be the chosen value of τ.

Suppose, firstly, that $||\mathbf{s}_j(\tau')||_2 = 0$. Then $p(\tau, \mathbf{d}_j) = \mathbf{d}_j$ for all $\tau \geq 0$, which is equivalent to being first-order critical, by Theorem 12.1.2, of Conn et al. (2000), and so $\hat{\pi}_j(\mathbf{d}_j) = 0$. So (6.6.4) holds.

Next, suppose that $||\mathbf{s}_j(\tau')||_2 \geq 1$. Then, using Theorem 12.1.5(i) of Conn et al. (2000) again:

$$\hat{\chi}_j(\mathbf{d}_j, ||\mathbf{s}_j(\tau')||_2) \geq \hat{\chi}_j(\mathbf{d}_j, 1). \tag{6.6.8}$$

Now suppose that $0 < ||\mathbf{s}_j(\tau')||_2 < 1$. We split this into two cases.

Case 1 Suppose that the value of τ' is restricted by constraint (6.6.6). Then

$$\begin{aligned} &\Rightarrow \quad \exists k \text{ such that } |\mathbf{s}_j(\tau')^k| = \Delta_j < 1 \\ &\Rightarrow \quad 1 > ||\mathbf{s}_j(\tau')||_2 \geq \Delta_j \\ &\Rightarrow \quad \hat{\chi}_j(\mathbf{d}_j, ||\mathbf{s}_j(\tau')||_2) \geq \hat{\chi}_j(\mathbf{d}_j, \Delta_j) \end{aligned}$$

where the last line follows from Theorem 12.1.5(i) of Conn et al. (2000). Theorem 12.1.5(ii) states that $\chi_j(\mathbf{d}_j, \theta)/\theta$ is non-increasing in θ, so $\hat{\chi}_j(\mathbf{d}_j, \Delta_j)/\Delta_j \geq \hat{\chi}_j(\mathbf{d}_j, 1)/1$. Therefore

$$\hat{\chi}_j(\mathbf{d}_j, ||\mathbf{s}_j(\tau')||_2) \geq \Delta_j \hat{\chi}_j(\mathbf{d}_j, 1). \tag{6.6.9}$$

Case 2 Suppose instead that $\mathbf{s}_j(\tau')$ is restricted by constraint (6.6.7) but not (6.6.6). Then we must be at the end of the projected-gradient path, that is, $\mathbf{s}_j(\tau') = p(\tau_m, \mathbf{d}_j) - \mathbf{d}_j$. Therefore

$$||\mathbf{s}_j(\tau')||_2 = \lim_{\tau \to \infty} ||p(\tau, \mathbf{d}_j) - \mathbf{d}_j||_2 < \infty,$$

so by Theorem 12.1.3 of Conn et al. (2000)

$$||\mathcal{P}_{\mathcal{T}_{\mathcal{D}}(\mathbf{d}_j + \mathbf{s}_j(\tau'))}[-\widehat{\nabla} g_j(\mathbf{d}_j)]||_2 = 0,$$

where $\mathcal{T}_{\mathcal{D}}(\mathbf{d}_j + \mathbf{s}_j(\tau'))$ is the Tangent cone of $\mathcal{D}$ at the point $\mathbf{d}_j + \mathbf{s}_j(\tau')$ (see Definition C.1.1 in Appendix C.1). Note that $||\mathbf{s}_j(\tau')||_\infty < \Delta_j$, so $||\mathbf{s}_j(\tau')||_2 < \sqrt{n}\Delta_j$. Thus, using Theorem 12.1.5(iii) of Conn et al. (2000):

$$\begin{aligned}\hat{\chi}_j(\mathbf{d}_j, \sqrt{n}\Delta_j) &\leq |\widehat{\nabla} g_j(\mathbf{d}_j)^T \mathbf{s}_j(\tau')| + 2\sqrt{n}\Delta_j ||\mathcal{P}_{\mathcal{T}_{\mathcal{D}}(\mathbf{d}_j+\mathbf{s}_j(\tau'))}[-\widehat{\nabla} g_j(\mathbf{d}_j)]||_2 \\ &= |\widehat{\nabla} g_j(\mathbf{d}_j)^T \mathbf{s}_j(\tau')| \\ &= \hat{\chi}_j(\mathbf{d}_j, ||\mathbf{s}_j(\tau')||_2).\end{aligned}$$

Then, using Theorem 12.1.5(ii) of Conn et al. (2000) again, we have that

$$\hat{\chi}_j(\mathbf{d}_j, ||\mathbf{s}_j(\tau')||_2) \geq \hat{\chi}_j(\mathbf{d}_j, \sqrt{n}\Delta_j) \geq \sqrt{n}\Delta_j \hat{\chi}_j(\mathbf{d}_j, 1). \tag{6.6.10}$$

Combining (6.6.5), (6.6.8), (6.6.9) and (6.6.10) we obtain:

$$\begin{aligned}\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j + \mathbf{s}_j(\tau')) &\geq \hat{\chi}_j(\mathbf{d}_j, 1) \min\{1, \Delta_j, \sqrt{n}\Delta_j\} \\ &= \hat{\chi}_j(\mathbf{d}_j, 1) \min\{1, \Delta_j\} \\ &\geq \hat{\pi}_j(\mathbf{d}_j) \min\{1, \Delta_j\}.\end{aligned}$$

where the final inequality comes from $\hat{\pi}_j(\mathbf{d}_j) = \min\{1, \hat{\chi}_j(\mathbf{d}_j, 1)\}$. This achieves the result. □

The next result considers the model decrease that can be achieved when the quadratic model is used.

Lemma 6.6.3. *Suppose that the Algorithm is in Stage II during iteration j. Then the proposed solution $\mathbf{d}_j^*$ produces the reduction:*

$$\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j^*) \geq \zeta_j := \mu \hat{\pi}_j(\mathbf{d}_j) \min\left\{\frac{\hat{\pi}_j(\mathbf{d}_j)}{\kappa_j}, \Delta_j\right\}. \tag{6.6.11}$$

If in the i^{th} inner loop of iteration j, the proposed solution $\mathbf{d}^*_{j_i}$ produces the reduction:

$$\hat{r}_{j_i}(\mathbf{d}_j) - \hat{r}_{j_i}(\mathbf{d}^*_{j_i}) \geq \zeta_{j_i} := \mu\hat{\pi}_{j_i}(\mathbf{d}_j)\min\left\{\frac{\hat{\pi}_{j_i}(\mathbf{d}_j)}{\kappa_{j_i}}, \Delta_{j_i}\right\}, \tag{6.6.12}$$

where $\mu \in (0,1)$.

Proof. We will consider the Stage II situation first. Firstly, note that an approximate Generalised Cauchy Point, $\mathbf{d}_j^{AGC}$, that satisfies the conditions (C.1) and (C.2) gives a model decrease of:

$$\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j^{AGC}) \geq \mu\hat{\chi}_j(\mathbf{d}_j,1)\min\left\{\frac{\hat{\chi}_j(\mathbf{d}_j,1)}{\kappa_j}, \Delta_j, 1\right\},$$

by Theorem 12.2.2 of Conn et al. (2000). This point $\mathbf{d}_j^{AGC}$ is simply a point on the projected-gradient path within a spherical trust region of radius Δ_j. Note that, by the properties of the ℓ_2 and ℓ_∞ norms, $||\mathbf{y}||_2 \geq ||\mathbf{y}||_\infty$ for all $\mathbf{y} \in \mathbb{R}^n$ and so $\{\mathbf{y} : ||\mathbf{y}||_2 \leq \Delta_j\} \subset \{\mathbf{y} : ||\mathbf{y}||_\infty \leq \Delta_j\}$. The algorithm in Section 6.4 finds the minimum along the projected-gradient path in the hyper-box trust region, $\mathbf{d}^*_j$, which includes all of the projected-gradient path within the corresponding feasible spherical trust region. Therefore, $\mathbf{d}^*_j$ satisfies

$$\begin{aligned}
& \hat{r}_j(\mathbf{d}^*_j) \leq \hat{r}_j(\mathbf{d}_j^{AGC}) \\
\Rightarrow \qquad & \hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}^*_j) \geq \hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j^{AGC}) \\
& \qquad \geq \mu\hat{\chi}_j(\mathbf{d}_j,1)\min\left\{\frac{\hat{\chi}_j(\mathbf{d}_j,1)}{\kappa_j}, \Delta_j, 1\right\} \\
& \qquad \geq \mu\hat{\pi}_j(\mathbf{d}_j)\min\left\{\frac{\hat{\pi}_j(\mathbf{d}_j)}{\kappa_j}, \Delta_j, 1\right\},
\end{aligned}$$

by the definition of $\hat{\pi}_j(\mathbf{d}_j) = \min\{1, \hat{\chi}_j(\mathbf{d}_j,1)\} \leq \hat{\chi}_j(\mathbf{d}_j,1)$. We also have that

$\hat{\pi}_j(\mathbf{d}_j) \leq 1$ and $\kappa_j \geq 1$ by definition (6.5.5). Therefore

$$\frac{\hat{\pi}_j(\mathbf{d}_j)}{\kappa_j} \leq \frac{1}{\kappa_j} \leq 1,$$

$$\Rightarrow \qquad \min\left\{\frac{\hat{\pi}_j(\mathbf{d}_j)}{\kappa_j}, \Delta_j, 1\right\} = \min\left\{\frac{\hat{\pi}_j(\mathbf{d}_j)}{\kappa_j}, \Delta_j\right\}.$$

Thus, we have the first result (6.6.11) that

$$\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j^*) \geq \mu\hat{\pi}_j(\mathbf{d}_j) \min\left\{\frac{\hat{\pi}_j(\mathbf{d}_j)}{\kappa_j}, \Delta_j\right\}.$$

The result for the inner loop (6.6.12) is shown using the same logic, but with an additional subscript i. □

6.6.3 The Inner Loop Can Always Find an Improvement

Having shown that the algorithm can produce an improvement in the meta-model, we are interested in how this translates into a reduction in the objective function, referring to the RC and SR tests. In the stochastic setting, this must account for the uncertainty in the objective and gradient estimators. Thus the statements are of a probabilistic nature, looking at the inner loop mechanism of STRONG.

First we put bounds on the error in the estimators during the inner loop.

Lemma 6.6.4. *Suppose that Assumptions AF.1, AF.2, AE.1 and AE.2 hold. Then, for any $\mathbf{d}_j \in \mathcal{D}$ and given j*

$$\Pr\left\{|\hat{g}_{j_i}(\mathbf{d}_j) - g(\mathbf{d}_j)| > \Delta_{j_i}^2 \;\; \textit{infinitely often}\right\} = 0,$$

$$\Pr\left\{||\widehat{\nabla} g_{j_i}(\mathbf{d}_j) - \nabla g(\mathbf{d}_j)||_2 > \Delta_{j_i} \;\; \textit{infinitely often}\right\} = 0.$$

The proof of the first statement is exactly the same as that of Lemma 2 in the STRONG paper (Chang et al., 2013), with the same assumptions on the growth of the number of observations with each inner loop, $N^c_{j_{i+1}} = \lceil (\gamma_0^{-4} + 1) N^c_{j_i} \rceil$. The authors also proved the second statement for OLS gradient estimators with orthogonal designs, based on the diagonal covariance matrix. Further work is required to extend this to a general OLS gradient estimator.

Using this result, we can show that for any centre point $\mathbf{d}_j$, the prediction error at the proposed point $\mathbf{d}^*_{j_i}$ will eventually shrink quadratically with the trust-region size.

Lemma 6.6.5. *Suppose that Assumptions AF.1, AF.2, AF.3, AE.1 and AE.2 hold. Then, for any* $\mathbf{d}_j \in \mathcal{D}$ *and given* j*,*

$$\Pr\left\{|\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})| > c\Delta^2_{j_i} \;\; \textit{infinitely often}\right\} = 0$$

for some constant $c > 0$*.*

Proof. The proof of this Lemma is very similar to the proof of Lemma 3 in Chang et al. (2013). The difference comes in a small adjustment because of the use of the ℓ_∞ norm changes the value of the constant c.

By the same arguments used in proving Lemma 6.6.4, we can also prove that

$$\Pr\left\{|\hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})| > \Delta^2_{j_i} \;\; \text{infinitely often}\right\} = 0. \tag{6.6.13}$$

Let $\mathbf{s}_{j_i} = \mathbf{d}^*_{j_i} - \mathbf{d}_j$ and note that

$$\begin{aligned} |\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})| &\le |\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})| + |\hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})| \\ &= \left|\hat{g}_{j_i}(\mathbf{d}_j) + \widehat{\nabla} g_{j_i}(\mathbf{d}_j)^T \mathbf{s}_{j_i} + \frac{1}{2}\mathbf{s}^T_{j_i} \widehat{H}_{j_i}(\mathbf{d}_j)\mathbf{s}_{j_i} - g(\mathbf{d}^*_{j_i})\right| \\ &\quad + |\hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})|, \end{aligned} \tag{6.6.14}$$

where we have used the definition of the quadratic meta-model $\hat{r}_{j_i}$, Equation (6.5.3). Using the second-order Taylor Theorem, we can say that, for some $\mathbf{y}_i$

$$g(\mathbf{d}^*_{j_i}) = g(\mathbf{d}_j + \mathbf{s}_{j_i}) = g(\mathbf{d}_j) + \nabla g(\mathbf{d}_j)^T \mathbf{s}_{j_i} + \frac{1}{2}\mathbf{s}_{j_i}^T H(\mathbf{y}_i)\mathbf{s}_{j_i},$$

where H is the true Hessian of g. Combining this with (6.6.14) we get

$$\begin{aligned}
&|\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})| \\
&\le \left|\hat{g}_{j_i}(\mathbf{d}_j) + \widehat{\nabla} g_{j_i}(\mathbf{d}_j)^T\mathbf{s}_{j_i} + \frac{1}{2}\mathbf{s}_{j_i}^T \widehat{H}_{j_i}(\mathbf{d}_j)\mathbf{s}_{j_i} - g(\mathbf{d}_j) - \nabla g(\mathbf{d}_j)^T\mathbf{s}_{j_i} - \frac{1}{2}\mathbf{s}_{j_i}^T H(\mathbf{y}_i)\mathbf{s}_{j_i}\right| \\
&\qquad + |\hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})| \\
&= \left|\hat{g}_{j_i}(\mathbf{d}_j) - g(\mathbf{d}_j) + \left(\widehat{\nabla} g_{j_i}(\mathbf{d}_j) - \nabla g(\mathbf{d}_j)\right)^T \mathbf{s}_{j_i} + \frac{1}{2}\mathbf{s}_{j_i}^T(\widehat{H}_{j_i}(\mathbf{d}_j) - H(\mathbf{y}_i))\mathbf{s}_{j_i}\right| \\
&\qquad + |\hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})| \\
&\le |\hat{g}_{j_i}(\mathbf{d}_j) - g(\mathbf{d}_j)| + \left|\left(\widehat{\nabla} g_{j_i}(\mathbf{d}_j) - \nabla g(\mathbf{d}_j)\right)^T \mathbf{s}_{j_i}\right| + \frac{1}{2}\left|\mathbf{s}_{j_i}^T(\widehat{H}_{j_i}(\mathbf{d}_j) - H(\mathbf{y}_i))\mathbf{s}_{j_i}\right| \\
&\qquad + |\hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})| \\
&\le |\hat{g}_{j_i}(\mathbf{d}_j) - g(\mathbf{d}_j)| + ||\widehat{\nabla} g_{j_i}(\mathbf{d}_j) - \nabla g(\mathbf{d}_j)||_2 \cdot ||\mathbf{s}_{j_i}||_2 + \frac{1}{2}||\widehat{H}_{j_i}(\mathbf{d}_j) - H(\mathbf{y}_i)|| \cdot ||\mathbf{s}_{j_i}||_2^2 \\
&\qquad + |\hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})|.
\end{aligned}$$

The final inequality holds through the Cauchy-Schwarz inequality and Operator Norm inequality. We know that $||\mathbf{s}_{j_i}||_\infty \le \Delta_{j_i}$, and so $||\mathbf{s}_{j_i}||_2 \le \sqrt{n}\Delta_{j_i}$. Furthermore, by AF.3 and Equation (6.5.4),

$$||\widehat{H}_{j_i}(\mathbf{d}_j) - H(\mathbf{y})|| \le ||\widehat{H}_{j_i}(\mathbf{d}_j)|| + ||H(\mathbf{y}_i)|| \le \kappa - 1 + \beta_H, \qquad \forall i.$$

Whilst $\mathbf{y}_i$ will be iteration dependent, the right-hand side of this inequality is not.

Therefore, we have that

$$|\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})| \leq |\hat{g}_{j_i}(\mathbf{d}_j) - g(\mathbf{d}_j)| + |\hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - g(\mathbf{d}^*_{j_i})| + \\ ||\widehat{\nabla} g_{j_i}(\mathbf{d}_j) - \nabla g(\mathbf{d}_j)||_2 \sqrt{n}\Delta_{j_i} + \frac{\kappa - 1 + \beta_H}{2} n\Delta^2_{j_i}.$$

Using Lemma 6.6.4 and Equation (6.6.13), we can see that, eventually, (that is, for sufficiently large i)

$$|\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})| \leq \Delta^2_{j_i} + \Delta^2_{j_i} + \sqrt{n}\Delta^2_{j_i} + \frac{\kappa - 1 + \beta_H}{2} n\Delta^2_{j_i}.$$

Taking $c = 2 + \sqrt{n} + n(\kappa - 1 + \beta_H)/2$ gives the result. □

The next Lemma shows that the inner loop will always find improvement if not at a first-order critical point, by finding a solution that passes both the RC and the SR tests (defined in Section 4.3.2).

Lemma 6.6.6. *Suppose that Assumptions AF.1, AF.2, AF.3, AF.4, AE.1 and AE.2 hold. Then for any $\mathbf{d}_j \in \mathcal{D}$ and given j, if $\pi(\mathbf{d}_j) > 0$, the algorithm can always find a satisfactory solution in iteration j.*

Proof. Take $\mathbf{d}_j \in \mathcal{D}$ such that $\pi(\mathbf{d}_j) = \xi > 0$. First we consider the RC test using ρ_{j_i} defined in Equation (4.3.12).

$$|\rho_{j_i} - 1| = \left| \frac{\hat{g}_{j_i}(\mathbf{d}_j) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})}{\hat{r}_{j_i}(\mathbf{d}_j) - \hat{r}_{j_i}(\mathbf{d}^*_{j_i})} - 1 \right| = \left| \frac{\hat{g}_{j_i}(\mathbf{d}_j) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{r}_{j_i}(\mathbf{d}_j) + \hat{r}_{j_i}(\mathbf{d}^*_{j_i})}{\hat{r}_{j_i}(\mathbf{d}_j) - \hat{r}_{j_i}(\mathbf{d}^*_{j_i})} \right|.$$

Using the definition of $\hat{r}_{j_i}$ in Equation (6.5.3), we have that $\hat{r}_{j_i}(\mathbf{d}_j) = \hat{g}_{j_i}(\mathbf{d}_j)$ and so

$$|\rho_{j_i} - 1| = \left| \frac{\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})}{\hat{r}_{j_i}(\mathbf{d}_j) - \hat{r}_{j_i}(\mathbf{d}^*_{j_i})} \right|.$$

By Lemma 6.6.3 and using $\kappa_{j_i} \leq \kappa$ for all i:

$$|\rho_{j_i} - 1| \leq \frac{|\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})|}{\mu\hat{\pi}_{j_i}(\mathbf{d}_{j_i}) \min\left\{\frac{\hat{\pi}_{j_i}(\mathbf{d}_{j_i})}{\kappa}, \Delta_{j_i}\right\}}. \tag{6.6.15}$$

Let us assume that the RC test is failed infinitely often, i.e., the algorithm gets stuck in this loop. Therefore the number of observations used for the gradient, $M_{j_i} \to \infty$, and the trust region shrinks, $\Delta_{j_i} \to 0$. Then, by the uniform convergence of $\hat{\pi}_{j_i}$ (Lemma 6.6.1):

$$\Pr\left\{\hat{\pi}_{j_i}(\mathbf{d}_j) < \xi/2 \text{ infinitely often}\right\} = 0,$$

and by the procedure of the inner loop

$$\Pr\left\{\Delta_{j_i} > \delta \text{ infinitely often}\right\} = 0, \qquad \forall\delta > 0.$$

Thus, for sufficiently large i, $\hat{\pi}_{j_i}(\mathbf{d}_j) > \xi/2$ and by Lemma 6.6.5

$$|\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})| < c\Delta^2_{j_i}.$$

Therefore, eventually,

$$\frac{|\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})|}{\mu\hat{\pi}_{j_i}(\mathbf{d}_{j_i}) \min\left\{\frac{\hat{\pi}_{j_i}(\mathbf{d}_{j_i})}{\kappa}, \Delta_{j_i}\right\}} \leq \frac{c\Delta^2_{j_i}}{\mu\frac{\xi}{2} \min\left\{\frac{\xi}{2\kappa}, \Delta_{j_i}\right\}} \qquad \text{and} \qquad \Delta_{j_i} < \delta.$$

$$\Rightarrow \Pr\left\{\frac{|\hat{r}_{j_i}(\mathbf{d}^*_{j_i}) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i})|}{\mu\hat{\pi}_{j_i}(\mathbf{d}_{j_i}) \min\left\{\frac{\hat{\pi}_{j_i}(\mathbf{d}_{j_i})}{\kappa}, \Delta_{j_i}\right\}} > \frac{2c\delta^2}{\mu\xi \min\left\{\frac{\xi}{2\kappa}, \delta\right\}} \text{ infinitely often}\right\} = 0, \tag{6.6.16}$$

for any $\delta > 0$.

We wish to choose δ such that:

$$\begin{aligned}
& \frac{2c\delta^2}{\mu\xi \min\left\{\frac{\xi}{2\kappa}, \delta\right\}} \leq 1-\eta_0 \\
\Rightarrow \quad & \frac{2c\delta^2}{\mu\xi\frac{\xi}{2\kappa}} \leq 1-\eta_0 \qquad \text{and} \qquad \frac{2c\delta^2}{\mu\xi\delta} \leq 1-\eta_0 \\
\Rightarrow \quad & \delta^2 \leq \frac{\mu\xi^2(1-\eta_0)}{4c\kappa} \qquad \text{and} \qquad \delta \leq \frac{\mu\xi(1-\eta_0)}{2c} \\
\Rightarrow \quad & \delta = \frac{\xi}{2}\min\left\{\sqrt{\frac{\mu(1-\eta_0)}{c\kappa}}, \frac{\mu(1-\eta_0)}{c}\right\}.
\end{aligned}$$

With this choice of δ, the right-hand side of condition (6.6.16) is less than $1-\eta_0$, and the left-hand side is greater than $|\rho_{j_i}-1|$ by (6.6.15). Thus, using (6.6.16) we have

$$\Pr\left\{|\rho_{j_i}-1| > 1-\eta_0 \text{ infinitely often}\right\} = 0$$

Using this, we can see that for sufficiently large i,

$$\begin{aligned}
& |\rho_{j_i}-1| \leq 1-\eta_0 \\
\Rightarrow \quad & 1-\rho_{j_i} \leq 1-\eta_0 \\
\Rightarrow \quad & \rho_{j_i} \geq \eta_0 \\
\Rightarrow \quad & \Pr\{\rho_{j_i} < \eta_0 \text{ infinitely often}\} = 0.
\end{aligned}$$

This contradicts the assumption that the RC test is failed infinitely often. Therefore, when i is sufficiently large, $\mathbf{d}^*_{j_i}$ will pass the RC test with probability 1.

Now consider the SR test. We know that the RC test has already been passed, so by Lemma 6.6.3

$$\hat{g}_{j_i}(\mathbf{d}_j) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i}) \geq \eta_0(\hat{r}_{j_i}(\mathbf{d}_j) - \hat{r}_{j_i}(\mathbf{d}^*_{j_i})) \geq \eta_0\zeta_{j_i}.$$

This implies that the test statistic

$$T^*_{j_i} = \frac{\hat{g}_{j_i}(\mathbf{d}_j) - \hat{g}_{j_i}(\mathbf{d}^*_{j_i}) - \eta_0^2\zeta_{j_i}}{S_{j_i}} \geq \frac{\eta_0(1-\eta_0)\zeta_{j_i}}{S_{j_i}},$$

where

$$S_{j_i}^2 = \frac{S^2(\mathbf{d}_j, N_{j_i})}{N_{j_i}} + \frac{S^2(\mathbf{d}_{j_i}^*, N_{j_i}^c)}{N_{j_i}^c}$$

is the sampled variance and $N_{j_i} \geq N_{j_i}^c$ is the accumulated number of replications at $\mathbf{d}_j$ until this point.

Suppose that the SR test is always failed: i.e., $T_{j_i}^* \leq t_{1-\alpha_j,\hat{\phi}}\ \forall i \in \mathbb{N}$. Then both $N_{j_i} \to \infty$ and $N_{j_i}^c \to \infty$, and so $S_{j_i}^2 \to 0$ with probability 1 and $t_{1-\alpha_j,\hat{\phi}} \to z_{1-\alpha_j}$ (the $1-\alpha_j$ quantile of the standard normal distribution). Furthermore, as the RC test is eventually always passed, Δ_{j_i} is bounded away from 0 with probability 1. Therefore.

$$\zeta_{j_i} = \mu \hat{\pi}_{j_i}(\mathbf{d}_{j_i}) \min\left\{\frac{\hat{\pi}_{j_i}(\mathbf{d}_{j_i})}{\kappa_{j_i}}, \Delta_{j_i}\right\}$$

is bounded away from 0. Thus, $T_{j_i}^* \to \infty$ as $i \to \infty$, so for i sufficiently large, $T_{j_i}^* > t_{1-\alpha_j,\hat{\phi}}$. This is a contradiction, and so $\mathbf{d}_{j_i}^*$ will pass the SR test for some i with probability 1. $\square$

6.6.4 Convergence of the Algorithm

Once it is established that the meta-model can be used to satisfactorally improve upon the current solution, the convergence proofs for deterministic trust-region algorithms go on to show that these individual iterations combine into a sequence with a first-order critical point as a limit point. STRONG and its extension rely on the inner loop, which characterises the state of information and the improvement found in the new solution. It is these quantities at the end of the iteration that influence convergence. Thus, we introduce new notation. If a quantity is primed, such as $\hat{\pi}_j'(\mathbf{d}_j)$ or Δ_j', this refers to its value at the end of the j^{th} iteration, including the inner loop if this is used.

The proof in the deterministic case starts by assuming that the sequence of solutions never reaches a first-order critical point and showing that when Δ'_j is sufficiently small, the RC test is always passed. Thus, $\{\Delta'_j\}_{j=0}^{\infty}$ is bounded away from 0. This implies that the sum of improvements is divergent (see the proof of Theorem 6.6.9). Due to stochasticity, this result is more complex for a simulation optimisation problem as $\{\Delta'_j\}_{j=0}^{\infty}$ is a stochastic sequence.

Conjecture 6.6.7. *Suppose that Assumptions AF.1, AF.2, AF.3, AF.4, AE.1 and AE.2 hold. Then, with probability 1:*

$$\liminf_{j\to\infty} \hat{\pi}'_j(\mathbf{d}_j) > 0 \qquad \Rightarrow \qquad \liminf_{j\to\infty} \Delta'_j > 0.$$

To establish the convergence requires this result or a similar one. For example, the weaker result limiting the speed of Δ'_j tending to 0:

$$\lim_{J\to\infty} \sum_{j=0}^{J} \Delta'_j = \infty \qquad \text{w.p. } 1 \tag{6.6.17}$$

could also be used for the proof. It is unclear at this point whether either property is true, and further work is required to establish or dismiss this conjecture.

As the algorithm does not deal with the true value of $\pi(\mathbf{d}_j)$, it is important to show that if the estimated criticality measure, $\hat{\pi}_j(\mathbf{d}_j)$, converges to 0, so does the truth. This is the purpose of the next result.

Lemma 6.6.8. *Suppose that Assumptions AF.1, AF.2, AF.3, AE.1 and AE.2 hold. If there is a subsequence of $\{\mathbf{d}_j\}_{j=0}^{\infty}$, denoted $\{\mathbf{d}_{j_k}\}_{k=0}^{\infty}$, such that $\lim_{k\to\infty} \hat{\pi}'_{j_k}(\mathbf{d}_{j_k}) = 0$*

with probability 1, then

$$\lim_{k\to\infty} \pi_{j_k}(\mathbf{d}_{j_k}) = 0$$

with probability 1.

Proof. Let M'_{j_k} be the total number of replications used for estimating the gradient, $\widehat{\nabla} g'_{j_k}(\mathbf{d}_{j_k})$, by the end of the inner loop of iteration j_k, and thus the number used to estimate $\hat{\pi}'_{j_k}(\mathbf{d}_{j_k})$. Note that $M'_{j_k} \geq M_{j_k}$, the number of observations used for the first estimate of $\pi(\mathbf{d}_{j_k})$. By the condition discussed in Section 6.5.2, $M_{j_k} \to \infty$ as $k \to \infty$. Therefore $M'_{j_k} \to \infty$. Then, by Lemma 6.6.1, $\hat{\pi}'_{j_k}(\mathbf{d}_{j_k})$ is a uniformly convergent estimator of $\pi(\mathbf{d}_{j_k})$ and so:

$$0 = \lim_{k\to\infty} \hat{\pi}'_{j_k}(\mathbf{d}_{j_k}) = \lim_{k\to\infty} \pi(\mathbf{d}_{j_k}),$$

with probability 1. □

The results discussed so far can now be combined into the final result of the sequence of solutions having a first-order critical point as a limit point with probability 1.

Theorem 6.6.9. *Suppose that Assumptions AF.1, AF.2, AF.3, AF.4, AE.1 and AE.2 hold. If the Algorithm has infinitely many successful iterations, then*

$$\liminf_{j\to\infty} \pi(\mathbf{d}_j) = 0 \tag{6.6.18}$$

with probability 1.

Proof. We analyse the outer loop, where $\mathbf{d}_j$ is random in the decision space. If $\liminf_{j\to\infty} \hat{\pi}'_j(\mathbf{d}_j) = 0$, then there exists a subsequence of solutions $\{\mathbf{d}_{j_k}\}_{k=0}^{\infty}$ such that

$\lim_{k\to\infty} \hat{\pi}'_{j_k}(\mathbf{d}_{j_k}) = 0$. Then, by Lemma 6.6.8 we have that $\lim_{k\to\infty} \pi(\mathbf{d}_{j_k}) = 0$ with probability 1 and so

$$\liminf_{j\to\infty} \pi(\mathbf{d}_j) = 0.$$

Therefore, it suffices to show that

$$\liminf_{j\to\infty} \hat{\pi}'_j(\mathbf{d}_j) = 0 \;\; w.p.1. \tag{6.6.19}$$

Suppose the contrary, that is, there exists a set of sample paths, $\Omega^\dagger$, such that each sample path $\omega \in \Omega^\dagger$ has a lower limit on the optimality measure, $\epsilon(\omega) > 0$:

$$\liminf_{j\to\infty} \hat{\pi}'_j(\mathbf{d}_j) = \epsilon(\omega).$$

and $\Pr(\Omega^\dagger) > 0$. The analysis focusses on sample paths in $\Omega^\dagger$. Then by Conjecture 6.6.7

$$\liminf_{j\to\infty} \Delta'_j > 0,$$

on $\omega \in \Omega^\dagger$. Then, for such sample paths, there exists a constant J_1 such that $\hat{\pi}'_j(\mathbf{d}_j) \geq \epsilon(\omega)$ and $\Delta'_{j_k} \geq \Delta_L$ for all $j \geq J_1$ for some Δ_L.

Now, let A_j be the event

$$\begin{aligned} A_j = &\left\{\omega \in \Omega^\dagger :\; g(\mathbf{d}_j) - g(\mathbf{d}_{j+1}) \leq \eta_0^2 \hat{\pi}'_j(\mathbf{d}_j) \min\{\Delta'_j, 1\} \text{ for stage I}\right\} \\ &\cup \left\{\omega \in \Omega^\dagger :\; g(\mathbf{d}_j) - g(\mathbf{d}_{j+1}) \leq \eta_0^2 \mu \hat{\pi}'_j(\mathbf{d}_j) \min\left\{\frac{\hat{\pi}'_j(\mathbf{d}_j)}{\kappa'_j}, \Delta'_j\right\}\right. \\ &\left. \qquad\qquad\qquad\qquad\qquad\qquad \text{for stage II or the inner loop}\right\}. \end{aligned}$$

This is the set of events that the SR test in iteration j has a Type I error, accepting a solution that does not satisfy the reduction criterion. The properties of the SR test guarantees that $\Pr(A_j) \leq \alpha_j$. Since $\sum_{j=0}^{\infty} \alpha_j < \infty$, by the first Borel-Cantelli Lemma (see for example Loève (1977) page 240), $\Pr(A_j$ infinitely often$) = 0$. Then, for all sample paths in $\Omega^{\dagger}$, there exists a constant $J_2 > 0$ such that

$$g(\mathbf{d}_j) - g(\mathbf{d}_{j+1}) > \eta_0^2 \hat{\pi}'_j(\mathbf{d}_j) \min\{\Delta'_j, 1\} \text{ for stage I or}$$
$$g(\mathbf{d}_j) - g(\mathbf{d}_{j+1}) > \eta_0^2 \mu \hat{\pi}'_j(\mathbf{d}_j) \min\left\{\frac{\hat{\pi}'_j(\mathbf{d}_j)}{\kappa'_j}, \Delta'_j\right\} \text{ for stage II or the inner loop,}$$

when $j \geq J_2$.

Let $J = \max\{J_1, J_2\}$. Then, for all sample paths in $\Omega^{\dagger}$,

$$\sum_{j=J}^{J_3} [g(\mathbf{d}_j) - g(\mathbf{d}_{j+1})] \geq \eta_0^2 \epsilon(\omega) N_I(J, J_3) \min\{\tilde{\Delta}, 1\}$$
$$+ \eta_0^2 \mu \epsilon(\omega) N_{II}(J, J_3) \min\left\{\frac{\epsilon(\omega)}{\kappa}, \Delta_L\right\} \tag{6.6.20}$$

where $N_I(J, J_3)$ and $N_{II}(J, J_3)$ are the number of successful iterations of stages I and II respectively between iterations J and J_3, $\tilde{\Delta}$ is the threshold trust-region radius to switch to stage II and κ comes from (6.5.4). As there are infinitely many successful iterations, $N_I(J, J_3) + N_{II}(J, J_3) \to \infty$ as $J_3 \to \infty$. Therefore, it is clear that, for almost all sample paths in $\Omega^{\dagger}$

$$\lim_{J_3 \to \infty} \sum_{j=J}^{J_3} [g(\mathbf{d}_j) - g(\mathbf{d}_{j+1})] = \infty.$$

However, note that

$$\sum_{j=J}^{\infty} [g(\mathbf{d}_j) - g(\mathbf{d}_{j+1})] \leq g(\mathbf{d}_J) - \liminf_{j \to \infty} g(\mathbf{d}_j) < \infty$$

because by AF.2, g is bounded below on $\mathcal{D}$. This is a contradiction. Therefore, equation (6.6.19) holds, completing the proof. □

Note the reliance on Conjecture 6.6.7 in the final proof. This could be adapted in Equation (6.6.20) if the weaker result of Equation (6.6.17) was guaranteed, achieving the same goal.

6.7 Conclusions and Further Work

This chapter has given a more detailed description of the extension to STRONG developed in this thesis to handle bound constraints and has presented a pathway for a potential proof of convergence of the algorithm. The adaptations involve using an alternative criticality measure, using the Generalised Cauchy Step and the application of Coordinate-Exchange Algorithms to build experimental designs from which to estimate the response surface. These were chosen in such a way to support convergence properties, in line with deterministic trust-region optimisation. The analyses in this chapter are designed to show the potential of the proposed algorithm regardless of the simulation model used in the multi-fidelity modelling approach to the ARP. To the best of our knowledge, no other work proposes the use of simulation directly in the search algorithm for the ARP.

The primary further work lies in two areas. Firstly, further experimentation on known functions with additional noise would enable us to build an understanding of the more general practical performance of the algorithm. Secondly, work is required to fill in the two blanks in the pathway to provable asymptotic behaviour. Whilst we believe that it is possible to prove the second statement of Lemma 6.6.4, it is currently unclear whether the conjecture that $\liminf_{j\to\infty} \hat{\pi}'_j(\mathbf{d}_j) > 0$ implies $\liminf_{j\to\infty} \Delta'_j > 0$

is true.

Improvements to the algorithm itself could be made. For example, our experimentation shows the steps in the linear model stages, Equation (6.4.1), are very sensitive to the sign of the gradient estimator, $\widehat{\nabla} g_j(\mathbf{d}_j)^k$, without accounting for it's magnitude. Thus, large steps can be taken in one component direction even with a very small gradient, potentially producing a poor proposal, which is not the case when using a Cauchy Point step (as in STRONG). One potential option would be to set $\widehat{\nabla} g_j(\mathbf{d}_j)^k$ to 0 if it is not significantly different from 0 to some confidence level greater than 0.5, making the probability of the third part of Equation (6.4.1) non-zero. In addition, during the inner loop, the design is augmented by creating a new design by CEA and appending this to the design matrix. An alternative would be to directly add points accounting for the design points already included in the design from previous iterations. This would improve the D-optimality measure of the whole design. One could also investigate the use of alternative gradient estimators where these are possible.

To go beyond bound constraints to general convex constraints would require some adaptations. The Generalised Cauchy Point method, as set out in Chapter 12 of Conn et al. (2000), could provide a framework for all problems with convex feasible regions. Much of the proof in Section 6.6 would remain valid, as the measure $\pi(\mathbf{d})$ is still applicable. The choice of step could be the approximate Generalised Cauchy Point, produced by Algorithm C.1, which would satisfy similar conditions to Lemmas 6.6.2 and 6.6.3. This would present challenges in calculating the projections and the choice of trust region. The main difficulty would be in finding an experimental design that could produce a good estimate of the gradient. Generalising to all convex constraints would mean the Coordinate-Exchange Algorithms could not be used, as constraints would involve relationships between (potentially all) variables. Standard exchange algorithms may be used in this context. An initial start may be to consider linear constraints and investigate a change of basis by a linear transformation to align the constraints with the region of interest. This would involve investigating the properties of a design under transformations.

Chapter 7

Empirical Evaluation of Simulation-Based Operational Decisions

7.1 Introduction

In its traditional setting, simulation has mostly been used for problems of system design, in making tactical or strategic decisions. For an aviation based example, Mujica Mota et al. (2017) use simulation to choose between airport apron and runway configurations to improve airport capacity. These are individual decisions, rather than repeated ones. Similarly, this has also been the focus of simulation optimisation techniques. The result is that, once a final decision has been made the simulation is unlikely to be re-used. The stakeholders then have to make the most of the decision without ongoing guidance from the simulation.

However, simulation has recently been used for repeated, operational level decisions, making use of up-to-date information from the system, such as the recovery from disruption considered in this thesis. An example of this is through symbiotic simulation (Fujimoto et al., 2002; Aydt et al., 2009), but it is not limited to this paradigm (Cheng, 2007). For repeated decisions, tracking the effectiveness of a simulation-based Decision Support System (DSS) becomes important, as consistent and systematically poor decisions due to a simulation not capturing critical system behaviour could be costly. Unlike the traditional case, however, the stake-holders will repeatedly observe what actually happened in the real system whilst the simulation is still in use. This gives some opportunity for a comparison between the prediction and the reality. However, this comparison is not as simple as asking "did this decision work out well?" Natural system variability may cause good solutions to lead to poor performance. If this does occur, it is incorrect to conclude that the solution was necessarily a poor decision.

Furthermore, each decision made will be under different circumstances, thus involving a series of heterogeneous decisions so that only one observation from the real system per decision is ever received. This means that recognising and diagnosing systematic error is a difficult problem. How can we tell if a single observation actually comes from the predicted distribution according to the simulation or not?

There are two primary components that contribute to a simulation-based DSS: the optimisation algorithm and the simulation itself. Both are important when considering the quality of the solutions found. Assessing the quality of the optimisation on a given real-time problem is very difficult as the optimal solution will not be known.

We will not look at this aspect of the problem. Instead we focus on a surrogate problem: whether the simulation-predicted performance matches up with the real-world performance. Even when the optimisation is perfect, if the simulation model is poor the optimisation is solving the wrong problem, probably leading to bad performance.

This may seem similar to the issue of model validation, an important part of building confidence in the model. However, validation of an online stochastic simulation is difficult. As pointed out in Section 5.1 of Nelson (2013), validation can only be made against an existing real system, and even this does not guarantee the validity for the conceptual systems simulated in the DSS. Oakley et al. (2020) acknowledge that validity depends on the initial system state. This could involve a combinatorially large set of possibilities, for which it is not possible to validate in all scenarios. If the system processes remain unchanged irrespective of the scenario, the Δ-method proposed by Oakley et al. (2020) can be used by assessing changes in state. If this is not the case, then the simulation will be unvalidated for many of the possible situations. Furthermore, as the intention is that the simulation model is reused over a long period of time, the processes in the real-world system it is modelling may evolve over time, producing a divergence between the system and the model.

Section 5.1 of Nelson (2013) categorises two errors that could occur. One is input uncertainty. The use of symbiotic simulation to bring in up-to-date data for distribution fitting and the use of Machine Learning based methods for calibration (as mentioned in Onggo et al. (2018)) can help to reduce this issue without major re-modelling. The second error, modelling error, is much harder to address. A particular scenario may highlight the importance of a process that did not feature strongly in the settings used for validation, leading to a large discrepancy in the simulation predictions.

This thesis has been focussed on the Aircraft Recovery Problem. In principle, the techniques proposed could form the basis of a symbiotic simulation DSS triggered when a disruption occurs by matching the initial conditions of the physical system. In this case, the issue of evaluating the decision certainly plays a role. No two disruptions will be identical, and there are a number of sources of natural variability within the industry. Furthermore, a model is likely to be validated on regular operations, but used when things do not go according to plan.

This chapter will focus on exploring questions related to evaluating a simulation-based DSS. To our knowledge, this area has not been previously studied and yet is an important issue when using simulation for operational decisions. Section 7.2 describes the problem and introduces some notation. Section 7.3 discusses an initial method for detecting systematic discrepancies between the simulation and the real world. Some experiments involving the airline disruption problem are presented in Section 7.4. Section 7.5 discusses some of the wider issues related to this problem before some concluding remarks and ideas for further work in Section 7.6.

7.2 Problem Definition and Notation

Suppose that we have a symbiotic simulation system, involving a real-world, or physical, system and a simulation model used as a DSS. Let $\{T_j = [t_{j1}, t_{j2}]\}_{j=1}^{J}$ be a series of intervals over which the simulation is used to make decisions. Here t_{j1} is the point in time at which the DSS is triggered, and t_{j2} is the time horizon for the period of interest. An example of this could be an airline disruption occurring at time t_{j1} and the end of the recovery window being t_{j2}. Over the j^{th} time period, T_j, a key performance measure in the real world will take some value, R_j. In the airline disruption problem, this could be disruption cost. R_j can be considered a random variable coming from a distribution dependent on the time interval T_j, the state of the system at t_{1j}, Ψ_j, which may include the history of the physical system, and the decision implemented $x \in \mathcal{X}_j$:

$$R_j \sim F\left(\,\cdot\, | \Psi_j, T_j, x \right). \tag{7.2.1}$$

Note that $F\left(\,\cdot\, | \Psi_j, T_j, x \right)$ is unknown and only a single observation of R_j will ever be seen.

Suppose we are using a simulation model and optimisation techniques to make our decision at time t_{j1}. The same performance measure according to the simulation is

$$S_j \sim G(\,\cdot\, | \hat{\Psi}_j, T_j, x). \tag{7.2.2}$$

Again the distribution of S_j is dependent on T_j, the decision x and the simulation state $\hat{\Psi}_j$, which will almost always be an incomplete description of the real system state. $G(\,\cdot\, | \hat{\Psi}_j, T_j, x)$ is also unknown. The aim of the simulation optimisation is to find some $x_j \in \mathcal{X}_j$ that 'optimises' the system in some way; this is often defined by

$$x_j = \arg\min_{x \in \mathcal{X}_j} \left\{ H_{G(\,\cdot\, | \hat{\Psi}_j, T_j, x)}[S_j] \right\} \tag{7.2.3}$$

where H is an operator with respect to the simulation distribution. Equation (7.2.3) defines a 'good' solution. A common choice of H is the expectation of S_j,

$$H_{G(\,\cdot\,|\hat{\Psi}_j,T_j,x)}[S_j] = \mathbb{E}\left[S_j\right].$$

The process involves simulating the period T_j multiple times under a variety of decisions x to estimate $H_{G(\,\cdot\,|\hat{\Psi}_j,T_j,x)}[S_j]$, often with a sample average, and propose the solution that minimises the estimate. This assumes that $G(\,\cdot\,|\hat{\Psi}_j, T_j, x)$ is a 'sufficiently good' approximation of $F\,(\,\cdot\,|\Psi_j, T_j, x)$.

Let x_j be the decision suggested by the DSS. This is then implemented in the real-world system over T_j, and we observe R_j at or after t_{j2}. Due to natural variability inherent in the physical system, it is possible that R_j will represent poor performance by coming from unfavourable portions of the support of $F\,(\,\cdot\,|\Psi_j, T_j, x_j)$, even if x_j is a good solution. Therefore, a single observation is not sufficient to evaluate the efficacy of the procedure.

Suppose that the DSS is used repeatedly across the sequence of intervals $\{T_j\}_{j=1}^{J}$ to propose solutions. The result will be a series of real-world observations, $\{R_j\}_{j=1}^{J}$, each coming from a different distribution. One question that arises is: how can we use these observations to see if the DSS is providing good predictions of solution performance?

7.3 Initial Transformation Approach

This section proposes a basic approach to the question by transforming $\{R_j\}_{j=1}^{J}$, aiming for a set of identically distributed random variables. To do so, it is assumed that the intervals $\{T_j\}_{j=1}^{J}$ are such that the performance measures of each interval are conditionally independent given the current system state. This may or may not be true for overlapping intervals, which is an area for future investigation.

Whilst $G(\,\cdot\,|\hat{\Psi}_j, T_j, x_j)$ is unknown, it can be approximated to arbitrary precision by simulating the consequences of decision x_j many times and using the empirical distribution. This can be done during the interval T_j. Using N replications of the simulation to obtain independent and identically distributed (i.i.d.) samples S_j^n, $n = 1, 2, ..., N$, $G(s|\hat{\Psi}_j, T_j, x_j)$ can be estimated using the Empirical Cumulative Distribution Function (ECDF)

$$G(s|\hat{\Psi}_j, T_j, x_j) \approx \hat{G}_j(s, N) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}(S_j^n \leq s), \tag{7.3.1}$$

where $\mathbb{I}(\cdot)$ is an indicator function. Note that, by the Glivenko-Cantelli theorem (see for example Loève (1977) page 20),

$$\Pr\left(\lim_{N\to\infty} \sup_{s\in\mathbb{R}} \left|\hat{G}_j(s, N) - G(s|\hat{\Psi}_j, T_j, x_j)\right| = 0\right) = 1. \tag{7.3.2}$$

That is, with probability 1, $\hat{G}_j(s, N) \to G(s|\hat{\Psi}_j, T_j, x_j)$ uniformly in s as $N \to \infty$.

To enable the real-world observation to be compared with the simulation predictions, we transform R_j using the Probability Integral Transformation based on the ECDF resulting from the intensive simulation of x_j. Define the transformed observations as

$$U_j = \hat{G}_j\left(R_j, N\right). \tag{7.3.3}$$

If R_j does come from $G(\,\cdot\,|\hat{\Psi}_j, T_j, x_j)$, then the properties of the CDF state that

$$U_j \approx G(R_j|\hat{\Psi}_j, T_j, x_j) \sim U(0, 1). \tag{7.3.4}$$

That is, $\{U_j\}_{j=1}^{J}$ are a set of (approximately) i.i.d. uniform random variables.

However, if the performance is systematically different from the simulation predictions due to the model not capturing an important real-world process, then $R_j \not\sim G(\,\cdot\,|\hat{\Psi}_j, T_j, x_j)$, so $\{U_j\}_{j=1}^{J}$ will not be uniformly distributed.

This can be tested using a goodness-of-fit hypothesis test:

$$H_0 : \; U_j \sim U(0,1) \qquad \text{versus} \qquad H_1 : \;\; U_j \not\sim U(0,1).$$

There are several possible test statistics that could be used to detect any departures from uniformity, such as the Kolmogorov-Smirnov test (see, for example, Thas (2010) pages 123-129). The choice of test statistic is explored in Section 7.4.4. If the null hypothesis is rejected, it would provide evidence towards the simulation requiring an update or modification before further use.

For example a possible departure from uniformity might be R_j regularly being above the 0.8 quantile of $G(\,\cdot\,|\hat{\Psi}_j, T_j, x_j)$. In this case the transformation (7.3.3) would lead to a high proportion of U_j being above 0.8, indicating that the simulation is underestimating the performance measure, which should be detected by a goodness-of-fit test.

Algorithm 7.1 summarises the proposed method.

7.4 Evaluation

To investigate the approach, we need a way to create real-world observations. As we do not have access to an airline's operations, for the purposes of this investigation, the simulation model is modified to create a range of proxy real worlds. The modifications are to characteristics of the service times on the runways, and are chosen so that the

Algorithm 7.1 Framework for Detecting Poor Simulation Prediction

```
1: Set significance level of hypothesis test, $\alpha$, and the number of past decisions to
   include in the test, $J$
2: for $j = 1, ..., J$ do
3:     At $t_{j1}$, DSS triggered and solution $x_j$ found
4:     Implement $x_j$ in real world
5:     During $T_j$, perform $N_j$ simulations of $x_j$ and calculate $\hat{G}_j(\,\cdot, N_j)$
6:     At $t_{j2}$, observe $R_j$
7:     $U_j \leftarrow \hat{G}_j(R_j, N_j)$
8: end for
9: Set up hypothesis test
       $$H_0: \ U_j \sim U(0,1) \qquad \text{versus} \qquad H_1: \ U_j \nsim U(0,1).$$
10: Test using a goodness-of-fit test, calculate the $p$-value $p_J$
11: if $p_J < \alpha$ then
12:     Investigate to identify issue in simulation
13: end if
```

scale of the difference is controllable.

7.4.1 Proxy Real Worlds

Whilst the model described in Section 4.2 used a deterministic model for service times at airport runway queues, in reality there is variability. Aircraft approaches and take-offs can be very tightly controlled, with minimal variation in the actual spacing, bar communication delays in take-off (Stamatopoulos et al., 2004), but there are different rules for separation between different aircraft types due to wake turbulence (ICAO, 2016). Given an order of take-offs or landings, the variability is very small. However, as the sequence is often random depending on the order of arrivals, many analytical runway queueing models, see for example Shone et al. (2019), use Erlang distributions to represent the uncertainty. The ICAO rules dictate a minimum separation that must be applied (ICAO, 2016), suggesting that if an aircraft is too close to the preceding aircraft, it will be told to reduce speed to maintain the minimum separation.

To incorporate these mechanisms into the model, the simulation is altered so that the i^{th} service time is modelled as

$$Q_i = \max\{s(t), Y_i\} + w(t). \tag{7.4.1}$$

Here $s(t)$ is a time dependent minimum allowable aircraft separation, increasing overnight (11pm-6am) due to noise pollution restrictions and $w(t)$ represents an increase in the separation due to poor meteorological conditions (see, for example, ICAO (2016)). Let

$$Y_i = \max\{s_{\min}, Z_i\} \tag{7.4.2}$$

where Z_i has a marginal Erlang distribution with k phases and $s_{\min} = \min_t\{s(t)\}$ is the lowest legal separation time. The marginal CDF of the process Y_i is therefore:

$$F_Y(y) = F_Z(y)\mathbb{I}(y \geq s_{\min}).$$

Air Traffic Controllers often wish to sequence similar sized aircraft together, as this can increase the runway capacity (Stamatopoulos et al., 2004). This leads to a dependence between successive service times. To model this, we allow the sequence Y_i to be autocorrelated, with a lag-1 autocorrelation ρ. When $\rho = 0$, Equation (7.4.2) is used with Z_i simply sampled from an Erlang distribution. When $\rho > 0$, we work with the CDF of Y_i directly, as the minimum alters the autocorrelation. This is simulated using the order 1 auto-regressive-to-anything (ARTA(1)) process (Cario and Nelson, 1996).

The ARTA(1) process transforms the order 1 auto-regressive process (AR(1)) using the standard normal CDF, Φ, and the inverse CDF of the marginal distribution of the desired process. As the autocorrelation is not guaranteed to be preserved by this transformation, one needs to calculate the base autocorrelation of the AR(1) process, r, to give the desired autocorrelation, ρ. One can use numerical methods for this, but here we use the simulation matching described for the normal-to-anything (NORTA) transformation for bivariate random variables in Nelson (2013, page 148).

For each k, the mean for the Erlang distribution is chosen such that the mean of the resulting service times during the day in good weather ($w(t) = 0$) is 78 seconds. Whilst this value is lower than those considered in some aircraft sequencing studies such as Bennell et al. (2017) or simulations of individual airports such as Mori

(2015), it was chosen to produce a reasonable level of congestion across airports using segregated operations (an assumption of the model discussed in Section 4.2). The restriction to two runways is lifted during busy periods (6am-3pm and 4pm-9pm) at the largest airports, namely Paris Charles de Gaulle Airport (CDG), Amsterdam Airport Schiphol (AMS) and Adolfo Suárez Madrid–Barajas Airport (MAD).

Twelve proxy real world systems are considered, defined by combinations of the variance and autocorrelation of the service time models. Specifically:

- variance of the service times at airport runways, using the maximum of an Erlang distribution with $k = 1000$, 100, 10 and 2 phases and $\min_t\{s(t)\} = 1$ minute;
- lag-1 autocorrelation of the service times at airport runways, using $\rho = 0$, 0.5 and 0.9.

The proxy real world with k phases and lag-1 autocorrelation ρ is denoted as (k, ρ) and a cost observation from (k, ρ) with a superscript k and ρ. The $(1000, 0)$ system is used as the model for the simulation optimisation. This has very low variance (coefficient of variation 0.001) and independent service times at the runways. As k gets smaller and ρ gets larger, the simulation deviates more from the proxy real worlds. Thus, we expect that detection should become easier. Other than the service times for the runway queues, the simulations are identical. Whilst there are many other possible and realistic deviations, we focussed solely on the service times as results from queueing theory provide an expectation of how changes in queueing behaviour will effect the system.

7.4.2 Hypothesis Tests for Uniformity

The choice of goodness-of-fit test may have a large impact on whether any differences between the real world and the simulation are detected. In this case, it is the power of the tests that is important, as we want to be able to detect a difference if it exists. A Type I error is not as important. There are many tests for uniformity, mostly designed for a specific alternative distribution. Quesenberry and Miller (1977) and Miller and Quesenberry (1979) performed power studies across a range of goodness-of-fit tests with four families of alternative distributions. The results gave a most powerful test for each distribution, as well as a set of generally good performing tests.

In the setting described above, one class of concerning deviations between the simulation and the real world are those that lead to an inconsistent underestimation of cost (or an overestimation of profit). That is, the solution does not perform as well as it was predicted, with an unpredictable bias that cannot be corrected for. This matches the first family of alternative distributions in Quesenberry and Miller (1977), where the mass of the distribution is concentrated towards one end of the unit interval. For this family, the Anderson-Darling test (see, for example, Thas (2010) pages 129-141) achieved the highest power. Thus, we shall include the Anderson-Darling (A-D) test in our experimentation.

Quesenberry and Miller (1977) and Miller and Quesenberry (1979) also find three other tests that are good all-round performers. These are the Watson test (W) (see, for example, Thas (2010) pages 142-144) and the Neyman-Smooth tests with 2 and 4 degrees (NS2 and NS4 respectively) (see, for example, Thas (2010) pages 82-95). As

Table 7.4.1: Summary of each problem.

Problem	Cause	Disrupted Aircraft	Time
Day 1	aircraft	1	16:00
Day 2	weather	2	9:40
Day 4	aircraft	1	13:20
Day 5	weather	2	10:40
Day 6	weather	5	10:10

we do not know for certain that discrepancies will lead to alternatives with a one-sided concentration we include these three tests in the experimentation.

For comparison purposes, we also include the Kolmogorov-Smirnov test (K-S) and the Cramer-von Mises test (C-vM) (see, for example, Thas (2010) pages 129-141).

7.4.3 Experimental Setup

For this experiment, we considered five problems originally discussed in Section 5.6. These are based on a fleet of 116 aircraft and their descriptions are summarised in Table 7.4.1. Day 3 is excluded as the IP solution contains no delays, so there is no need for simulation optimisation. The 'Cause' column refers to either a technical problem grounding an aircraft (labelled 'aircraft') or poor weather conditions leading to heavy airport congestion. Certain types of disruptions or system states may be more sensitive to a deviation than others, which would alter both the need and the ability to detect a deviation. For example, one would expect that a disruption caused by airport congestion would be quite sensitive to deviations in queueing behaviour. Therefore, testing the proposed method on a sequence of disruptions with different causes, scales and settings could create too many moving parts in the experiment to allow for a thorough analysis of its ability to detect differences. To control for this variable sensitivity, rather than using a sequence of distinct disruptions, our 'sequence' consists of the same disruption solved multiple times. Effectively, an identical disruption occurs on each day of the sequence. As the solution depends in part on the starting seed of the optimisation algorithm, there will be several distinct solutions with different ECDFs. This allows us to partly mimic the real world.

The sequence of intervals $\{T_j\}_{j=1}^J$ will be a repeat of the interval T_1. For each, the multi-fidelity optimisation procedure described in Chapters 3 and 4 is used to generate a solution $x_j = (\mathbf{x}, \mathbf{d}_j)$, which is simulated $N = 1000$ times under the $(1000, 0)$ system to produce an ECDF $\hat{G}_j$. Each solution is simulated under each proxy real world to obtain a sequence of real-world observations $\{R_j^{k\rho}\}_{j=1}^J$. These are transformed using Equation (7.3.3) to obtain the supposedly uniform random variables $\{U_j^{k\rho}\}_{j=1}^J$, on which we perform each goodness-of-fit test for uniformity. The full procedure is summarised in Algorithm 7.2.

As the simulation produces random outputs, the rejection of the null hypothesis is also random. Thus, performing one test does not give a meaningful indication of the behaviour of the test. The M repetitions of steps 7-13 in Algorithm 7.2 results in a set of M different p-values for the system (k, ρ), each coming from a possible realisation of the proxy real world. This allows us to look at the rejection rate for the system (k, ρ), helping to account for the dependence on the starting seed in our

Algorithm 7.2 Experimental Procedure

```
1: for $j = 1$ to $J(=25)$ do
2:     Solve the disruption to get solution $x_j$
3:     Perform $N = 1000$ simulations of $x_j$ using CRNs to produce ECDF $\hat{G}_j$
4: end for
5: for Each Proxy Real World System $(k, \rho)$ do
6:     for $m = 1$ to $M(=100)$ do
7:         for $j = 1$ to $J$ do
8:             Simulate $x_j$ once under $(k, \rho)$ to get one cost observation $R_j^{k\rho}$
9:             Transform $R_j^{k\rho}$ using Equation (7.3.3): $U_j^{k\rho} \leftarrow \hat{G}_j(R_j^{k\rho}, N)$
10:        end for
11:        Result is two sequences, one of real-world observations, $\{R_j^{k\rho}\}_{j=1}^{J}$, and their
           transformed values $\{U_j^{k\rho}\}_{j=1}^{J}$.
12:        Perform the hypothesis test:

               $H_0 : U^{k\rho} \sim U(0,1) \qquad \text{versus} \qquad H_1 : U^{k\rho} \nsim U(0,1)$

           using each goodness-of-fit test for uniformity on the set $\{U_j^{k\rho}\}_{j=1}^{J}$.
13:        Note $p-$value of test
14:    end for
15:    Calculate the empirical power of test in setting $(k, \rho)$
16: end for
```

proxy real world.

7.4.4 Initial Results

This section presents and discusses the results from the five experiments. In each case, the simulation optimisation has been used $J = 25$ times to give 25 solutions. We take $M = 100$ observations of each solution for each system (k, ρ). Days 1, 2 and 6 are discussed in detail. Results for Days 4 and 5 are similar, and the plots are shown in Appendix D.

Figure 7.4.1 shows the P-P plots of $\{U_j^{k\rho}\}_{j=1}^{J}$ for each system (k, ρ) for the Day 1 disruption. Each line represents the ECDF of one set of $\{U_j^{k\rho}\}_{j=1}^{J}$, produced by performing steps 7-10 of Algorithm 7.2 once. Each goodness-of-fit test measures how different a line is from the diagonal. If the simulation is systematically underestimating (overestimating) the cost of a solution, one would expect most of the lines to lie below (above) the diagonal. The plots seem to suggest that the cost of each solution is not overly sensitive to the deviations considered here. The first appreciable difference (by eye) appears to be the (10,0.9) system. Here there is a clear tendency for the lines to be below the diagonal, suggesting that the simulation underestimates the cost. A similar observation can be made from the (2,0.5) system. In the (2,0.9) system, the most extreme case, the discrepancy is clear for all 100 lines.

The effect of autocorrelation seems to increase as k decreases and the variance gets larger. For high k, the deviation from the mean is very small, meaning that even a sequence of relatively 'high' service times due to autocorrelation make little difference compared with the uncorrelated case. As the variance increases, the longer service

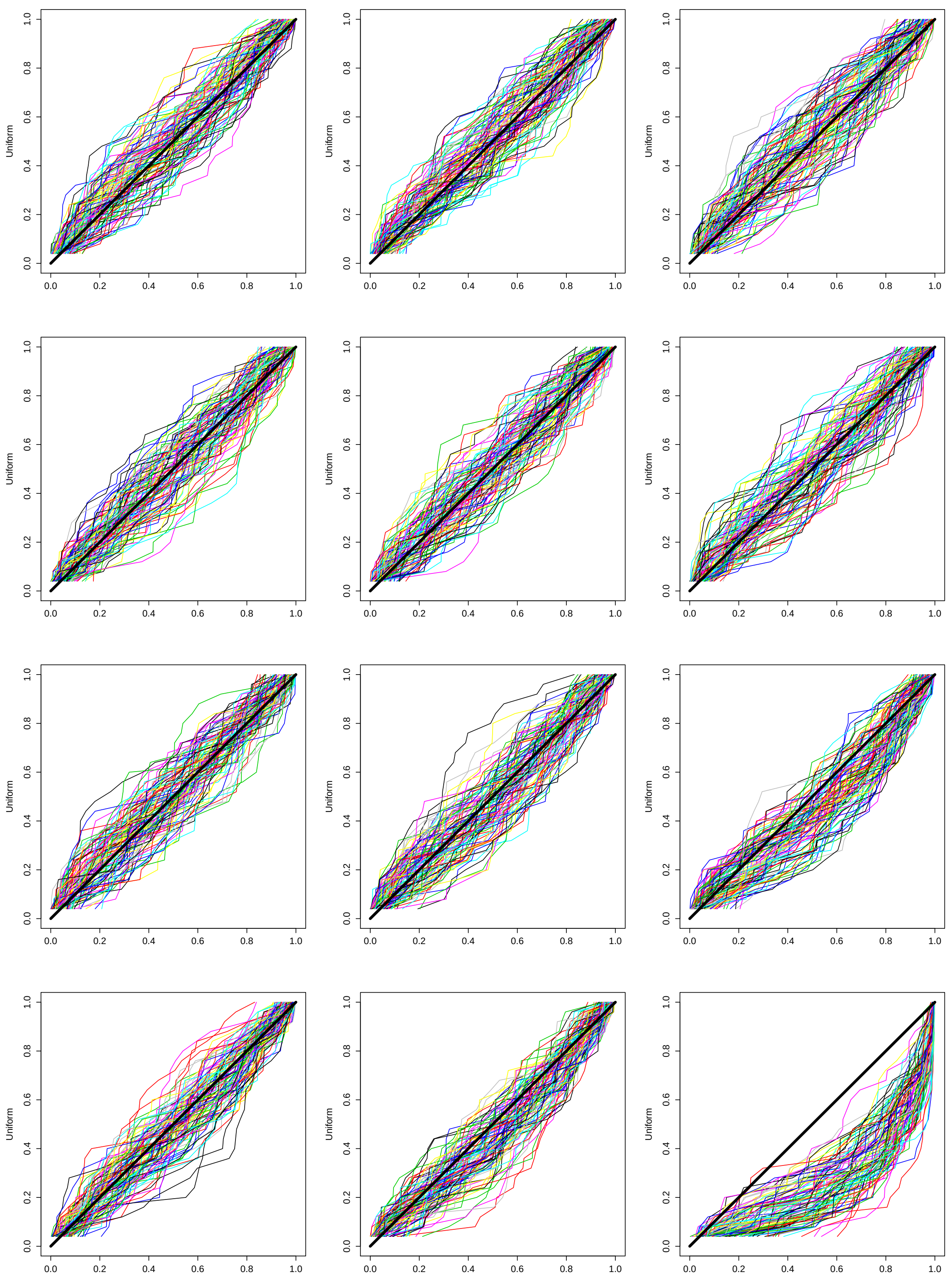

Figure 7.4.1: Day 1 P-P plots of the sets $\{U_j^{k\rho}\}_{j=1}^J$, for each (k, ρ). Rows are k=1000, 100, 10, 2, columns are $\rho = 0, 0.5, 0.9$.

times have more effect on the waiting times, and a sequence of longer service times becomes more likely when autocorrelation is included.

The sensitivity of a system to such deviations plays an important role in whether or not a difference is detected. To quantify this, we define sensitivity to the deviation (k, ρ) as

$$\mathcal{S}_{k\rho} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \frac{\bar{R}^{k\rho}(x_j) - \bar{R}^{1000,0}(x_j)}{\bar{R}^{1000,0}(x_j)}. \tag{7.4.3}$$

here $\mathcal{J}$ is the set of distinct solutions in $\{x_j\}_{j=1}^{J}$ and $\bar{R}^{k\rho}(x_j)$ is the mean of 1000 simulations of solution x_j under the system (k, ρ). The use of CRN reduces the uncertainty in $\mathcal{S}_{k\rho}$. The widths of the 95% confidence intervals of $\mathcal{S}_{k\rho}$ are generally one order of magnitude smaller than $\mathcal{S}_{k\rho}$, with the widest having a half-width of 2.5%. For Day 1, only three systems had $\mathcal{S}_{k\rho}$ greater than 1%, $\mathcal{S}_{10,0.9}$=2.9%, $\mathcal{S}_{2,0.5}$=2.4% and $\mathcal{S}_{2,0.9}$=22%.

Figure 7.4.2 shows the rejection frequency at the 5% significance level for each statistical test in the (k, ρ) system, i.e., how many lines from Figure 7.4.1 did the goodness-of-fit test reject. It is an estimate of the test's power. The results show fairly low powers for all but the (2,0.9) system. In general the Anderson-Darling test performs the best. This is consistent with the results of Quesenberry and Miller (1977), as the distributions of $U^{k\rho}$ are concentrated towards one end of the [0,1] interval. The power from the (10,0.9) and (2,0.5) systems are disappointing, as we would hope the test would pick out deviations that lead to a visible discrepancy in the P-P plots.

The story for Day 2 is very similar. Figures 7.4.3 and 7.4.4 show the P-P plots

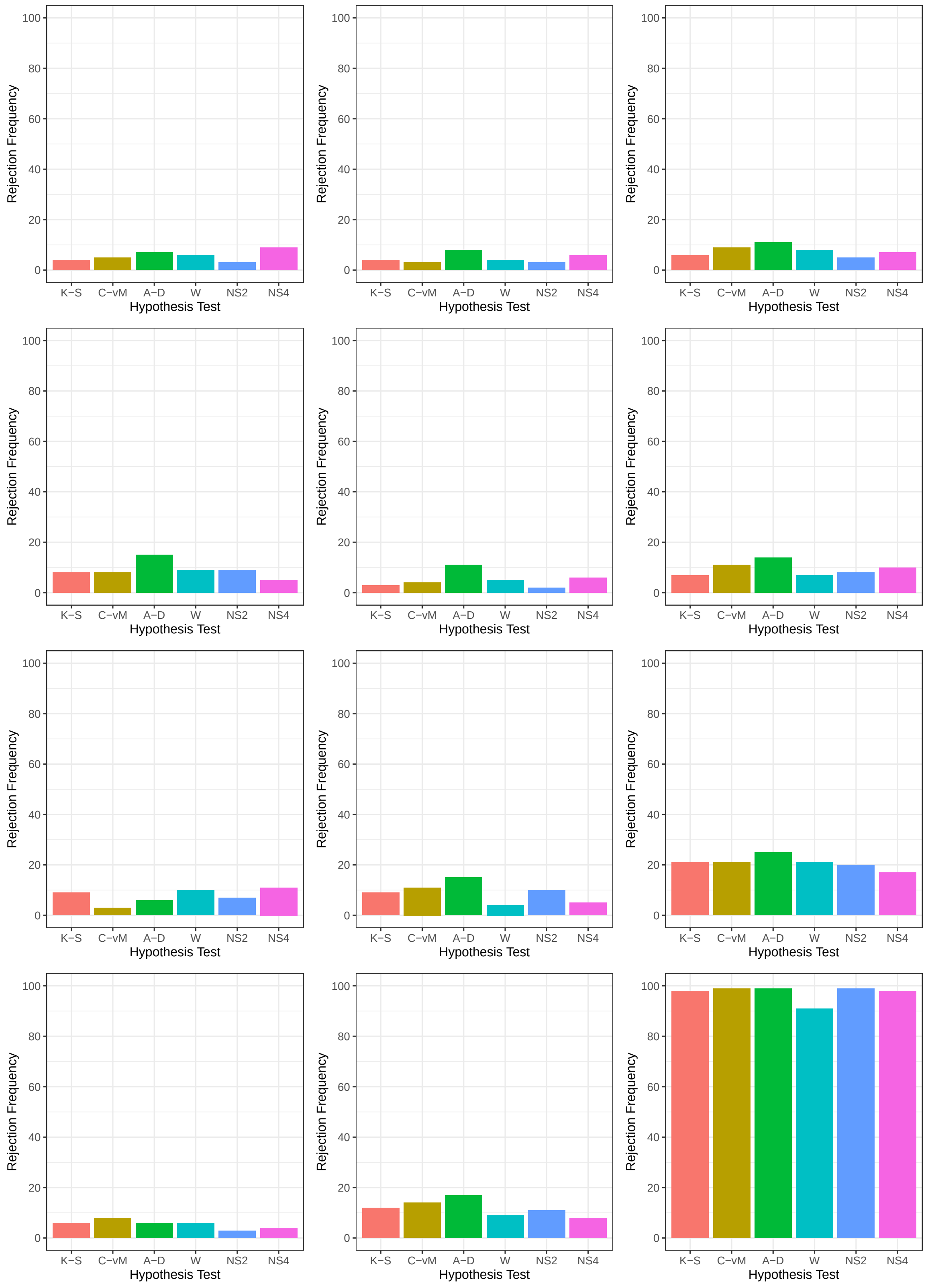

Figure 7.4.2: Number of rejections for each hypothesis test in each system (k, ρ) in the Day 1 disruption. Rows are $k = 1000, 100, 10, 2$, columns are $\rho = 0, 0.5, 0.9$.

and rejection frequencies. This disruption is based around congestion at AMS, a large airport. Furthermore, it occurs earlier in the day, meaning that the knock on effects of the disruption are larger and so one would expect the cost to be more sensitive to queueing behaviour. On face value, the results appear to agree and the visible difference in the (10,0.9), (2,0.5) and (2,0.9) systems are greater, with $\mathcal{S}_{2,0.9} = 73\%$. However, even here the sensitivity $\mathcal{S}_{k\rho}$ only exceeds 1% in the five most extreme deviations.

Figure 7.4.4 shows the power to have improved over the Day 1 problem, but it is still quite low, even for the (2,0.5) system. This time the Anderson-Darling test is never beaten in rejection frequency, again consistent with the observations of Quesenberry and Miller (1977), suggesting that general underestimation is the problem. The power has improved in the (10,0.9) system due to an increased sensitivity of 8.8%.

Day 4 and Day 5 follow a similar pattern to Days 1 and 2 respectively. The results of these are shown in Figures D.1 to D.4 of Appendix D. The Day 6 results in Figures 7.4.5 and 7.4.6, however, show a slightly different pattern. The lines in the (10,0.9), (2,0.5) and (2,0.9) systems often begin above the diagonal before shifting to below, quite a different shape from the (2,0.9) systems on the other days, which show a strong tendency to stay below the diagonal. This suggests a concentration at both ends of the [0,1] interval, more akin to the third and fourth alternative distributions in Miller and Quesenberry (1979). Their results suggest that the Neyman Smooth tests outperform the Anderson-Darling test. This is mirrored in Figure 7.4.6.

The full results across all five days of sensitivity against power are summarised in Figures 7.4.7 and 7.4.8. It is clear that for systems with a sensitivity of more than

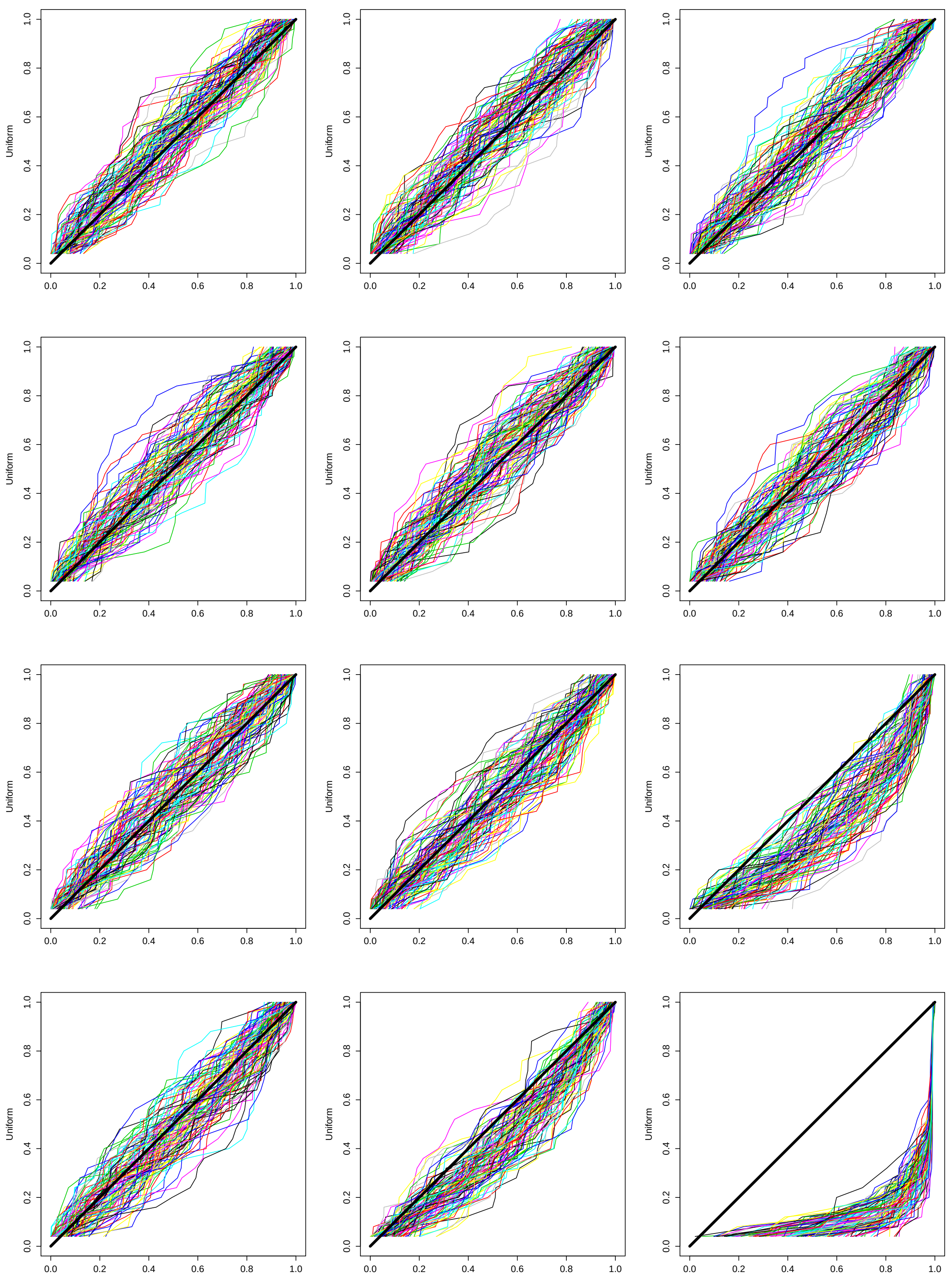

Figure 7.4.3: Day 2 P-P plots of the sets $\{U_j^{k\rho}\}_{j=1}^{J}$, for each (k, ρ). Rows are k=1000, 100, 10, 2, columns are $\rho = 0, 0.5, 0.9$.

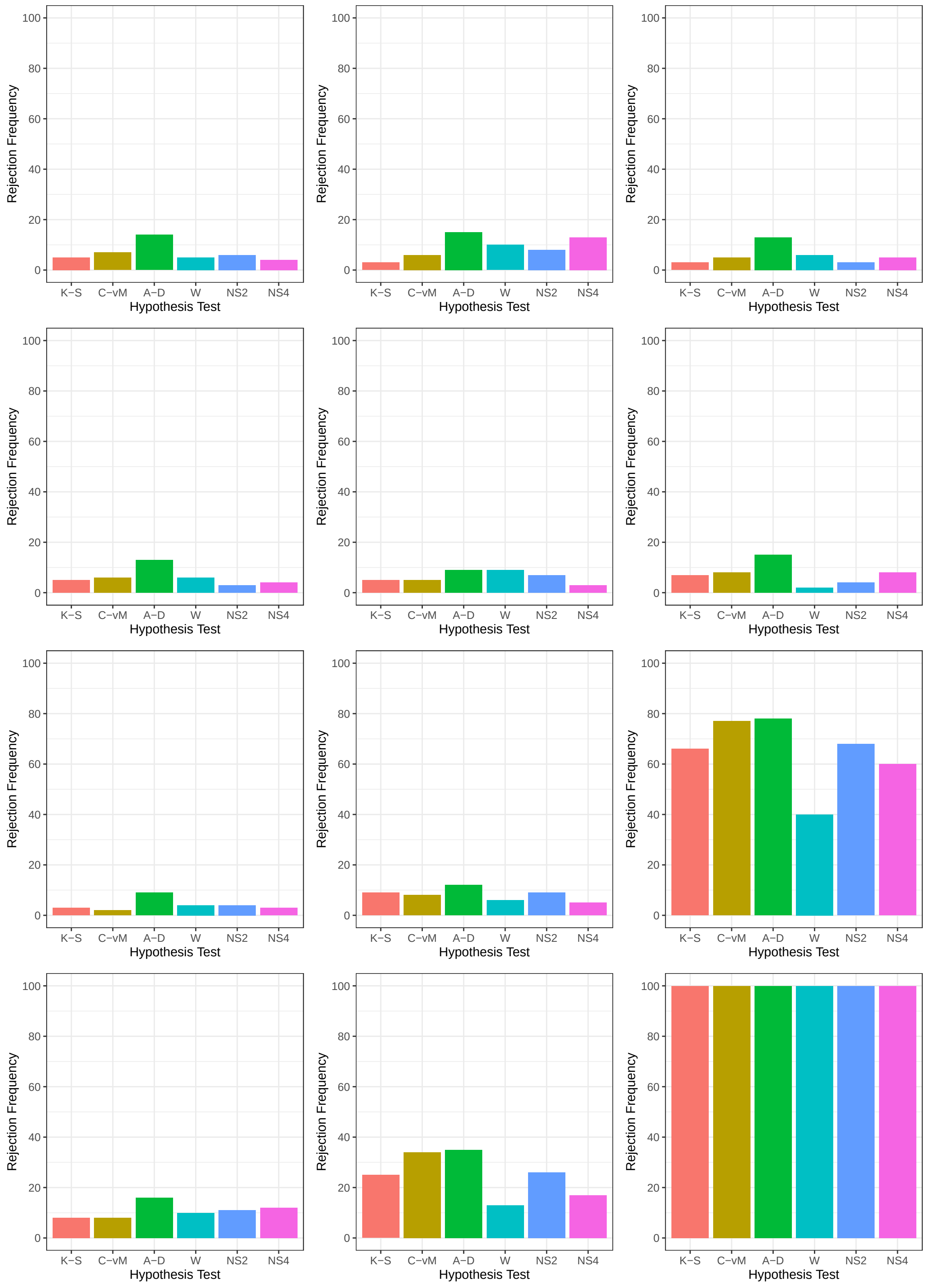

Figure 7.4.4: Number of rejections for each hypothesis test in each system (k, ρ) in the Day 2 disruption. Rows are $k = 1000, 100, 10, 2$, columns are $\rho = 0, 0.5, 0.9$.

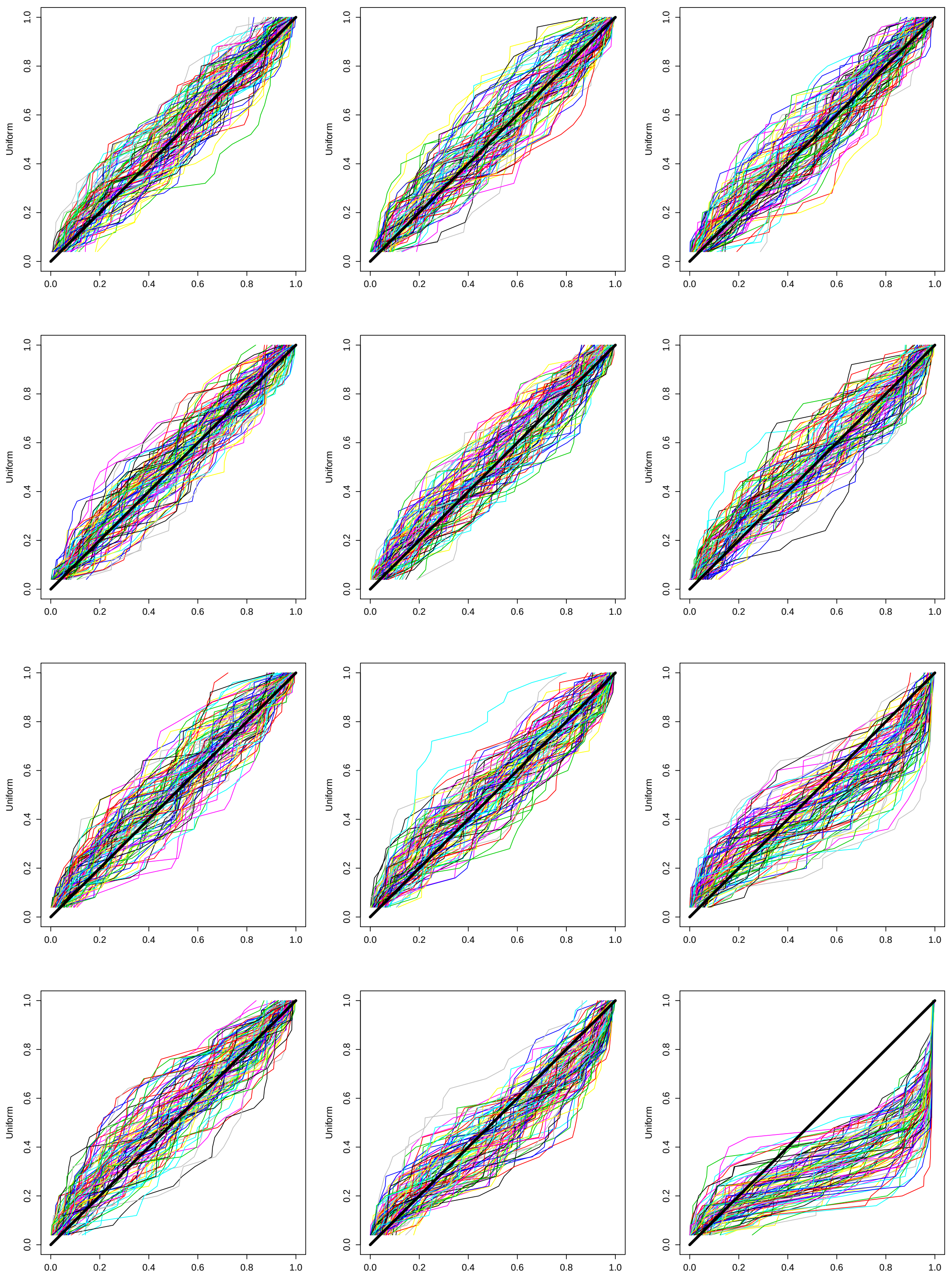

Figure 7.4.5: Day 6 P-P plots of the sets $\{U_j^{k\rho}\}_{j=1}^{J}$, for each (k, ρ). Rows are k=1000, 100, 10, 2, columns are $\rho = 0, 0.5, 0.9$.

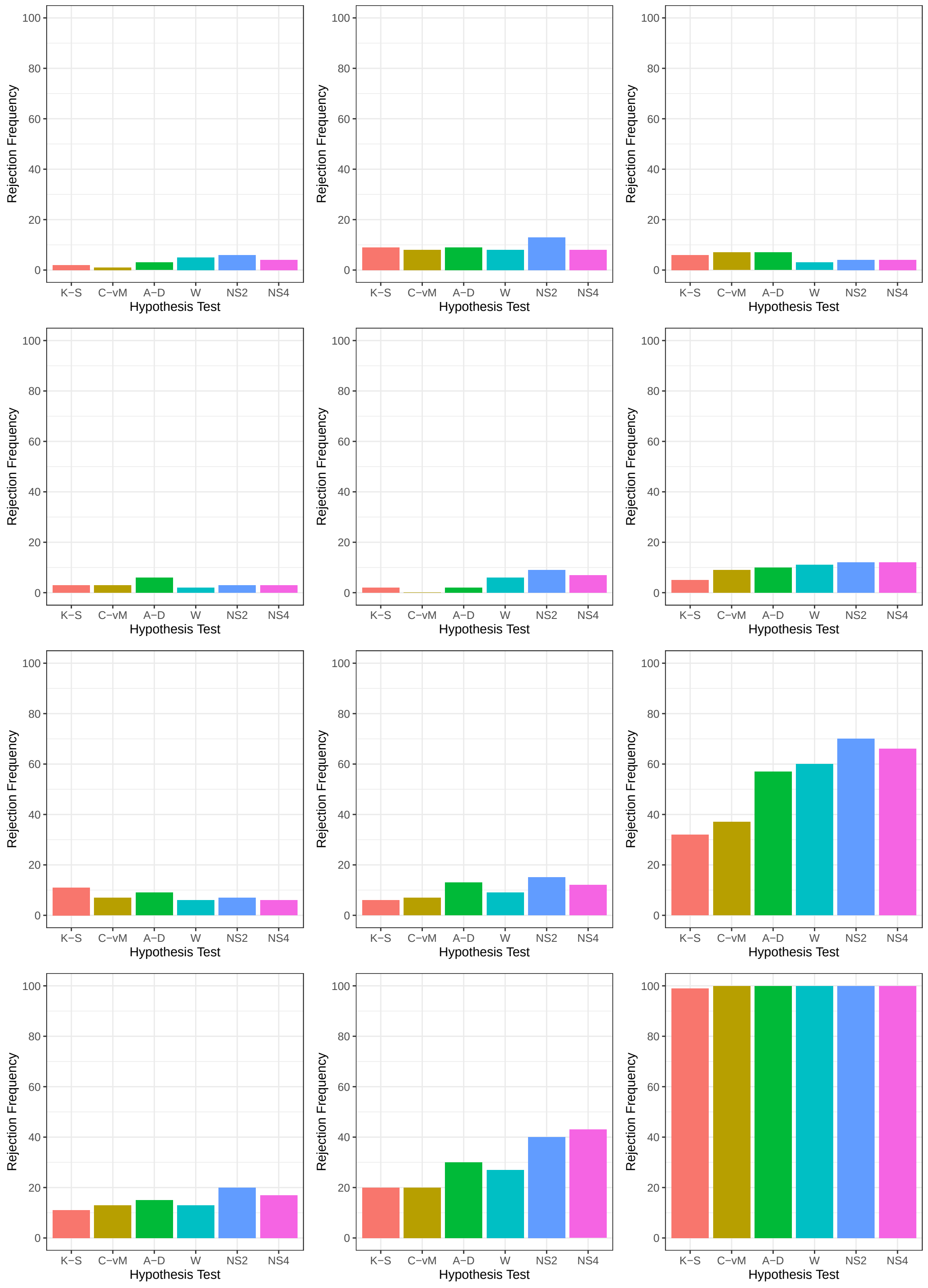

Figure 7.4.6: Number of rejections for each hypothesis test in each system (k, ρ) in the Day 6 disruption. Rows are $k = 1000, 100, 10, 2$, columns are $\rho = 0, 0.5, 0.9$.

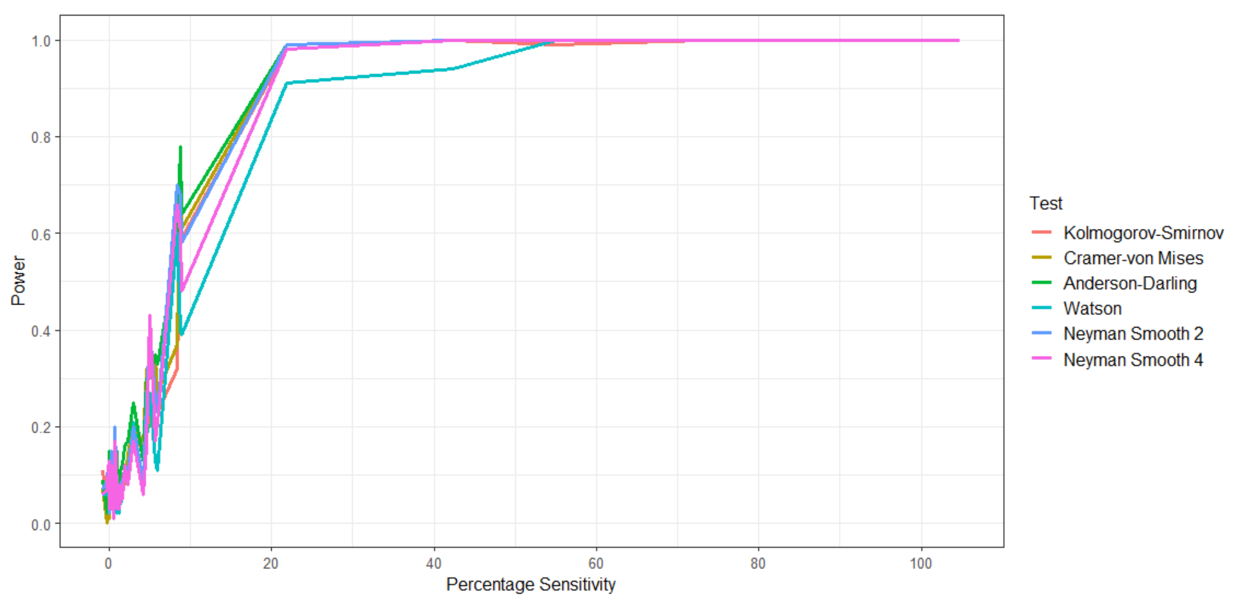

Figure 7.4.7: Relationship between sensitivity and power.

20%, the proposed transformation leads to a good power. With a sample size of 25, all the tests have a power above 0.9, and only the Watson test dips below 95%. This suggests that the proposed method would reliably detect deviations of this magnitude. The increase in power is particularly sharp in the interval from 5% to 20% sensitivity. However, if deviations of less than 5% are important to detect, these tests do not have sufficient power (highlighted particularly in the log-scale plot in Figure 7.4.8). In these cases, it seems that the Anderson-Darling test is the best performer for the goodness-of-fit statistic. Further work is required to develop the power for smaller deviations.

The second observation is counter to our initial expectations. The disruption costs do not appear to be very sensitive to the queueing behaviour. A large proportion of the observations have $\mathcal{S}_{k\rho} < 1\%$. This observation surprised us. The runway queue

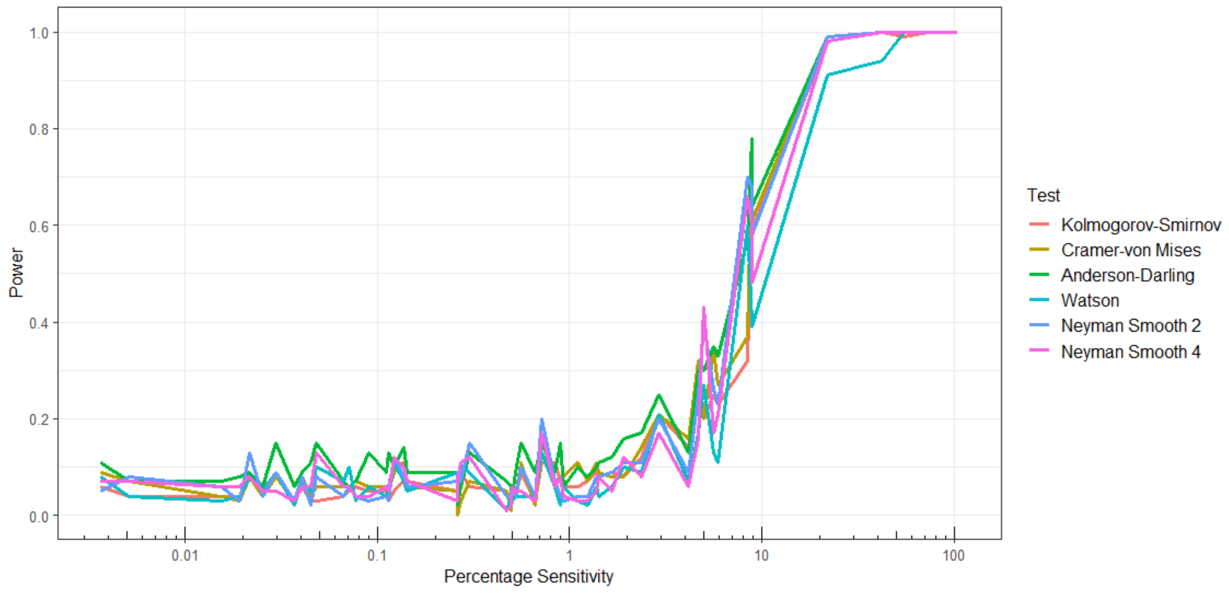

Figure 7.4.8: Relationship between absolute sensitivity and power, plotted on the log scale.

service times were chosen for the deviations in the proxy real world for two reasons. Firstly, the model described in Chapter 4 did not account for some of the queueing mechanisms of the real system; secondly there are well known results from queueing theory to guide the expected result. The introduction of greater variance increases the expected time spent in queues, causing more delay and thus increasing cost. Under high traffic levels, such as in poor weather conditions, we anticipated a more noticeable effect. The autocorrelation, intended to model aircraft sequencing, also meant that sequences of long service times would be more likely, again increasing delays.

The observed sensitivity of cost does not match our expectations. As the mean service time stays the same, the expected delay due to queueing is only dependent on the number in the queue. The effect of increased service time variance on this appears

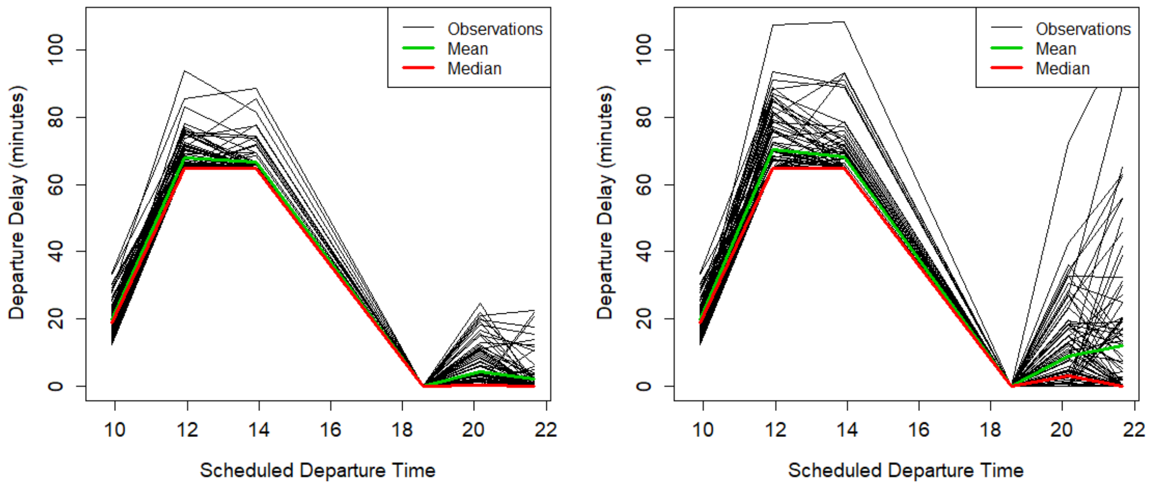

Figure 7.4.9: Trace data of the Day 2 delay of a disrupted aircraft's flights in the $(1000, 0)$ (left) and $(2, 0.9)$ (right) systems from 100 replications. Departure time against delay. Also shows the mean and median across 100 replications.

small, until the extreme case of k=2. The effect of high autocorrelation can lead to shorter queues if $Y_1 < 78$s, or increased waiting times if $Y_1 > 78$s. Simulation results in Fiems et al. (2013) show that for an auto-regressive $M/AR(1)/1$ queue, the expected number in the queue at steady-state decreased when increasing the autocorrelation.

Furthermore, the expected number initially in the queue has no dependence on k or ρ. Thus, if an aircraft arrives at a queue with a large backlog near the beginning of the simulation, its queueing delay and hence its knock on costs will not be greatly influenced by k or ρ, except in the extreme cases. This is what occurs in Day 2, and is shown for one of the disrupted aircraft under the $(1000, 0)$ and $(2, 0.9)$ systems in Figure 7.4.9. The departure delay does not include queueing for the take-off runway, therefore the effect of the first congested queue is seen in the delay of the second flight (at around 12 noon). Whilst there is a clear difference between the two systems it is not that large and the median delay does not change at all. As time progresses, the queue lengths seem to become more dependent on k and ρ, the distribution becoming more skewed (the mean delays increase much more than the median). Therefore, if aircraft join a disrupted queue later, but before recovery, the delays are much more variable, and can contribute more to the cost. This is observed in Day 6. In fact, quite a significant proportion of the additional cost in the (2,0.9) system actually comes from queues not related to the original disruption, occasionally causing large delays for 'non-disrupted aircraft' later in the day. This is seen in the later part of the day in Figure 7.4.9. After the delay has uniformly gone to 0, meaning that normal operation has been resumed, the behaviour of the next two queues change dramatically. The mean is more heavily affected by the change than the median, suggesting that it is a change in the tail behaviour of the queue length distribution rather than the lower behaviour. It is also worth pointing out that scheduled block time often contains sufficient slack to recover from minor queueing delays without additional cost.

The full results for sensitivity are shown in Table D.1 in Appendix D.

7.5 Wider Discussion

In this section, we discuss and highlight some broader issues around this topic of evaluating a simulation-based DSS. This discussion is intended to provide a basis for further investigations in this area.

7.5.1 Aims of Detection

The key questions of interest for this chapter are the performance of a simulation-based DSS and whether it is making good decisions. In the light of this, the most important deviations between the real world and the simulation are those that affect the performance measure R_j. It is these departures from a performance that we are aiming to detect, rather than more general departures from the physical system. A deviation in the modelling of a process that the difference between $F(\,\cdot\,|\Psi_j, T_j, x)$ and $G(\,\cdot\,|\hat{\Psi}_j, T_j, x)$ is not sensitive to is not important. All models make simplifications, which are desirable from many perspectives, as long as this simpler model leads to good decisions being made.

Any system used for tracking the efficacy of simulation-based DSS must account for the fact that even the optimal solution to the problem with distribution $F(\,\cdot\,|\Psi_j, T_j, x)$ could result in poor performance when implemented due to inherent system variation. A single poor performance may not be indicative of an error in the procedure. Over-reaction to this could lead to unnecessary alterations made to the system. Rather, it is systematic deviations which result in systematically poor decisions that we wish to detect. In this regard, what we are aiming for is not dissimilar from the Control Charting methodology in quality and process control, where a system is either deemed in or out of control. The out-of-control system could be identified after a series of rejected hypothesis tests based on a rolling time window.

As our primary focus is on whether or not use of the DSS is leading to good decisions, there may be some types of deviations that a user would not be concerned about. For example, consider the case where there exists a constant t such that

$$F\left(r|\Psi_j, T_j, x\right) = G(r - t|\hat{\Psi}_j, T_j, x) \qquad \forall r \in \mathbb{R}, \forall x \in \mathcal{X}_j. \tag{7.5.1}$$

Here, the simulation does deviate from the true performance, but this bias is uniform and simply shifts the distributions. In this case, the optimal solution to the simulation optimisation problem is also the optimal for the physical system. Whilst the underestimation may be of interest, it is likely to be a secondary concern if the optimal is still reached. Thus, one of the most common aims of a statistical test, that of a difference in mean, is not a primary concern. This example demonstrates a complication in the problem: we are mostly interested in detecting differences between $F\left(\,\cdot\,|\Psi_j, T_j, x\right)$ and $G(\,\cdot\,|\hat{\Psi}_j, T_j, x)$ that alter the best course of action. Part of the future work in this area could be to develop methods that are insensitive to some simple deviations, such as a shift, that leave the optimal solution unchanged.

7.5.2 Types of Deviations

There are a number of ways that a simulation model could be misspecified, leading to a deviation from the physical system. We have grouped these into four main (non-exhaustive) categories.

(i) A modeller may have a good understanding of the system logic, but the real processes are modelled by a simpler process, such as a single distribution. A distribution may differ from reality by:

- misspecification of the mean (though this is less common as the mean is often matched);

- misspecification of process variability or not accounting for all sources of variability, due to both internal and external factors;
- assuming independence when dependence plays a role in the operation.

The deviations in service times in Section 7.4 are of this type. As part of the abstraction, a simulation model could ignore factors and complexities in a process that happen to be important in a particular decision.

(ii) Another group of misspecifications could be that a physical process depends on the state of the system or varies in time whilst being modelled as a static process. This stationary assumption may impact predictions, without being important during normal operations. In our model, the availability of airport resources, such as stands, will be less predictable during a busy period for the airport, increasing the variability of turn times. This in not included in our simulation model, but may alter the optimal solution as leaving more slack at this time of day may be advantageous.

(iii) A more extreme version of the two previous groups is that a process is not modelled at all. It is common for some real-world processes to be ignored when building a model, especially when they do not appear to influence the output of the simulation. However they may be key in some problems, particularly under irregular operations.

(iv) Our final class of deviations is that the full system state is not described in the simulation. A simulation may fail to capture the entirety of the initial state of the system. This is not to say that the information would be incorrect, but incomplete. There are circumstances where this could impact the prediction. An example from our simulation could be that Engine Health Management data are not used directly, which would impact on the time-to-failure models.

7.5.3 Considerations for the Transformation and Test

Due to the heterogeneity of problems faced, particularly that $F(\,\cdot\,|\Psi_j, T_j, x_j)$ is almost certainly not the same as $F(\,\cdot\,|\Psi_i, T_i, x_i)$, it is likely that some transformation will be necessary for any hypothesis testing. We have discussed the use of the simulation ECDF to make this transformation. If the simulation cannot be run many times or the transformation deviates significantly from uniformity, a smoothed version of the ECDF may help to improve the estimation of the true distribution. It would be useful to quantify this deviation and to consider if the hypothesis test should be changed to account for this approximation. An alternative could be to use a parametric distribution, fitted to the observations from the simulation. This would enable more powerful tests, specific for that distribution type, but at the price of increased uncertainty due to the necessary choice of distribution. This may be applicable in some cases more than others, dependent on the knowledge of the system.

The performance measure of interest plays a large role in the efficacy of the detection. Whilst a particular quantity may be of central importance, and so used as the objective for an optimisation procedure, it may not always be easily or instantly observable in all contexts. There may be alternative performance measures, which are of secondary concern but are better understood and more easily observed that could provide a better insight into how well the simulation model is doing. The more that is understood about R_j, the more detailed a detection procedure could be. For example, throughout this thesis the key performance measure used for the simulation optimisation procedure has been the cost due to disruption and delays. This includes aspects like passenger compensation, which may not actually be observed for a week or so after the events being simulated. The total delay, on the other hand, is related to cost and very easily observed during T_j, and so could be tested very soon after t_{j2} rather than waiting.

The transformation and hypothesis tests in Section 7.3 assume some level of independence between intervals. The short-haul setting involves a natural time horizon for the simulation, i.e., working days are well defined and, on the whole, operate independently of each other. Many delays are absorbed over night, meaning that major issues of a disruption rarely give rise to further problems the next day. Therefore, the cost and delay observations from the real world on different days are largely independent, and so transforming the observations through the simulation ECDF leads to a set of independent random variables. This is not to say that the intervals themselves must be non-overlapping or that they cannot impact each other. The key is that we have independent outcomes given the past decisions. We believe that the minimum required is that the intervals are independent given the state of the system at the time the simulation-based DSS is triggered. However, this must apply to both the system and the simulation. If the state of the simulation system, $\hat{\Psi}$, did not capture all the required information from the system's history, then the independence may not be guaranteed. In this case, a time-series treatment may be a useful approach to help model the dependence. This is an area for further work.

Our approach is concerned with the performance measures over the entire period T_j. This has been the standard approach to validating simulation models for a long time. An alternative perspective that has gained some traction recently would be to consider simulation analytics (see, for example, Nelson (2016)). This would allow the processes themselves to be compared, rather than a total or time averaged performance measure. If one could use areas of time-series or functional statistics, there is the potential to improve the detection of deviations as well as the identification of the causes of deviation.

7.6 Conclusions

This chapter has highlighted and discussed the issue of the evaluation of simulation-based Decision Support Systems. It has been demonstrated that the tracking of the efficacy of such a system is not as straightforward as measuring the outcome, as natural variability within a system may lead to an optimal solution giving a poor result. To our knowledge, this area has not been previously studied and yet is an important issue when using simulation for operational decisions.

This chapter presents a method based on the Probability Integral Transformation to detect discrepancies between the predicted and observed performance. It has been demonstrated within the airline disruption context by comparing a simulation with another simulation acting as a proxy real world. Example results based on 25 disruptions show that the transformation-based method is able to detect deviations of 20%

or more fairly consistently, but there is quite a dramatic fall off in power as deviations fall below 10%.

A simple extension of this work would be to investigate the benefits of increasing the number of decisions used in the test. As this increases, we would expect power to improve. However, in the context of airline disruption, 25 disruptions could represent over a month of use, by which time a user may hope to be evaluating a DSS. Hence further work could also be to investigate the method under a sequence of disruptions, rather than a single disruption, mimicking the actual use, as well as experimenting with the transformation to increase the power of the hypothesis tests.

Section 7.5 provided a wider discussion of issues related to the evaluation of the simulation-based DSS, with potential literatures suggested to help inform the investigations. There is clearly an interesting range of issues worthy of further research in the search for an objective method for evaluating a simulation-based DSS.

Chapter 8

Conclusions and Further Work

This thesis has presented research into how one could use simulation within the airline industry to manage disruption to schedules, specifically looking at the Aircraft Recovery Problem (ARP). The proposed methodology aims to find and evaluate potential rescheduling options, either allocating an aircraft and a planned delay or a cancellation to each flight, whilst accounting for the inherent uncertainty in the environment. To the best of our knowledge, this is the first investigation of a high-fidelity discrete event simulation model within the search process for the ARP.

In addition to the development of this methodology, contributions have been made in (i) extending the STRONG algorithm to handle more general problems with bound constraints and (ii) addressing the question of how to evaluate a simulation-based decision support system in an environment with natural variability.

This chapter summarises the research contributions of this thesis in Section 8.1. Identified areas for further work and improvement are given in Section 8.2. Final comments on the work are offered in Section 8.3.

8.1 Contributions

Solving the Aircraft Recovery Problem

The primary contribution of this thesis is the development of a multi-fidelity modelling approach designed to find a set of good solutions to the Aircraft Recovery Problem. This algorithm was developed in Chapters 3 and 4. The low-fidelity model takes the form of an Integer Program (IP) which is solved in a multi-objective manner using the ϵ-constraint method. The use of the IP means that the combinatorial aspects of the ARP do not have to be performed within the simulation optimisation, a problem that many current methods struggle to handle. The IP is able to quickly find good aircraft allocations that display different optimal balances between the cost, delay and schedule alteration objectives. The result is a set of promising solutions that can then be evaluated within a high-fidelity simulation model.

The IP model developed in Chapter 3 is mainly based on established methods. The primary novelty is its interaction with the simulation model. As the error of a low-fidelity model may not be uniform across the solution space, the simulation optimisation performs an iterative local search to improve the cost objective of each of the promising solutions. The optimisation takes place over the planned delays, treating them as continuous variables, as described in Chapter 4. To perform this optimisation, an extended version of STRONG was developed to handle the bound constraints on delays. This was necessary as the optimal solution may lie on the boundary of the feasible region.

The overall method enables the selective use of simulation in a combinatorially constrained problem. Furthermore, it is used directly within the search procedure.

An empirical study of the performance of the method based on a set of three example problems with different practical challenges is presented in Chapter 5. As the true optimal solution and optimal value to the simulation optimisation problem is unknown, we evaluated the cost performance of each solution produced by the algorithm using a set of 1000 replications of the simulation. These results demonstrated that the multi-fidelity modelling approach finds good solutions to the problem, and that using the simulation directly in the search typically improves upon the IP-based solution. The final solution varies with the starting seed of the optimisation; it is not guaranteed to improve the solution (see Section 5.6), though no examples of decreasing quality were found. As the ARP is a one-time-only decision, it is important to consider how the solution performs in a variety of possible realisations of the future. To this end, we also reported the histograms of the 0.95 quantile of cost, which sometimes increased compared to the IP-based solution. This should be considered when choosing a solution to implement.

Extending STRONG

The empirical results in Chapter 5 rely on a particular simulation model (and a slight variant in Section 5.6). If an airline wanted to include additional problem aspects in the model the simulation model would change. To show that the proposed method of local improvement could apply beyond the particular model used here, a more theoretical perspective is given in Chapter 6. Here, the extensions to STRONG are described in more detail. One extension implements ideas from constrained trust-region optimisation to generalise the optimality measure and the method for proposing a new solution. The other draws upon ideas from experimental design, using a coordinate-exchange algorithm to build designs for meta-model estimation when the trust region is not entirely feasible. These extensions allow the STRONG framework to handle a general continuous simulation optimisation problem with bound constraints, and are a contribution to the wider simulation optimisation literature. The adaptations were developed to support the convergence of the algorithm, intuiting that convergence theory from constrained trust-region optimisation would have relevance in this context. Whilst a full proof of asymptotic behaviour is not obtained, a pathway for the proof is developed, highlighting where further work is required.

Evaluating a Simulation-based Decision Support System

If implemented, the proposed method would be used repeatedly for operational decisions regarding the rescheduling of flights and aircraft on a regular basis. The final contribution of this thesis highlights the difficulty of evaluating the decisions made by such a simulation-based DSS and proposes some solutions. Chapter 7 discusses this issue and argues that the problem is difficult due to natural variability of the system, meaning that even an optimal solution can perform badly, giving the impression that the DSS has done a bad job. There are two parts to a DSS in this context: the simulation used to predict the performance of a decision and an optimisation methodology, which includes repeated usage of the simulation. Both of these parts contribute to the quality of the solutions found. The evaluation of the optimisation is a very difficult problem due to its interplay with the simulation, and is beyond the scope of this work. However, in Chapter 7 we propose a method to evaluate the simulation model, attempting to detect the presence of systematic inaccuracies. The natural variability and uniqueness of conditions faced means that statistical detection of the deviations is difficult. This thesis proposes a simple transformation using the simulation ECDF combined with a goodness-of-fit hypothesis test to detect issues. This method was evaluated using the proposed ARP methodology and a set of proxy real worlds created by changing the runway service times in the simulation. The results suggest that, with a sample size of 25 decisions, the test can consistently detect differences between the simulation and proxy real world of more that 20%. The results also show that the power of the test is poor for deviations of less than 10%.

8.2 Further Work

Throughout this thesis, areas of further work are identified, both in terms of methods already existing in the literature but not implemented here and in developments that require further research. This section gives an overview of some of these.

There are aspects of airline disruption that have not been accounted for in this research. Whilst we believe many of the key properties are included within the simulation, there are others that could enhance the utility of the method. For example, one could include passengers within the simulation and IP to explicitly model the costs of missed connections, or expand the IP to include different forms of aircraft.

Integer Programming Model Development

The IP model described in this work may not be the best model for solving the ARP. There are a number of improvements that could be made. These include problem reduction heuristics, such as that proposed by Rosenberger et al. (2003), and more problem-specific solution methodologies. The results in Chapter 5 indicate that, whilst the first solution minimises cost, the pure ϵ-constraint algorithm takes several iterations to reduce the aircraft exchanges to the Pareto optimal. Initial experiments with an alternative hybrid multi-objective approach reported in Appendix B appear to help this issue.

Reducing the Computational Burden of the Simulation Optimisation

Before implementing the simulation optimisation, it is advised that a parameter tuning experiment is undertaken, particularly for those parameters that control the trust-region radius and the threshold for switching to the quadratic model. The algorithm can be parallelised fairly easily to reduce the computational burden of the method, particularly during the experimental designs where each design point could be given to a separate processor.

At each Stage I iteration, the proposed solution from the linear model is very sensitive to the sign of the gradient estimator components, regardless of magnitude (due to the projected gradient step). One way to improve the efficiency of the algorithm could be to set all components that are not significantly non-zero to 0. Rather than moving to a corner of the hyper-box trust region, the proposal could then be at the centre of a face. This would prevent large steps in a direction with a very small gradient.

The results in Chapter 5 seem to indicate that the simulation optimisation approach tends to run out of budget rather than converge to an optimal. As the convergence criterion, $\pi(\mathbf{d}) = 0$, was not obtained, the key aim could be revised to achieve local improvement within the fixed budget. On reflection, this could support the use of an inexpensive gradient estimator, such as those used in Stochastic Approximation methods, rather than the expensive experimental designs. This could be achieved by looking at the simulation sample paths more closely.

Theoretical Properties of the Simulation Optimisation

There is also further work regarding the simulation optimisation as a general algorithm, such as an investigation of its performance on known analytical functions and research into filling the gaps in the proof of convergence presented in Chapter 6. The full proof relies on the trust-region radius either being bounded away from 0 or tending to zero in such a way that the sum over the iterations is infinite. One other gap remains, that of proving that the errors from a general OLS gradient estimator only exceed the trust-region size finitely many times. If these conjectures can be proven, the path towards asymptotic convergence to a first-order optimal point will be complete.

Dynamic Application

Throughout this thesis, each disruption has been treated as a static problem, that is, there is only one decision point during the disruption. In reality, disruptions evolve with time, and once some uncertainties become clear, the optimal decision may change. This motivates two possible research directions for improvement. The first is to use the power of symbiotic simulation to track the evolution, and periodically re-evaluate what the best decision is, possibly extending the methods proposed in this thesis. An extension of this is preventative work, where the simulation is operating continuously, using weather and aircraft health data to forecast disruption and suggest schedule rearrangements to avoid the disruption before it happens. This style of symbiotic simulation was suggested by Aydt et al. (2008b).

The second possibility is to account for some form of recourse action that could be taken later on in the day and only committing to immediate decisions. Recourse action could be retiming of flights when some uncertain quantities became known. To include this within the decision making structure may require changes to the optimisation methodology considered within this thesis.

Both of these directions are open research topics that could enhance the use of simulation within disruption management.

Simulation-based Decision Support Systems

The method discussed in Chapter 7 is an early approach to detecting differences between the simulation and the real world. Further work is required to test the method on the more realistic scenario of a series of different disruptions to see if this changes the performance of the test. To understand the properties of this method more fully, further experiments should be run, considering the effect of sample size and variations on the transformation (such as using a smoothed ECDF). Ideally this should be across a variety of simulation models and applications.

Chapter 7 identified a number of wider issues that are not accounted for in this research. These include considering how dependence between decisions can be modelled, how one could detect differences that alter the optimal solution to the problem and how to find the problem in the simulation given that a deviation has been detected. Further research into these areas could help to develop a detection method to evaluate simulation-based DSS in practical problems, and detect differences that really affect the decisions made.

8.3 Final Comment

This thesis has investigated the use of simulation to aid airlines to manage disruption to their schedule by finding and evaluating potential rescheduling options, particularly focussed on the Aircraft Recovery Problem. To do so, a multi-fidelity methodology combining deterministic optimisation and simulation optimisation has been proposed and demonstrated, with promising computational results. We have identified possible directions for further work to develop the methodology, both theoretically and to reduce the computation time.

In addition, the issue of how to evaluate this tool once implemented has also been discussed and a method for detecting poor simulation predictions has been suggested. The experiments imply the approach has potential, being able to detect substantial changes with moderate sample sizes.

Appendix A

Simulation Model

This appendix gives more details of the simulation model, and considers the extended version used in Chapter 7. The simulation is built in AnyLogic 8.2.3 (The AnyLogic Company, 2017), a Java based simulation software. Throughout this appendix, 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.

A.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 A.1.1. 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

Table A.1.1: 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

the initial activity times to come from a conditional distribution. To increase the correlation when using CRN, each aircraft has its own random number generator. 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 A.1.1 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 A.1. The right-hand side of Equation (A.1.1) is the Weibull(α,β) CDF evaluated at ‘flightTime’. The right-hand side of Equation (A.1.2) is the inverse CDF of the same distribution. The proof that this creates the appropriate conditional distribution is straightforward, but not included here.

Algorithm A.1 Generating the failure time of an aircraft

```
1: Calculate the probability of failing before ‘flightTime’:
       $$u_0 = 1 - \exp\left\{-\left(\frac{\text{flightTime}}{\beta}\right)^{\alpha}\right\} \qquad (A.1.1)$$
2: Sample a uniform random number: $U \sim U(u_0, 1)$
3: Calculate failure time using the Inverse Probability Integral Transformation:
       $$\text{failure} = \beta(-\log(1-U))^{1/\alpha} \qquad (A.1.2)$$
4: **return** failure
```

A.2 Main

In the simulation model, the aircraft move in two environments. One is the Airport, which is discussed in Section A.3. 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 A.2.1. 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.

Table A.2.1: 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
busyPeriods	double[4]	Start and end of the peak periods
autoCorr	double	Autocorrelation of runway service times
phase	integer	Phase number of runway service distribution
erlangMean	double	Erlang mean of runway service distribution
closedAirports	ArrayList	List of currently closed airports
cost	double	Output variable used to accumulate cost

A.2.1 Flights

Once the aircraft has been served by the airport take-off runway, the aircraft leaves the airport class (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 (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

$$\text{duration} = \alpha \left(\frac{U}{1-U} \right)^{1/\beta} - \text{SpacingDep}(t) - \text{SpacingArr}(t), \tag{A.2.1}$$

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 A.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.

A.2.2 Runways

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, the minimum spacing is set to 1 minute between 6am and 11pm, and 3 minutes overnight.

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. This assumption produces unreasonably high congestion at peak hours at larger airports, namely AMS, CDG and MAD (each of which have at least four runways). Thus, for peak periods of the day, these airports use two runways for each queue. These periods are defined by the 'busyPeriods' parameter: the first two elements define the morning interval; the second two define the evening interval. The 'peakTimeStart' event increases the capacity of each runway resource to 2 (see Section A.3) and schedules both the end of the peak period, the 'peakTimeEnd' event, and the subsequent 'peakTimeStart' event. The 'peakTimeEnd' event decreases each resource capacity back to 1. This only applies to the largest airports.

A.2.3 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.

A.2.4 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.

A.2.5 Initialisation

The Main environment controls the initialisation of the simulation, including the fleet and airport populations which are described in the relevant sections. The ‘erlangMean’ parameter is changed depending on the ‘phase’ parameter using a look-up table. This ensures that the marginal mean of the runway service time distribution is 78 seconds.

The start time is used to calculate whether the large airports are in a busy period or not. If it is during a busy period, a ‘peakTimeStart’ event is scheduled to happen immediately. Otherwise, the next peak period is identified and a ‘peakTimeStart’ event is scheduled for that time.

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.

A.3 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 A.3.1 (the longitude and latitude coordinates are also supplied for visualisation purposes but play no role in the calculations). In principle, they could vary across airports and be read in from an input file. In this work, 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 'erlangMean' and the ArrayLists.

A.3.1 Runway Resources

The Airport environment has two resource entities: 'landingRunway' and 'takeoffRunway'. These resources serve the landing and take-off queues, respectively. The default is that each will have a capacity of 1. For the three large airports in the database, AMS, CDG and MAD, the capacity is controlled by the peak time events in the Main environment (see Section A.2.2). Under poor weather conditions, the capacity can be set to 0, closing the airport (see Section A.3.5).

Table A.3.1: 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
autoCorr	double	Autocorrelation of runway service times
phase	integer	Phase number of runway service distribution
erlangMean	double	Erlang mean of runway service distribution
artaAirspace	ARTA1Erlang	Generator of service times for landing runway
artaRunway	ARTA1Erlang	Generator of service times for take-off runway
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

A.3.2 Path through Airport for our Aircraft

Figure A.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.

A.3.3 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

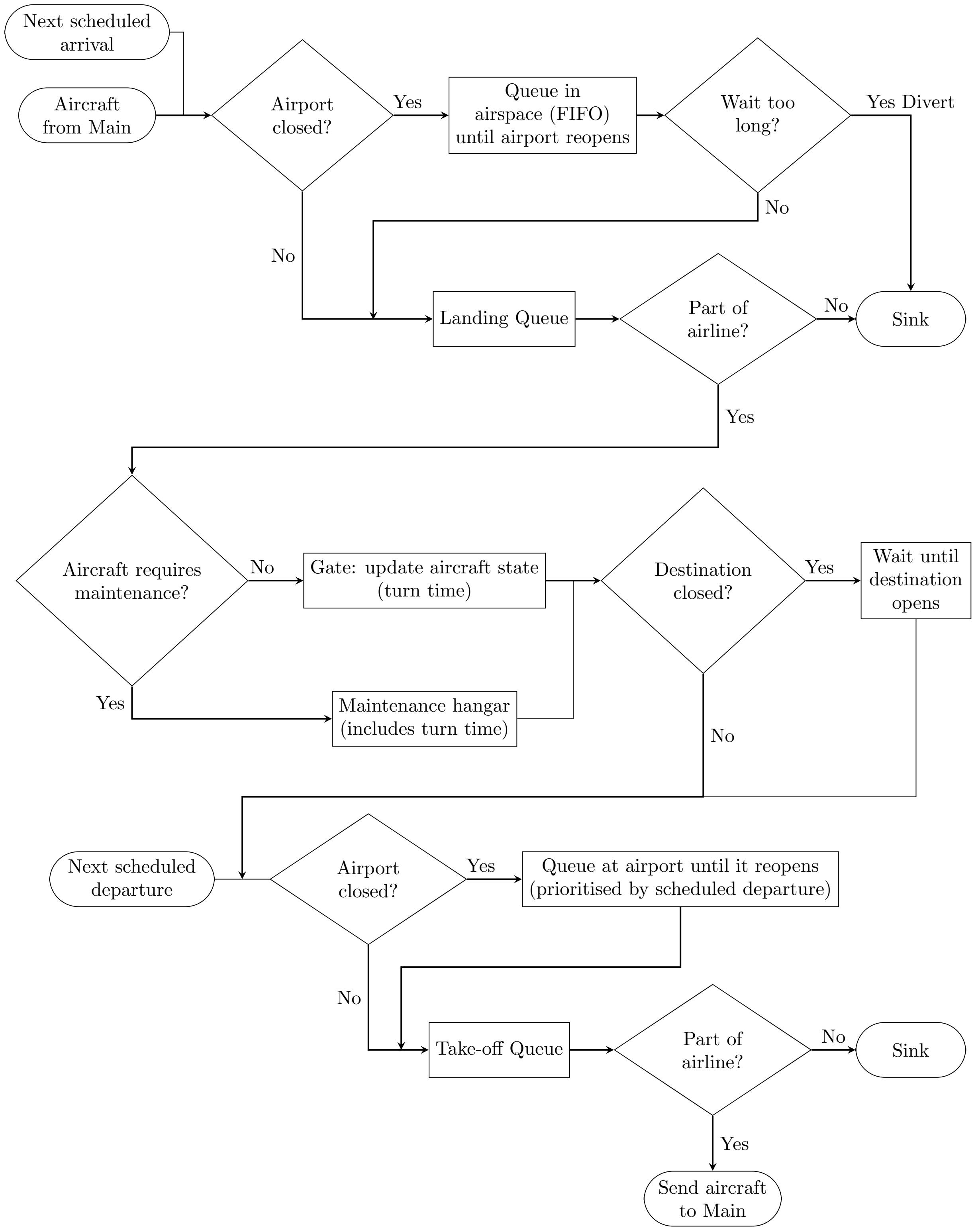

Figure A.1: Flow diagram of the path taken by an Aircraft through the Airport environment in the simulation.

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 parameters are modelled individually for each airport. 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.

A.3.4 ARTA Service Times

The service rates follow a first-order ARTA process (Cario and Nelson, 1996). This transforms a first-order auto-regressive, AR(1), process using the Probability Integral Transform and its inverse. The base autocorrelation, r, of the AR(1) process to match the desired ARTA(1) autocorrelation must be calculated. This was done offline, and a look-up table is used in the simulation to find the appropriate base autocorrelation (based on the 'autoCorr' parameter).

Algorithm A.2 Generating the Runway Service Times

```
1: Calculate the time-dependent minimum spacing, $s(t)$, from the appropriate step-function (SpacingDep or SpacingArr)
2: if ‘autoCorr’=0 then
3:     Generate $Y_i$, sampling from an Erlang distribution, with ‘phase’ phases and scale ‘erlangMean’/‘phase’
4: else
5:     Sample from Normal noise: $\varepsilon \leftarrow N(0, 1 - r^2)$
6:     Generate next AR(1) sample: $X_i \leftarrow rX_{i-1} + \varepsilon$
7:     Store $X_i$ as ‘currentNormal’ variable for the ARTA1Erlang object
8:     Transform to the service time distribution $Y_i \leftarrow F_Y^{-1}(\Phi(X_i))$
9: end if
10: Calculate spacing: $Q_i \leftarrow \max\{Y_i, s(t)\}$+‘currentWeather’
11: return $Q_i$
```

The process for generating the runway service times is given in Algorithm A.2. In step 8, F_Y is the CDF of the marginal service time distribution:

$$F_Y(y) = F_Z(y)\mathbb{I}\left(y \geq \min_t\{s(t)\}\right) \tag{A.3.1}$$

where F_Z is the CDF of the Erlang distribution and $\mathbb{I}(\cdot)$ is the indicator function. The modelling assumptions are described in Section 7.4.1. If the autocorrelation is 0, the ARTA process is not used, and Y_i is simply sampled from the marginal Erlang distribution.

At the beginning of the service period of the take-off queue, the aircraft’s ‘flight-

Start' 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.

A.3.5 Weather Change

The weather conditions are represented by the airport variable 'currentWeather', which gives the additional separation time required between aircraft using the same runway (0 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 A.3.

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 Algorithm A.2), the 'currentWeather' variable also indicates whether the airport is open or closed. For the purposes of this research, we assume that if 'currentWeather' exceeds 4, an airport closes. Much of Algorithm A.3 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

Algorithm A.3 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 $\geq 4$ & 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 $< 4$ & 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 forecastTime$[i]$
```

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 A.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.

A.3.6 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 A.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 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.

Appendix B

Hybrid Multi-objective Optimisation

This appendix gives the results of using a hybrid multi-objective solution approach to the IP problem in Chapter 3. A hybrid approach, such as that described in Section 4.2.2 of Ansari et al. (2018), finds weakly efficient solutions by solving the problem with a weighted-sum of all the objective functions as the problem objective, subject to the objective values belonging to a particular interval. Other than the time to solve each IP, the main issue observed in Chapter 5 with the solution method was that the algorithm found cost optimal solutions that were dominated by other solutions with fewer aircraft exchanges. Thus, the only alteration made is that the objective function used when solving the IP is a weighted sum of the cost and exchanges:

$$\min \qquad \left(\sum_{a \in A} \sum_{f_\delta \in L} c^{f_\delta} x_a^{f_\delta} + \sum_{f \in F} C^f y^f \right) + \varepsilon \sum_{a \in A} \sum_{f_\delta \in L} (1 - o_a^f) x_a^{f_\delta}. \qquad \text{(B.1)}$$

If ε is chosen small enough, the discretised nature of the objective function means that we can still produce a cost optimal solution whilst having a small penalty for unnecessary exchanges. In the early results, we chose $\varepsilon = 0.00001$, which is smaller than any unit of cost in the problem. As costs and delays are linked in the objective function, it is not necessary to include this in objective (B.1).

Tables B.1 and B.2 show the results of applying this approach to Problems 2 and 3 in Chapter 5. The settings are identical to those used in Section 5.4.2 and 5.4.3. The asterisks indicate that more than one combination of limits were searched before finding this feasible solution. The time reported includes the time to search through these regions as well. The results suggest that the Pareto optimal set is reached much more quickly than using the original method, and so the time budget is better spent exploring the Pareto frontier between the objectives. Many of the solutions are the same as those originally collected, suggesting that the choice of penalty is sufficiently large to encourage the reduction without inhibiting the cost optimality of the solutions. For the case in Table B.2, much of the time was spent searching infeasible regions in the solution space.

Table B.1: Solutions from the IP for Problem 2 using hybrid multi-objective approach.

Solution	Cost (€1000)	Tail Number Exchanges	Delay (minutes)	Cancellations	Solution Time (seconds)
1	6.00	19	120	0	23.43
2	6.75	17	135	0	27.49
3	9.75	14	195	0	41.91
4	11.06	13	195	0	35.51
5	12.75	12	255	0	47.97
6	14.06	11	255	0	48.77
7	17.75	10	330	0	47.46
8	19.06	9	330	0	48.62
9	18.31	10	315	0	102.29*
10	18.31	9	315	0	49.59
11	13.31	12	240	0	258.44*
12	13.31	11	240	0	52.83

Table B.2: Solutions from the IP for Problem 3 using hybrid multi-objective approach.

Solution	Cost (€1000)	Tail Number Exchanges	Delay (minutes)	Cancellations	Solution Time (seconds)
1	27.00	17	540	0	75.13
2	29.00	13	580	0	44.12
3	31.00	9	620	0	48.39
4	33.00	8	660	0	68.71
5	35.00	4	700	0	117.33
6	41.00	3	820	0	40.69
7	47.00	0	940	0	102.96

Appendix C

Trust-Region Theorems

This section will discuss a few additional aspects of the trust-region algorithms, including the properties of the feasible region $\mathcal{D}$ and some theorems regarding trust-region optimisation for constrained problems.

C.1 Assumptions on the Feasible Region

In Theorem 6.2.2, it is assumed that a first-order *constraint qualification* applies at the optimal solution $\mathbf{d}^*$. These conditions essentially state that a linear approximation to $\mathcal{D}$ at $\mathbf{d}^*$ captures the essential geometry of $\mathcal{D}$ in the neibourhood of $\mathbf{d}^*$. This is discussed more fully in Chapter 12 of Nocedal and Wright (2006). We make this more precise below. This involves defining the Tangent cone and the set of linearised feasible directions.

Firstly, let us assume that we can characterise $\mathcal{D}$ by a set of constraints:

$$\mathcal{D} = \{\mathbf{d} \in \mathbb{R}^n : c_i(\mathbf{d}) = 0 \ \forall i \in \mathcal{E},\ c_i(\mathbf{d}) \geq 0 \ \forall i \in \mathcal{I}\}$$

where each $c_i : \mathbb{R}^n \to \mathbb{R}$. Let $\mathcal{A}(\mathbf{d})$ be the set of active constraints at $\mathbf{d}$, that is:

$$\mathcal{A}(\mathbf{d}) = \mathcal{E} \cup \{i \in \mathcal{I} : c_i(\mathbf{d}) = 0\}.$$

Definition C.1.1. *Let $\mathcal{D}$ be a closed, convex subset of $\mathbb{R}^n$. The tangent cone of $\mathcal{D}$ at $\mathbf{d} \in \mathcal{D}$ is the closed set*

$$\mathcal{T}_{\mathcal{D}}(\mathbf{d}) := \overline{\{\theta(\mathbf{y} - \mathbf{d}) \ : \ \theta \geq 0, \mathbf{y} \in \mathcal{D}\}}.$$

The set of linearised feasible directions of $\mathcal{D}$ at $\mathbf{d}$ is

$$\mathcal{F}_{\mathcal{D}}(\mathbf{d}) := \left\{\mathbf{y} \ : \ \mathbf{y}^T \nabla c_i(\mathbf{d}) = 0 \ \forall i \in \mathcal{E}, \ \mathbf{y}^T \nabla c_i(\mathbf{d}) \geq 0 \ \forall i \in \mathcal{A}(\mathbf{d}) \cap \mathcal{I}\right\}$$

Both $\mathcal{T}_{\mathcal{D}}(\mathbf{d})$ and $\mathcal{F}_{\mathcal{D}}(\mathbf{d})$ describe a linear approximation to small movements about $\mathbf{d}$ that remain feasible. They can be very different sets, see Nocedal and Wright (2006) Chapter 12 pages 319-320 for an example, though these are in quite complex problems. In such cases the characterisation of optimality is more difficult. First-order constraint qualifications ensure that $\mathcal{T}_{\mathcal{D}}(\mathbf{d})$ and $\mathcal{F}_{\mathcal{D}}(\mathbf{d})$ are sufficiently similar, or even equal.

There are several examples of first-order constraint qualifications, listed below:

Definition C.1.2. *Given a point $\mathbf{d} \in \mathcal{D}$ and its active set $\mathcal{A}(\mathbf{d})$, the Linear Independence Constraint Qualification holds if $\{\nabla c_i(\mathbf{d}) : i \in \mathcal{A}(\mathbf{d})\}$ is linearly independent.*

Definition C.1.3. *The Mangasarian-Fromovitz Constraint Qualification holds if there exists a vector $\mathbf{y} \in \mathbb{R}^n$ such that:*

$$\nabla c_i(\mathbf{d})^T \mathbf{y} > 0 \qquad \forall i \in \mathcal{A}(\mathbf{d}^*) \cap \mathcal{I},$$

$$\nabla c_i(\mathbf{d})^T \mathbf{y} = 0 \qquad \forall i \in \mathcal{E},$$

and $\{\nabla c_i(\mathbf{d}) : i \in \mathcal{E}\}$ is linearly independent.

Lemma 12.7 from Nocedal and Wright (2006) gives the appropriate condition for the bound constraints case.

Lemma 12.7: Suppose that for some $\mathbf{d}^* \in \mathcal{D}$ all active constraints c_i, $i \in \mathcal{A}(\mathbf{d}^*)$, are linear functions. Then $\mathcal{T}_{\mathcal{D}}(\mathbf{d}^*) = \mathcal{F}_{\mathcal{D}}(\mathbf{d}^*)$.

Note that Theorem 6.2.2 only requires this to be true around an optimal solution; $\mathcal{D}$ does not need to be so well behaved everywhere. Furthermore, these are often sufficient, rather than necessary conditions.

C.2 Theorems on Constrained Trust-Region Optimisation

This section quotes some theorems from Conn et al. (2000). Some of the assumptions are left out as they implicitly hold for our choice of feasible region, model or norm etc.

Theorem 12.1.2: Suppose that a first order constraint qualification holds. Then a point $\mathbf{d}^*$ is first-order critical for the problem $\min\{g(\mathbf{d}) : \mathbf{d} \in \mathcal{D}\}$ if and only if

$$p(\tau, \mathbf{d}^*) = \mathbf{d}^* \qquad \forall \tau \geq 0.$$

Theorem 12.1.3: Suppose that AF.1 holds and that $\mathbf{d} \in \mathcal{D}$. Then the function

$$\phi(\tau) = ||p(\tau, \mathbf{d}) - \mathbf{d}||_2$$

is non-decreasing for $\tau \geq 0$. Furthermore

$$\lim_{\tau \to \infty} \phi(\tau) < \infty \qquad \Rightarrow \qquad \lim_{\tau \to \infty} ||\mathcal{P}_{\mathcal{T}(p(\tau,\mathbf{d}))}[-\nabla g(\mathbf{d})]||_2 = 0,$$

where $\mathcal{T}(\mathbf{d})$ is the tangent cone at $\mathbf{d}$ with respect to $\mathcal{D}$.

Theorem 12.1.4: Suppose that AF.1 holds and $\mathbf{d} \in \mathcal{D}$. Then for each point $p(\tau, \mathbf{d})$ on the projected-gradient path, $p(\tau, \mathbf{d}) - \mathbf{d}$ is a solution of the problem

$$\min_{\mathbf{s}} \{\nabla g(\mathbf{d})^T \mathbf{s} : \mathbf{d} + \mathbf{s} \in \mathcal{D}, ||\mathbf{s}||_2 \leq \theta\},$$

where $\theta = ||p(\tau, \mathbf{d}) - \mathbf{d}||_2$. Furthermore, $p(\tau, \mathbf{d}) - \mathbf{d}$ is also a solution of this problem $\forall \theta \geq ||p(\tau, \mathbf{d}) - \mathbf{d}||_2$ whenever

$$-\nabla g(\mathbf{d}) \in \mathcal{N}(p(\tau, \mathbf{d})).$$

Theorem 12.1.5: Suppose that AF.1 holds and $\mathbf{d} \in \mathcal{D}$. Then

(i) $\chi(\mathbf{d}, \theta)$ is continuous and non-decreasing in θ for $\theta \geq 0$;

(ii) $\chi(\mathbf{d}, \theta)/\theta$ is non-increasing in θ for $\theta > 0$.

(iii) for any $\mathbf{s}$ such that $\mathbf{d} + \mathbf{s} \in \mathcal{D}$, the inequality

$$\chi(\mathbf{d}, \theta) \leq |\nabla g(\mathbf{d})^T \mathbf{s}| + 2\theta \left|\left|\mathcal{P}_{\mathcal{T}(\mathbf{d}+\mathbf{s})}[-\nabla g(\mathbf{d})]\right|\right|$$

holds for all $\theta > ||\mathbf{s}||_2$.

The conditions that Conn et al. (2000) want from the Approximate Generalised Cauchy Point are that both

$$\left.\begin{array}{c} ||\mathbf{s}_j(\tau)||_2 \leq \Delta_j \\ \text{and} \\ \hat{r}_j(p(\tau, \mathbf{d}_j)) \leq \hat{r}_j(\mathbf{d}_j) + \kappa_{ubs} \nabla g(\mathbf{d}_j)^T \mathbf{s}_j(\tau) \end{array}\right\} \quad \text{(C.1)}$$

are satisfied, and at least one of

$$\left.\begin{array}{c} ||\mathbf{s}_j(\tau)||_2 \geq \kappa_{frd} \Delta_j \\ \text{or} \\ \hat{r}_j(p(\tau, \mathbf{d}_j)) \geq \hat{r}_j(\mathbf{d}_j) + \kappa_{lbs} \nabla g(\mathbf{d}_j)^T \mathbf{s}_j(\tau) \\ \text{or} \\ ||\mathcal{P}_{\mathcal{T}(p(\tau, \mathbf{d}_j))}[-\nabla g(\mathbf{d}_j)]||_2 \leq \kappa_{epp} \frac{|\nabla g(\mathbf{d}_j)^T \mathbf{s}_j(\tau)|}{\Delta_j} \end{array}\right\} \quad \text{(C.2)}$$

are satisfied, where

$$0 < \kappa_{ubs} < \kappa_{lbs} < 1, \qquad \kappa_{frd} \in (0, 1), \qquad \kappa_{epp} \in (0, 1/2).$$

The algorithm for finding a point that satisfies is described in Algorithm 12.2.2 in (Conn et al., 2000) (pages 455-456). For details see Algorithm C.1. This algorithm is guaranteed to end in finitely many iterations.

Theorem 12.2.1: Suppose AF.1, $\hat{r}_j$ is twice continuously differentiable on $\mathcal{B}_j$ and a first order constraint qualification hold. Then Algorithm C.1 terminates in a finite number of iterations.

Theorem 12.2.2: Suppose that the step $\mathbf{s}_j(\tau)$ satisfies (C.1) and (C.2). Then

$$\hat{r}_j(\mathbf{d}_j) - \hat{r}_j(\mathbf{d}_j + \mathbf{s}_j(\tau)) \geq \mu \chi(\mathbf{d}_j) \min\left\{ \frac{\chi(\mathbf{d}_j)}{\beta_j}, \Delta_j, 1 \right\},$$

where $\mu = \min\{\kappa_{ubs}\kappa_{frd}/2, 2\kappa_{ubs}(1 - \kappa_{lbs})\} \in (0, 1)$.

Algorithm C.1 Generating an Approximate Generalised Cauchy Point

```
1: $\Delta_j$, $\mathbf{d}_j \in \mathcal{D}$, model $\hat{r}_j$ are given as are $\kappa_{epp}, \kappa_{lbs}, \kappa_{ubs}, \kappa_{frd}$
2: Set $\tau_{\min} = 0, \tau_{\max} = \infty, \tau_0 = \Delta_j / ||\widehat{\nabla} g_j(\mathbf{d}_j)||_2$, $i = 0$ and FOUND=FALSE
3: repeat
4:     $p(\tau_i, \mathbf{d}_j) \leftarrow \mathcal{P}_{\mathcal{D}}[\mathbf{d}_j - \tau_i \widehat{\nabla} g_j(\mathbf{d}_j)]$ and $\mathbf{s}_j(\tau_i) \leftarrow p(\tau_i, \mathbf{d}_j) - \mathbf{d}_j$
5:     Evaluate $\hat{r}_j(\mathbf{d}_j + \mathbf{s}_j(\tau_i))$
6:     if one of (C.1) is violated then
7:         $\tau_{\max} \leftarrow \tau_i$ and go to Line 13
8:     else if all (C.2) are violated then
9:         $\tau_{\min} \leftarrow \tau_i$ and go to Line 13
10:    else
11:        $\mathbf{s}^{GC} \leftarrow \mathbf{s}_j(\tau_i)$, FOUND $\leftarrow$ TRUE and go to Line 19
12:    end if
13:    if $\tau_{\max} = \infty$ then
14:        $\tau_{i+1} \leftarrow 2\tau_i$
15:    else
16:        $\tau_{i+1} \leftarrow (\tau_{\min} + \tau_{\max})/2$
17:    end if
18:    $i \leftarrow i + 1$
19: until FOUND is TRUE
```

Appendix D

Further Results from Chapter 7

This appendix presents the remaining results on the experiments reported in Chapter 7. Table D.1 shows the sensitivity (defined in Equation (7.4.3)) of each disruption to the deviations defined by changing the phase number of the Erlang distribution and the autocorrelation of the runway service times. Figures D.1 and D.2 show the P-P plots and the rejection frequency of each test statistic for the Day 4 problem, which is based on an aircraft with technical problems at 13:20. These follow a similar pattern to Figures 7.4.1 and 7.4.2. Figures D.3 and D.4 show the P-P plots and the rejection frequency of each test statistic for the Day 5 problem, which is based on a small hub airport closure at 10:10. This effects two aircraft. These follow a similar pattern to Figures 7.4.3 and 7.4.4.

Table D.1: Sensitivity of each system in each disruption scenario.

		Sensitivity (4dp)				
Phases	Autocorrelation	Day 1	Day 2	Day 4	Day 5	Day 6
1000	0.5	-0.0002	0.0005	0.0001	-0.0002	0.0002
	0.9	-0.0000	0.0012	0.0004	0.0003	0.0008
100	0	-0.0003	0.0009	0.0007	0.0007	0.0004
	0.5	0.0005	0.0026	0.0014	0.0011	-0.0026
	0.9	0.0014	0.0089	0.0050	0.0047	-0.0012
10	0	0.0027	0.0067	0.0052	0.0050	-0.0084
	0.5	0.0056	0.0165	0.0124	0.0110	0.0030
	0.9	0.0293	0.0883	0.0591	0.0889	0.0841
2	0	0.0094	0.0192	0.0141	0.0137	0.0072
	0.5	0.0235	0.0563	0.0415	0.0464	0.0495
	0.9	0.2182	0.7306	0.4233	1.0484	0.5506

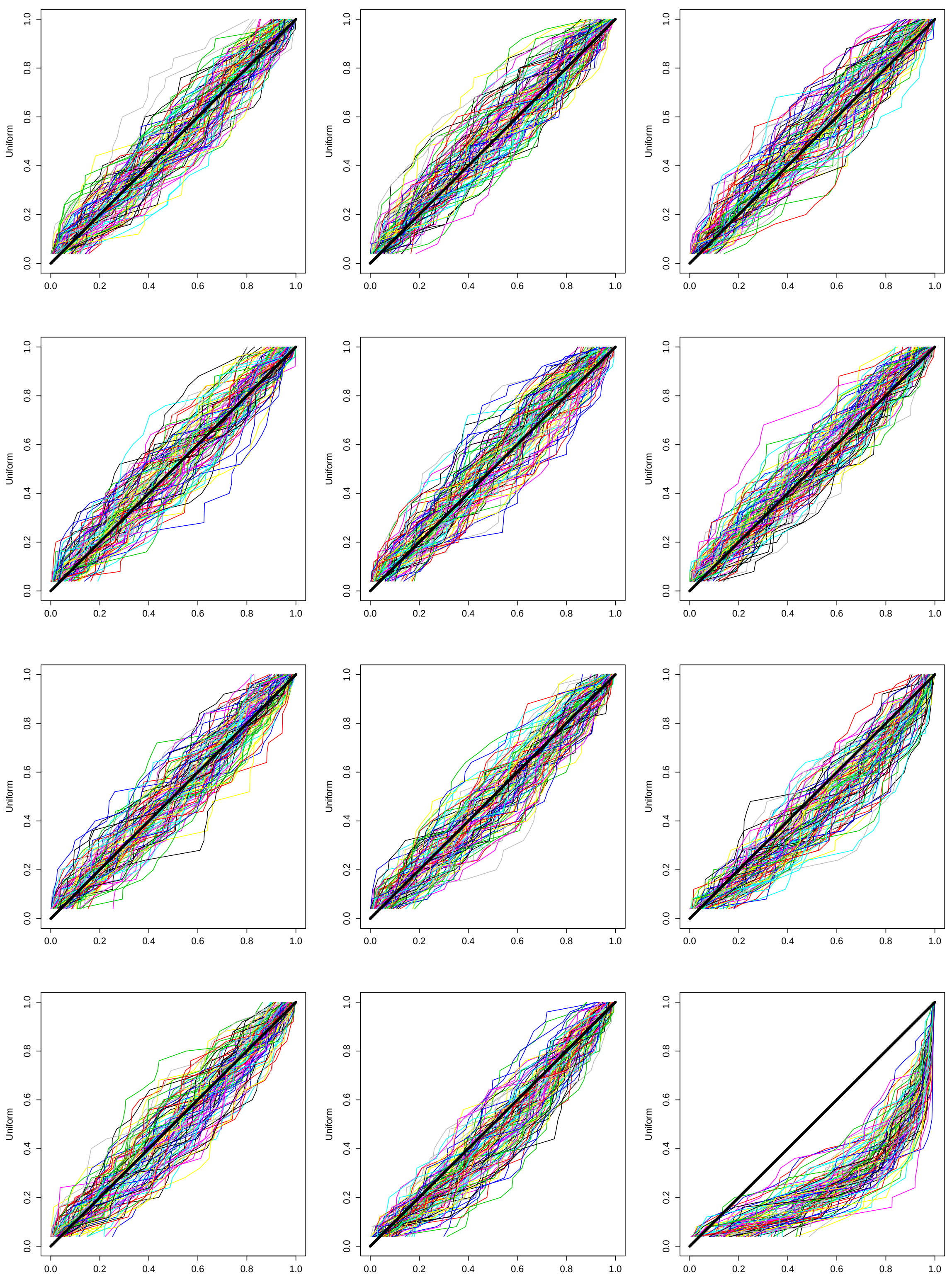

Figure D.1: Day 4 P-P plots of the sets $\{U_j^{k\rho}\}_{j=1}^{J}$, for each (k, ρ). Rows are k=1000, 100, 10, 2, columns are $\rho = 0, 0.5, 0.9$.

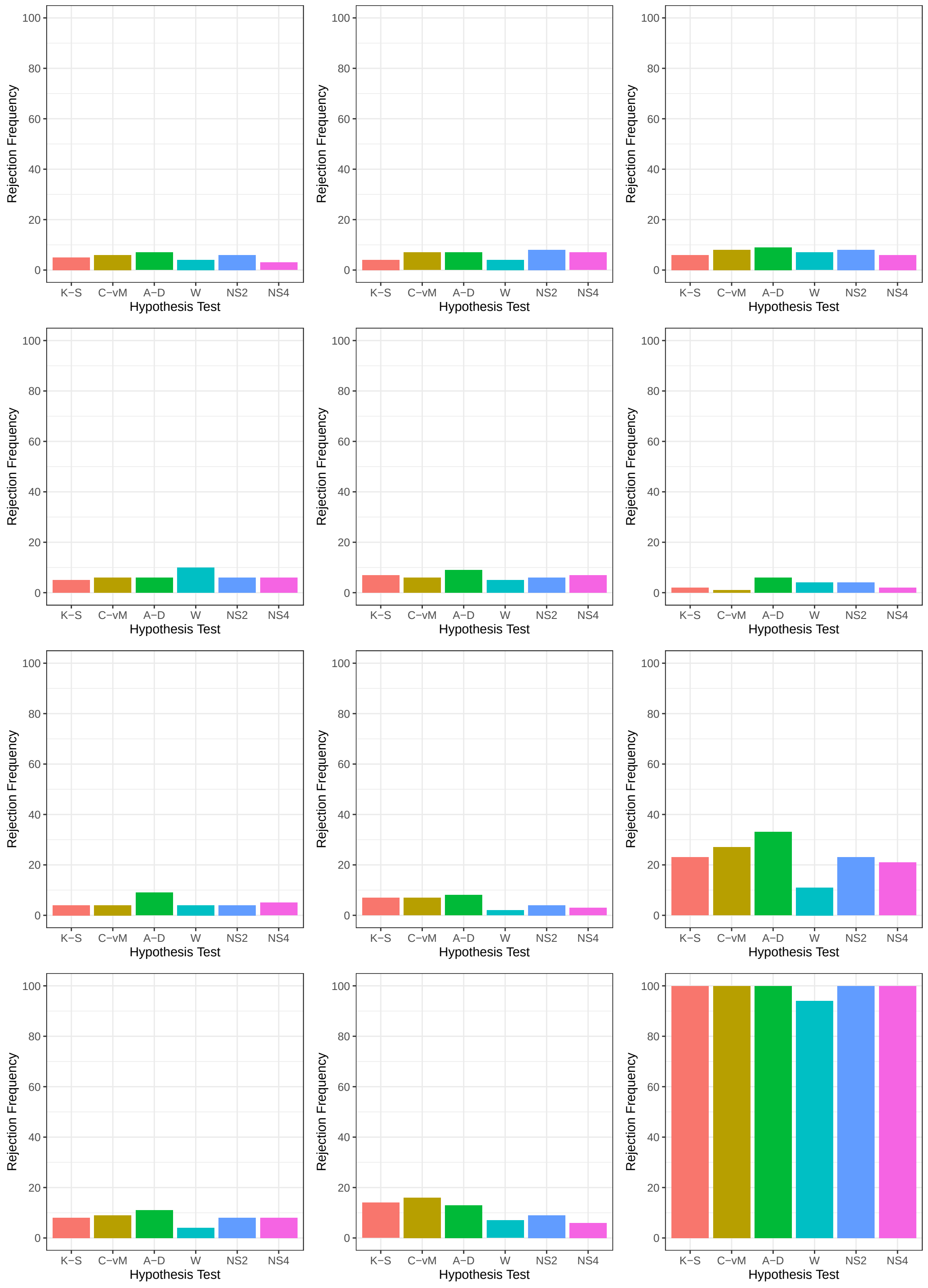

Figure D.2: Number of rejections for each hypothesis test in each system (k, ρ) in the Day 4 disruption. Rows are $k = 1000, 100, 10, 2$, columns are $\rho = 0, 0.5, 0.9$.

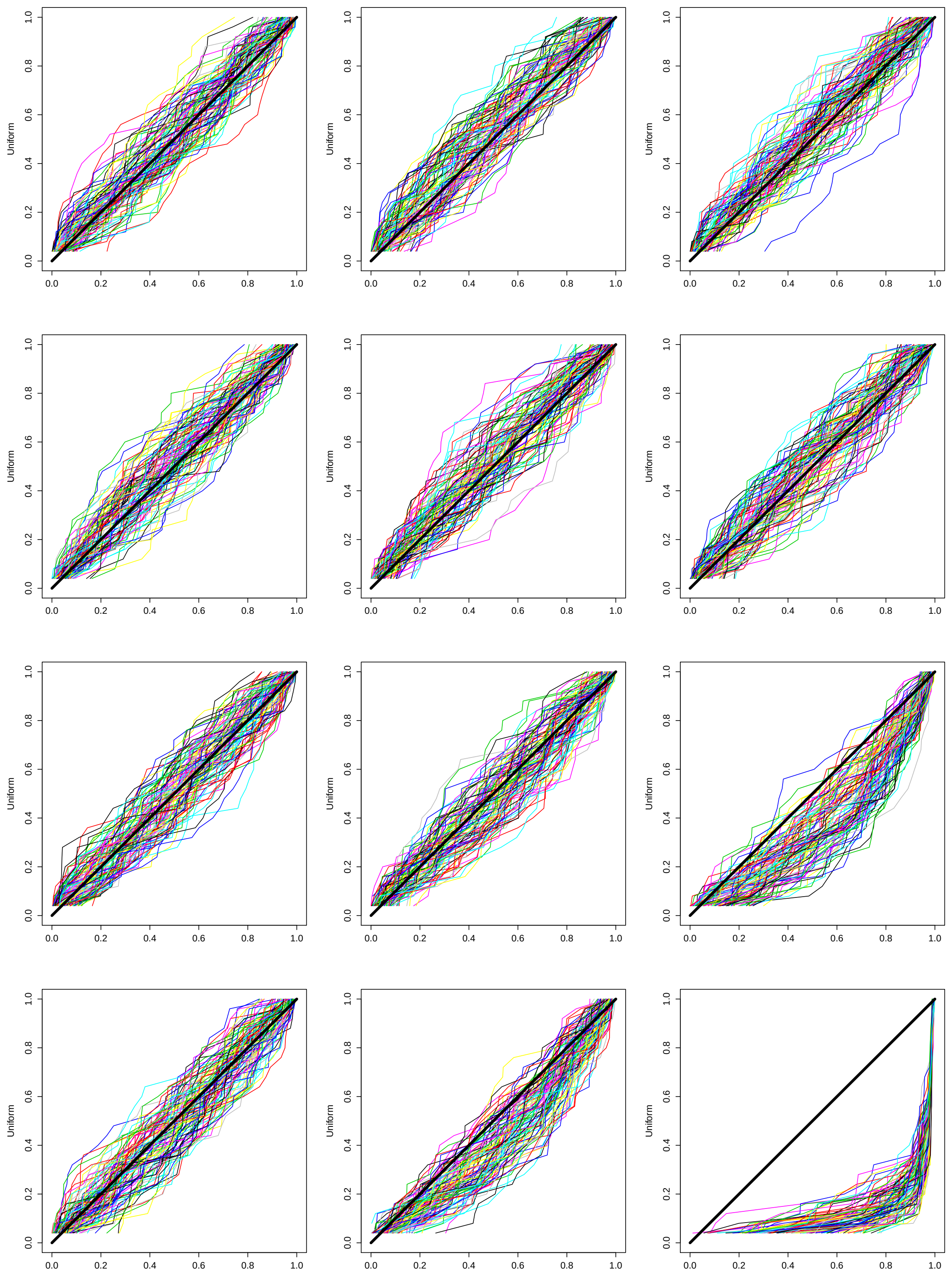

Figure D.3: Day 5 P-P plots of the sets $\{U_j^{k\rho}\}_{j=1}^{J}$, for each (k, ρ). Rows are k=1000, 100, 10, 2, columns are $\rho = 0, 0.5, 0.9$.

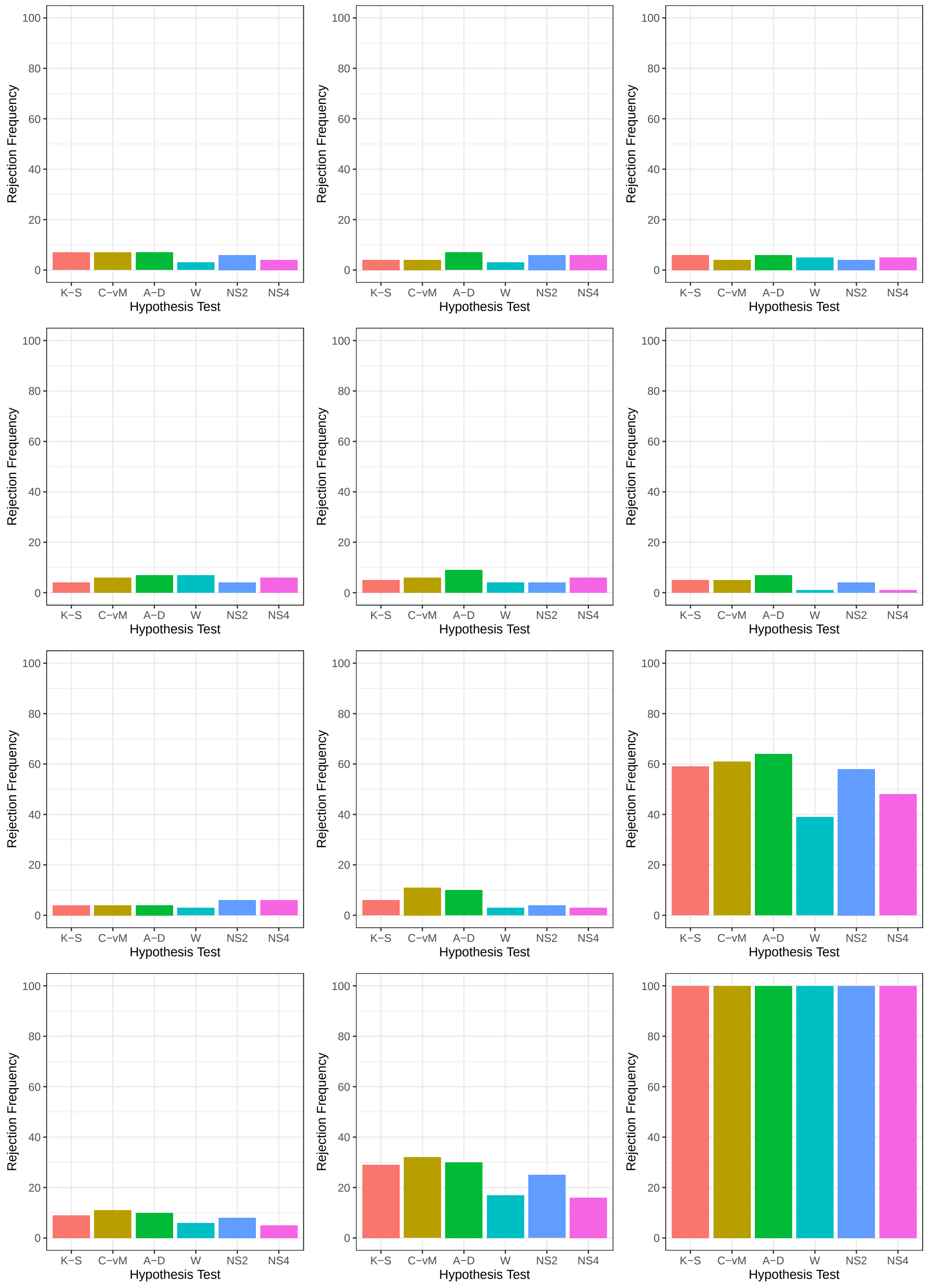

Figure D.4: Number of rejections for each hypothesis test in each system (k, ρ) in the Day 5 disruption. Rows are $k = 1000, 100, 10, 2$, columns are $\rho = 0, 0.5, 0.9$.

Bibliography

Abdelghany, K. F., Abdelghany, A. F., and Ekollu, G. (2008). An Integrated Decision Support Tool for Airlines Schedule Recovery during Irregular Operations. *European Journal of Operational Research*, 185(2):825–848.

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`.

Amaran, S., Sahinidis, N. V., Sharda, B., and Bury, S. J. (2016). Simulation Optimization: A Review of Algorithms and Applications. *Annals of Operations Research*, 240(1):351–380.

Andradóttir, S. (2014). A Review of Random Search Methods. In Fu, M. C., editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research and Management Science*, chapter 10, pages 277–292. Springer, New York. Retrieved from `https://ebookcentral.proquest.com` (visited 19/12/2019).

Ansari, Q. H., Köbis, E., and Uao, J.-C. (2018). *Vector Variational Inequalites and*

Vector Optimization: Theory and Applications. Springer International Publishing AG, Cham, Switzerland.

Arias, P., Mujica Mota, M., Guimarans, D., and Boosten, G. (2013). A Methodology Combining Optimization and Simulation for Real Applications of the Stochastic Aircraft Recovery Problem. In *Proceedings - 8th EUROSIM Congress on Modelling and Simulation, EUROSIM 2013*, pages 265–270, Washington, DC, USA. IEEE Computer Society.

Avramidis, A. N., Chan, W., Gendreau, M., L'Ecuyer, P., and Pisacane, O. (2010). Optimizing Daily Agent Scheduling in a Multiskill Call Center. *European Journal of Operational Research*, 200(3):822–832.

Aydt, H., Cai, W., Turner, S. J., and Gan, B. P. (2011). Symbiotic Simulation for Optimisation of Tool Operations in Semiconductor Manufacturing. In S. Jain et al., editor, *Proceedings of the 2011 Winter Simulation Conference*, pages 2088–2099, Piscataway, New Jersey. IEEE.

Aydt, H., Lees, M., and Knoll, A. (2012). Symbiotic Simulation for Future Electro-mobility Transportation Systems. In C. Laroque et al., editor, *Proceedings of the 2012 Winter Simulation Conference*, pages 1–12, Piscataway, New Jersey. IEEE.

Aydt, H., Turner, S. J., Cai, W., and Low, M. Y. H. (2008a). Symbiotic Simulation Systems: An Extended Definition Motivated by Symbiosis in Biology. In *Proceedings - 22nd Workshop on Principles of Advanced and Distributed Simulation, PADS*, pages 109–116, Los Alamitos, CA. IEEE Computer Society.

Aydt, H., Turner, S. J., Cai, W., and Low, M. Y. H. (2009). Research Issues in Symbiotic Simulation. In M. D. Rossetti et al., editor, *Proceedings of the 2009 Winter Simulation Conference*, pages 1213–1222, Piscataway, New Jersey. IEEE.

Aydt, H., Turner, S. J., Cai, W., Low, M. Y. H., Lendermann, P., Gan, B. P., and Ayani, R. (2008b). Preventive What-if Analysis in Symbiotic Simulation. In S. J. Mason et al., editor, *Proceedings of the 2008 Winter Simulation Conference*, pages 750–758, Piscataway, New Jersey. IEEE.

Ball, R. C., Branke, J., and Meisel, S. (2018). Optimal Sampling for Simulated Annealing Under Noise. *INFORMS Journal on Computing*, 30(1):200–215.

Bang, J.-Y. and 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.

Bard, J. F., Yu, G., and Arguello, M. F. (2001). Optimizing Aircraft Routings in Response to Groundings and Delays. *IIE Transactions*, 33(10):931–947.

Barnhart, C. and Smith, B. C., editors (2012). *Quantitative Problem Solving Methods in the Airline Industry: A Modeling Methodology Handbook*, volume 169 of *International Series in Operations Research and Management Science*. Springer Science+Business Media, LLC, New York.

Barton, R. R. (2015). Tutorial: Simulation Metamodeling. In L. Yilmaz et al., editor, *Proceedings of the 2015 Winter Simulation Conference*, pages 1765–1779, Piscataway, New Jersey. IEEE.

Bennell, J. A., Mesgarpour, M., and Potts, C. N. (2017). Dynamic Scheduling of Aircraft Landings. *European Journal of Operational Research*, 258:315–327.

Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J. (2009). A Survey on Metaheuristics for Stochastic Combinatorial Optimization. *Natural Computing*, 8(2):239–287.

Boesel, J., Nelson, B. L., and Kim, S.-H. (2003). Using Ranking and Selection to "Clean Up" After Simulation Optimization. *Operations Research*, 51(5):814–825.

Box, G. E. P. and Wilson, K. B. (1951). On the Experimental Attainment of Optimum Conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1):1–38.

Bratu, S. and Barnhart, C. (2006). Flight Operations Recovery: New Approaches Considering Passenger Recovery. *Journal of Scheduling*, 9(3):279–298.

Brent, R. P. (1973). *Algorithms for Minimzation without Derivatives.* Prentice-Hall Series in Automatic Computation. Prentice-Hall, Inc, Englewood Cliffs, N.J.

Broyden, C. (1970). The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics (Institute of Mathematics and Its Applications)*, 6(1):76–90.

Can, B., Beham, A., and Heavey, C. (2008). A Comparative Study of Genetic Algorithm Components in Simulation-based Optimisation. In S. J. Mason et al., editor, *Proceedings of the 2008 Winter Simulation Conference*, pages 1829–1837, Piscataway, New Jersey. IEEE.

Can, B. and Heavey, C. (2012). A Comparison of Genetic Programming and Artificial Neural Networks in Metamodeling of Discrete-event Simulation Models. *Computers and Operations Research*, 39(2):424–436.

Cario, M. C. and Nelson, B. L. (1996). Autoregressive to Anything: Time-series Input Processes for Simulation. *Operations Research Letters*, 19(2):51–58.

Chang, K.-H. (2015). Improving the Efficiency and Efficacy of Stochastic for Simulation Optimization. *IEEE Transactions on Automatic Control*, 60(5):1235–1243.

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.

Chang, K.-H., Li, M. K., and Wan, H. (2014). Combining STRONG with Screening Designs for Large-scale Simulation Optimization. *IIE Transactions*, 46:357–373.

Chau, M. and Fu, M. C. (2014). An Overview of Stochastic Approximation. In Fu, M. C., editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research and Management Science*, chapter 6, pages 149–178. Springer, New York. Retrieved from `https://ebookcentral.proquest.com` (visited 23/12/2019).

Chau, M., Fu, M. C., Qu, H., and Ryzhov, I. O. (2014). Simulation Optimization: A Tutorial Overview and Recent Developments in Gradient-based Methods. In A. Tolk et al., editor, *Proceedings of the 2014 Winter Simulation Conference*, pages 21–35, Piscataway, New Jersey. IEEE.

Cheng, R. (2007). Determining Efficient Simulation Run Lengths for Real Time Decision Making. In S. G. Henderson et al., editor, *Proceedings of the 2007 Winter Simulation Conference*, pages 340–345, Piscataway, New Jersey. IEEE.

Chiroma, E., Higgins, M., and Chandler, A. (2018). Implementation of a Data-centric Symbiotic Simulation Technique for Enhancing Ford PTME Throughtput Simulation Processes. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 4105–4106, Piscataway, New Jersey. IEEE.

Civil Aviation Authority (2015). Your Rights When You Fly. Accessed July 23rd , 2018. `https://www.caa.co.uk/Passengers/Resolving-travel-problems/Delays-cancellations/Your-rights/Your-rights-when-you-fly/`.

Civil Aviation Authority (2019). Punctuality Statistics 2018. Accessed January 12th , 2020. `https://www.caa.co.uk/Data-and-analysis/UK-aviation-market/Flight-reliability/Datasets/Punctuality-data/Punctuality-statistics-2018/`.

Clausen, J., Larsen, A., Larsen, J., and 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., Sartenaer, A., and Toint, P. L. (1993). Global Convergence of a Class of Trust Region Algorithms for Optimization using Inexact Projections on Convex Constraints. *SIAM Journal on Numerical Analysis*, 3(1):164–221.

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.

Cook, A. and Tanner, G. (2015). European Airline Delay Cost Reference Values - Updated and Extended Values. Technical Report V4.1, University of Westminster.

Cook, R. D. and Nachtsheim, C. J. (1980). A Comparison of Algorithms for Constructing Exact D-Optimal Designs. *Technometrics*, 22(3):315–324.

Deng, G. and Ferris, M. C. (2006). Adaptation of the UOBYQA Algorithm for Noisy Functions. In L. F. Perrone et al., editor, *Proceedings of the 2008 Winter Simulation Conference*, pages 312–319, Piscataway, New Jersey. IEEE.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking Optimization Software with Performance Profile. *Mathematical Programming Series A*, 91:199–206.

EUROCONTROL (2018). *Standard Inputs for EUROCONTROL Cost Benefit Analyses.* Brussels, 8.0 edition.

Fanchao, Z., Turner, S. J., and Aydt, H. (2009). Symbiotic Simulation Control in Supply Chain of Lubricant Additive Industry. In *Proceedings - 2009 13th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 165–172, Washington, DC, USA. IEEE Computer Society.

Fedorov, V. V. (1972). *Theory of Optimal Experiments.* (W.J. Studden and E.M.

Klimko, Trans.). Probability and Mathematical Statistics, 12. Academic Press, INC., New York.

Fiems, D., Prabhu, B., and De Turck, K. (2013). Analytic Approximations of Queues with Lightly- and Heavily-correlated Autoregressive Service Times. *Annals of Operations Research*, 202:103–119.

Fletcher, R. (1970). New Approach to Variable Metric Algorithms. *Computer Journal*, 13(3):317–322.

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

Frazier, P., Powell, W., and Dayanik, S. (2009). The Knowledge-Gradient Policy for Correlated Normal Beliefs. *INFORMS Journal on Computing*, 21(4):599–613.

Frazier, P. I. (2010). Decision-Theoretic Foundations of Simulation Optimization. In J.J. Cochran et al., editor, *Wiley Encyclopedia of Operations Research and Management Science*, number 1. John Wiley & Sons, Inc.

Frazier, P. I. (2014). A Fully Sequential Elimination Procedure for Indifference-Zone Ranking and Selection with Tight Bounds on Probability of Correct Selection. *Operations Research*, 62(4):926–942.

Frazier, P. I. (2018). Bayesian Optimization. In *INFORMS TutORials in Operations Research*, pages 255–278. Published online: 19 Oct 2018. INFORMS, Maryland, USA.

Fu, M. C., editor (2014). *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research and Management Science*. Springer, New York. Retrieved from `https://ebookcentral.proquest.com` (visited 16/12/2019).

Fujimoto, R., Lunceford, D., Page, E., and Uhrmacher, A. M. (editors) (2002). Grand Challenges for Modeling and Simulation: Dagstuhl Report. Technical Report 350, Seminar No. 02351. Schloss, Dagstuhl.

Fujimoto, R. M., Celik, N., Damgacioglu, H., Hunter, M., Jin, D., Son, Y.-J., and Xu, J. (2016). Dynamic Data Driven Application Systems for Smart Cities and Urban Infrastructures. In T. M. K. Roeder et al., editor, *Proceedings of the 2016 Winter Simulation Conference*, pages 1143–1157, Piscataway, New Jersey. IEEE.

Galil, Z. and Kiefer, J. (1980). Time- and Space-saving Computer Methods, Related to Mitchell's DETMAX, for Finding D-Optimum Designs. *Technometrics*, 22(3):301–313.

Ghasemi, A., Heavey, C., and Laipple, G. (2018). A Review of Simulation-optimization Methods with Applications to Semiconductor Operational Problems. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 3672–3683, Piscataway, New Jersey. IEEE.

Glankwamdee, W., Linderoth, J., Shen, J., Connard, P., and Hutton, J. (2008). Combining Optimization and Simulation for Strategic and Operational Industrial Gas Production and Distribution. *Computers and Chemical Engineering*, 32(11):2536–2546.

Goldfarb, D. (1970). A Family of Variable-metric Methods Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26.

Gonzalez-Martin, S., Juan, A., Riera, D., Elizondo, M., and Ramos, J. (2018). A Simheuristic Algorithm for Solving the Arc Routing Problem with Stochastic Demands. *Journal of Simulation*, 12(1):53–66.

Goudarzi, H., Budeanu, D., Coles, H., Flint, P., de Gheldere, C., King, B., Markou, C., Oliver, J., Price, A., Simard, M.-C., Touraine, S., Capois, A., Girard, C., and Shores, M. (2019). Data Science Hype or Ripe for Aviation? International Air Transport Association (IATA) Aviation Data White Paper Series.

Guimarans, D., Arias, P., and Mujica Mota, M. (2015). Large Neighbourhood Search and Simulation for Disruption Management in the Airline Industry. In M. Mujica Mota et al., editor, *Applied Simulation and Optimization: In Logistics, Industrial and Aeronautical Practice*, pages 169–201. Springer International Publishing, Cham.

Gunn, W. A. (1964). Airline System Simulation. *Operations Research*, 12(2):206–229.

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

Haimes, Y., Lasdon, L. S., and 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.

Hashemi, H., Abdelghany, K. F., and Abdelghany, A. F. (2017). A Multi-agent Learning Approach for Online Calibration and Consistency Checking of Real-time Traffic Network Management Systems. *Transportmetrica B*, 5(3):369–389.

Hatami, S., Calvet, L., Fernández-Viagas, V., Framiñán, J. M., and Juan, A. A. (2018). A Simheuristic Algorithm to Set Up Starting Times in the Stochastic Parallel Flowshop Problem. *Simulation Modelling Practice and Theory*, 86:55–71.

Hong, L. J. and Nelson, B. L. (2006). Discrete Optimization via Simulation using COMPASS. *Operations Research*, 54(1):115–129.

Hong, L. J. and Nelson, B. L. (2009). A Brief Introduction to Optimisation via Simulation. In *Proceedings of the 2009 Winter Simulation Conference*, pages 75–85, Piscataway, New Jersey. IEEE.

Hong, L. J., Nelson, B. L., and Xu, J. (2014). Discrete Optimization via Simulation. In Fu, M. C., editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research and Management Science*, chapter 2, pages 9–44. Springer, New York. Retrieved from `https://ebookcentral.proquest.com` (visited 19/12/2019).

Hu, Y., Liao, H., Zhang, S., and Song, Y. (2017). Multiple Objective Solution Approaches for Aircraft Rerouting Under the Disruption of Multi-aircraft. *Expert Systems With Applications*, 83:283–299.

Hu, Y., Xu, B., Bard, J. F., Chi, H., and Gao, M. (2015). Optimization of Multifleet Aircraft Routing Considering Passenger Transiting under Airline Disruption. *Computers and Industrial Engineering*, 80:132–144.

Huang, D., Allen, T. T., Notz, W. I., and Miller, R. A. (2006a). Sequential Kriging Optimization using Multiple-fidelity Evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382.

Huang, D., Allen, T. T., Notz, W. I., and Zeng, N. (2006b). Global Optimization of Stochastic Black-box Systems via Sequential Kriging Meta-models. *Journal of Global Optimization*, 34(3):441–466.

Huang, Y., Seck, M. D., and Verbraeck, A. (2010). Towards Automated Model Calibration and Validation in Rail Transit Simulation. *Procedia Computer Science*, 1(1):1259–1265.

Huang, Y. and Verbraeck, A. (2009). A Dynamic Data-driven Approach for Rail Transport System Simulation. In M. D. Rossetti et al., editor, *Proceedings of the 2009 Winter Simulation Conference*, pages 2553–2562, Piscataway, New Jersey. IEEE.

Hutchison, D. W. and Hill, S. D. (2001). Simulation Optimization of Airline Delay with Constraints. In B. A. Peters et al., editor, *Proceedings of the 2001 Winter Simulation Conference*, pages 1017–1022, Piscataway, New Jersey. IEEE.

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

ICAO (2016). European Guidance Material on All Weather Operations. Technical Report September, 5th edition, International Civil Aviation Organitzation.

Inanlouganji, A., Pedrielli, G., Fainekos, G., and Pokutta, S. (2018). Continuous Simulation Optimization with Model Mismatch using Gaussian Process Regression. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 2131–2142, Piscataway, New Jersey. IEEE.

Izady, N. and 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.

Jalali, H., Van Nieuwenhuyse, I., and Picheny, V. (2017). Comparison of Kriging-based Algorithms for Simulation Optimization with Heterogeneous Noise. *European Journal of Operational Research*, 261(1):279–301.

Jeng, C.-R. (2012). Real-time Decision Support for Airline Schedule Disruption Management. *African Journal of Business Management*, 6(27):8071–8079.

Jian, N. and Henderson, S. G. (2015). An Introduction to Simulation Optimization. In L. Yilmaz et al., editor, *Proceedings of the 2015 Winter Simulation Conference*, pages 1780–1794, Piscataway, New Jersey. IEEE.

Jimenez Serrano, F. J. and Kazda, A. (2017). Airline Disruption Management: Yesterday, Today and Tomorrow. *Transportation Research Procedia*, 28:3–10.

Johnson, M. E. and Jackman, J. (1989). Infinitesimal Perturbation Analysis: A Tool for Simulation. *The Journal of the Operational Research Society*, 40(3):243–254.

Jones, D. R., Schonlau, M., and W. J. Welch (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., and Figueira, G. (2015). A Review of Simheuristics: Extending Metaheuristics to Deal with Stochastic Combinatorial Optimization Problems. *Operations Research Perspectives*, 2:62–72.

Juan, A. A., Kelton, W. D., Currie, C. S., and Faulin, J. (2018). Simheuristic Applications: Dealing with Uncertainty in Logistics, Transportation, and Other Supply Chain Areas. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 3048–3059, Piscataway, New Jersey. IEEE.

Kim, S., Pasupathy, R., and Henderson, S. G. (2014). A Guide to Sample Average Approximation. In Fu, M. C., editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research and Management Science*, chapter 8, pages 207–243. Springer, New York. Retrieved from `https://ebookcentral.proquest.com` (visited 22/12/2019).

Kim, S.-H. and Nelson, B. L. (2001). A Fully Sequential Procedure for Indifference-Zone Selection in Simulation. *ACM Transactions on Modeling and Computer Simulation*, 11(3):251–273.

Kleijnen, J. (1998). Experimental Design for Sensitivity Analysis, Optimization, and Validation of Simulation Models. In Banks, J., editor, *Handbook of Simulation - Principles, Methodology, Advances, Applications, and Practice*, chapter 6, pages 173–223. John Wiley & Sons, New York.

Kleijnen, J. P. (2005). An Overview of the Design and Analysis of Simulation Experiments for Sensitivity Analysis. *European Journal of Operational Research*, 164(2):287–300.

Kleijnen, J. P. C. (2014). Response Surface Methodology. In Fu, M. C., editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research and Management Science*, chapter 4, pages 81–104. Springer, New York. Retrieved from `https://ebookcentral.proquest.com` (visited 24/12/2019).

Kohl, N., Larsen, A., Larsen, J., Ross, A., and Tiourine, S. (2007). Airline Disruption Management - Perspectives, Experiences and Outlook. *Journal of Air Transport Management*, 13(3):149–162.

Lapp, M., AhmadBeygi, S., Cohn, A., and Tsimhoni, O. (2008). A Recursion-based Approach to Simulating Airline Schedule Robustness. In S. J. Mason et al., editor, *Proceedings of the 2008 Winter Simulation Conference*, pages 2661–2667, Piscataway, New Jersey. IEEE.

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.

Lee, L. H., Lee, C. U., and Tan, Y. P. (2007). A Multi-objective Genetic Algorithm for Robust Flight Scheduling using Simulation. *European Journal of Operational Research*, 177(3):1948–1968.

Li, Y. and Fu, M. (2018). Sequential First-order Response Surface Methodology Augmented with Direct Gradients. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 2143–2154, Piscataway, New Jersey. IEEE.

Liang, Z., Xiao, F., Qian, X., Zhou, L., Jin, X., Lu, X., and 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.

Linz, D. D., Huang, H., and Zabinsky, Z. B. (2017). Multi-fidelity Simulation Optimization with Level Set Approximation using Probabilistic Branch and Bound. In W. K. V. Chan et al., editor, *Proceedings of the 2017 Winter Simulation Conference*, pages 2057–2068, Piscataway, New Jersey. IEEE.

Liu, T. K., Chen, C. H., and Chou, J. H. (2010). Optimization of Short-haul Aircraft Schedule Recovery Problems using a Hybrid Multiobjective Genetic Algorithm. *Expert Systems with Applications*, 37(3):2307–2315.

Loève, M. (1977). *Probability Theory I*, volume 45 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, NY, 4th edition.

Løve, M., Sørensen, K. R., Larsen, J., and 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. and Nachtsheim, C. J. (1995). The Coordinate-exchange Algorithm for Constructing Exact Optimal Experimental Designs. *Technometrics*, 37(1):60–69.

Miller, Jr., F. L. and Quesenberry, C. P. (1979). Power Studies of Tests for Uniformity, II. *Communications in Statistics - Simulation and Computation*, 8(3):271–290.

Mitchell, T. J. (1974). An Algorithm for the Construction of "D-Optimal" Experimental Designs. *Technometrics*, 16(2):203–210.

Montgomery, D. C. (2009). *Design and Analysis of Experiments*. John Wiley and Sons (Asia) Pte Ltd, 7th edition.

Mori, R. (2015). Development of Fast-time Stochastic Airport Ground and Runway Simulation Model and its Traffic Analysis. *Mathematical Problems in Engineering*, 2015:1–11.

Mujica Mota, M., Boosten, G., De Bock, N., Jimenez, E., and de Sousa, J. P.

(2017). Simulation-based Turnaround Evaluation for Lelystad Airport. *Journal of Air Transport Management*, 64:21–32.

Nelson, B. L. (2013). *Foundations and Methods of Stochastic Simulation: A First Course*, volume 187 of *International Series in Operations Research and Management Science*. Springer, NY.

Nelson, B. L. (2014). Optimization via Simulation Over Discrete Decision Variables. In *INFORMS Tutorials in Operations Research*, chapter 9, pages 193–207. Institute for Operations Research and the Management Sciences (INFORMS), Maryland, USA.

Nelson, B. L. (2016). 'Some Tactical Problems in Digital Simulation' for the Next 10 Years. *Journal of Simulation*, 10(1):2–11.

Nocedal, J. and Wright, J. (2006). *Numerical Optimization*. Springer-Verlag, New York, NY, second edition.

Oakley, D., Onggo, B. S., and Worthington, D. (2020). Symbiotic simulation for the operational management of inpatient beds: model development and validation using Δ-method. *Health Care Management Science*, 23:153–169.

Onggo, B. S., Mustafee, N., Smart, A., Juan, A. A., and Molloy, O. (2018). Symbiotic Simulation System: Hybrid Systems Model Meets Big Data Analytics. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 1358–1369, Piscataway, New Jersey. IEEE.

Onggo, B. S., Panadero, J., Corlu, C. G., and 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. and Bierlaire, M. (2013). A Simulation-Based Optimization Framework for Urban Transportation Problems. *Operations Research*, 61(6):1333–1345.

Osorio, C. and Chong, L. (2015). A Computationally Efficient Simulation-Based Optimization Algorithm for Large-Scale Urban Transportation Problems. *Transportation Science*, 49(3):623–636.

Osorio, C. and Selvam, K. K. (2017). Simulation-based Optimization: Achieving Computational Efficiency Through the use of Multiple Simulators. *Transportation Science*, 51(2):395–411.

Palhazi Cuervo, D., Goos, P., and Sörensen, K. (2016). Optimal Design of Large-scale Screening Experiments: A Critical Look at the Coordinate-exchange Algorithm. *Statistics and Computing*, 26(1-2):15–28.

Papathanasopoulou, V., Markou, I., and Antoniou, C. (2016). Online Calibration for Microscopic Traffic Simulation and Dynamic Multi-step Prediction of Traffic Speed. *Transportation Research Part C: Emerging Technologies*, 68:144–159.

Pearce, M. and Branke, J. (2017). Bayesian Simulation Optimization with Input Uncertainty. In W. K. V. Chan et al., editor, *Proceedings of the 2017 Winter Simulation Conference*, pages 2740–2751, Piscataway, New Jersey. IEEE.

Pei, L., Nelson, B. L., and Hunter, S. (2018). A New Framework for Parallel Ranking & Selection using an Adaptive Standard. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 2201–2212, Piscataway, New Jersey. IEEE.

Petersen, J. D., Sölveling, G., Clarke, J.-P., Johnson, E. L., and Shebalov, S. (2012). An Optimization Approach to Airline Integrated Recovery. *Transportation Science*, 46(4):482–500.

Quansheng, L., DongQing, J., Peng, Z., and TingWei, M. (2013). Research on the Disrupted Airline Scheduling. In *Proceedings of the 10th International Conference on Service Systems and Service Management*, pages 332–336.

Quesenberry, C. P. and Miller, Jr., F. L. (1977). Power Studies Of Some Tests for Uniformity. *Journal of Statistical Computation and Simulation*, 5(3):169–191.

Rhodes-Leader, L. (2020). `www.lancaster.ac.uk/pg/rhodesle/PhD_data/PhD_simulation_input_data.php`.

Rosenberger, J. M., Johnson, E. L., and 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., and Nemhauser, G. L. (2000). SIMAIR: A Stochastic Model of Airline Operations. In J. A. Joines et al., editor, *Proceedings of the 2000 Winter Simulation Conference*, pages 1118–1122, Piscataway, New Jersey. IEEE.

Rosenberger, J. M., Schaefer, A. J., Goldsman, D., Johnson, E. L., Kleywegt, A. J., and Nemhauser, G. L. (2002). A Stochastic Model of Airline Operations. *Transportation Science*, 36(4):357–377.

Salemi, P., Song, E., Nelson, B. L., and Staum, J. (2019). Gaussian Markov Random Fields for Discrete Optimization via Simulation: Framework and Algorithms. *Operations Research*, 67(1):250–266.

Santos, B. F., Wormer, M. M., Achola, T. A., and Curran, R. (2017). Airline Delay Management Problem with Airport Capacity Constraints and Priority Decisions. *Journal of Air Transport Management*, 63:34–44.

Sarac, A., Batta, R., and Rump, C. M. (2006). A Branch-and-Price Approach for Operational Aircraft Maintenance Routing. *European Journal of Operational Research*, 175(3):1850–1869.

Scala, P., Mujica, M., and Delahaye, D. (2017). A Down to Earth Solution: Applying a Robust Simulation-optimization Approach to Resolve Aviation Problems. In W. K. V. Chan et al., editor, *Proceedings of the 2017 Winter Simulation Conference*, pages 617–631, Piscataway, New Jersey. IEEE.

Scala, P., Mujica, M., Wu, C.-L., and Delahaye, D. (2018). Sim-Opt in the Loop: Algorithmic Framework for Solving Airport Capacity Problems. In M. Rabe et al., editor, *Proceedings of the 2018 Winter Simulation Conference*, pages 2261–2272, Piscataway, New Jersey. IEEE.

Scott, W., Frazier, P., and Powell, W. (2011). The Correlated Knowledge Gradient for Simulation Optimization of Continuous Parameters using Gaussian Process Regression. *SIAM Journal on Optimization*, 21(3):996–1026.

Shanno, D. F. (1970). Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656.

Shone, R., Glazebrook, K., and Zografos, K. G. (2019). Resource Allocation in Congested Queueing Systems With Time-varying Demand: An Application to Airport Operations. *European Journal of Operational Research*, 276(2):566–581.

Sinclair, K., Cordeau, J.-F., and 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.

Spall, J. C. (1992). Multivariate Stochastic Approximation using a Simultaneous Perturbation Gradient Approximantion. *IEEE Transactions on Automatic Control*, 37(3):332–341.

Stamatopoulos, M. A., Zografos, K. G., and Odoni, A. R. (2004). A Decision Support System For Airport Strategic Planning. *Transportation Research Part C: Emerging Technologies*, 12(2):91–117.

Staum, J. (2009). Better Simulation Metamodeling: The Why, What, and How of Stochastic Kriging. In M. D. Rossetti et al., editor, *Proceedings of the 2009 Winter Simulation Conference*, pages 119–133, Piscataway, New Jersey. IEEE.

Sunderrajan, A., Viswanathan, V., Cai, W., and Knoll, A. (2016). Data Driven Adaptive Traffic Simulation of an Expressway. In T. M. K. Roeder et al., editor, *Proceedings of the 2016 Winter Simulation Conference*, pages 1194–1205, Piscataway, New Jersey. IEEE.

Teodorović, D. and Guberinić, S. (1984). Optimal Dispatching Strategy on an Airline Network After a Schedule Perturbation. *European Journal of Operational Research*, 15(2):178–182.

Thas, O. (2010). *Comparing Distributions.* Springer Series in Statistics. Springer Science+Business Media, NY.

The AnyLogic Company (2017). Anylogic 8.2.3. Accessed July 23$^{\text{rd}}$, 2018. `http://www.anylogic.com`.

Thengvall, B. G., Bard, J. F., and Yu, G. (2000). Balancing User Preferences for Aircraft Schedule Recovery During Irregular Operations. *IIE Transactions*, 32(3):181–193.

Vu, V.-A., Park, G., and Tan, G. (2013). Symbiotic Simulation for the Generation and Simulation of Incident Management Strategies. In G. Tan et al., editor, *AsiaSim 2013. Communications in Computer and Information Science, vol 402.*, pages 397–402. Springer,Berlin, Heidelberg.

Wang, D., Wu, Y., Hu, J.-Q., Liu, M., Yu, P., Zhang, C., and Wu, Y. (2019). Flight Schedule Recovery: A Simulation-based Approach. *Asia-Pacific Journal of Operational Research*, 36(6):1–19.

Wang, H., Pasupathy, R., and Schmeiser, B. W. (2013). Integer-Ordered Simulation Optimization using R-SPLINE: Retrospective Search with Piecewise-Linear Interpolation and Neighborhood Enumeration. *ACM Transactions on Modeling and Computer Simulation*, 23(3):17:1–24.

Wang, W., Wan, H., and Chang, K. H. (2016). Randomized Block Coordinate Descendant STRONG for Large-Scale Stochastic Optimisation. In T. M. K. Roeder et al., editor, *Proceedings of the 2016 Winter Simulation Conference*, pages 614–625, Piscataway, New Jersey. IEEE.

Welch, B. L. (1938). The Significance of the Difference Between Two Means when the Population Variances are Unequal. *Biometrika*, 29(3):350–362.

Wu, Z., Li, B., Dang, C., Hu, F., Zhu, Q., and Fu, B. (2017). Solving Long Haul Airline Disruption Problem Caused by Groundings using a Distributed Fixed-point Computational Approach to Integer Programming. *Neurocomputing*, 269:232–255.

Xu, J. (2012). Efficient Discrete Optimization via Simulation using Stochastic Kriging. In C. Laroque et al., editor, *Proceedings of the 2012 Winter Simulation Conference*, pages 1–12, Piscataway, New Jersey. IEEE.

Xu, J., Nelson, B. L., and Hong, L. J. (2013). An Adaptive Hyperbox Algorithm for High-Dimensional Discrete Optimization via Simulation Problems. *INFORMS Journal on Computing*, 25(1):133–146.

Xu, J., Zhang, S., Chen, C.-H., Huang, E., Lee, L. H., and Celik, N. (2014). Efficient Multi-fidelity Simulation Optimisation. In A. Tolk et al., editor, *Proceedings of the 2014 Winter Simulation Conference*, pages 3940–3951, Piscataway, New Jersey. IEEE.

Xu, J., Zhang, S., Huang, E., Chen, C.-h., Lee, L. H., and 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.

Yan, S., Tang, C. H., and Shieh, C. L. (2005). A Simulation Framework for Evaluating Airline Temporary Schedule Adjustments Following Incidents. *Transportation Planning and Technology*, 28(3):189–211.

Zhang, D., Henry Lau, H. Y. K., and 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., and Gao, Q. (2015). A Stochastic Programming Approach on Aircraft Recovery Problem. *Mathematical Problems in Engineering*, 2015:1–9.