RAILWAY CREW RESCHEDULING FOR DISRUPTION MANAGEMENT

by

JIE YUAN

A thesis submitted to the University of Birmingham for the degree of

DOCTOR OF PHILOSOPHY

Birmingham Centre for Railway Research and Education

School of Engineering

College of Engineering and Physical Sciences

University of Birmingham

February 2023

ABSTRACT

Unforeseen events from external influences such as major weather events, and internal causes such as infrastructure failures disrupt daily dense train operations. Such disruptions can quickly spread over the network and cause planned crew schedules to become infeasible to follow. Being one of the important steps in recovery of the railway service following a disruption, if crew rescheduling is not properly considered, it can jeopardise the return to stable service. This thesis mainly focuses on railway crew rescheduling for disruption management.

This thesis studies real-time railway crew rescheduling in theory and practice. By carefully examining the current literature on railway crew rescheduling, this thesis presents a detailed analysis and comparison of the current methods. This thesis provides models and methods for the crew rescheduling problems caused by two distinct types of disruptions: minor disruptions and significant disruptions. Sensitivity tests are conducted on several parameters to explore the impact on solutions. Meanwhile, this thesis considers that optimisation tools for solving the railway crew rescheduling problem cannot be a standalone optimisation tool for controllers to use. If no solution and no further information is given by an optimisation tool, time will be wasted. If no feedback is given by an optimisation tool, there will be no resolution. When such situations occur, controllers usually do not know what has happened inside an optimisation tool and how to get potential solutions. A feedback mechanism is proposed to output the reasons for not producing solutions and to adjust parameter values used in the crew rescheduling problems to give a good chance of generating results with revised values.

A timetable rescheduling model is proposed to model the impact on train services of a disruption and predict the recovery period. The recovery period measures how quickly a timetable can return to its normal level. A disruption neighbourhood is introduced as an idea, which is used to identify the drivers that should be considered in the crew rescheduling model for significant disruptions. It is characterised by the drivers who are included in the model and the recovery period. Algorithms are proposed to find the drivers that should be considered in a disruption neighbourhood to obtain good solutions. Several mathematical techniques and methods are proposed to speed up the solution time for the crew rescheduling problem for significant disruptions.

Further, the integrated rolling stock and crew rescheduling problem is still an immature research area. This thesis presents detailed formulations to model the problem and explores this problem with retiming possibilities. It can provide mutually feasible rescheduling solutions between rolling stock rescheduling and crew rescheduling. Several goals that relate to rolling stock and crew during disruption management are considered, analysed and further grouped into different objectives. Two kinds of multicriteria decision making (MCDM) methods are used to produce a set of optimal solutions for the integrated problem.

PAPERS AND CONFERENCES

During the study of my PhD, the following articles have been published. The co-authors in the articles listed below have advised on conceptualization, review and supervision.

Publications

1. Yuan J, Jones D, Nicholson G. Flexible Real-time Railway Crew Rescheduling using Depth-first Search. Journal of Rail Transport Planning & Management, Volume 24, 1-17, 2022. Doi: 10.1016/j.jrtpm.2022.100353
2. Yuan J, Jones D, Nicholson G. Real-time Crew Rescheduling with Pre-learned Crew Scheduling Constraints and Disruption Impact, Proceedings of World Congress on Railway Research 2022 Conference (WCRR2022), Birmingham, 6-10 June 2022
3. Yuan J, Jones D, Nicholson G. Flexible Real-time Railway Crew Rescheduling during Disruption, Proceedings of the 9th International Conference on Railway Operations Modelling and Analysis (RailBeijing 2021), Beijing, 3-7 November 2021

ACKNOWLEDGEMENTS

I would like to thank my supervisors: Clive Roberts, Gemma Nicholson and Daniel Jones, for their consistent support and guidance during the PhD projects. They always encouraged me to explore the research directions that interested me and gave detailed and thoughtful comments on my writings including this thesis. I am especially grateful to Gemma for the weekly meetings where we also discussed study, career and various top trending topics in society. I would also like to express my gratitude to the School of Engineering, the University of Birmingham which generously supported me for the PhD project. Furthermore, I would like to thank the Rail Operations team at Worldline UK&I, whose expertise and advice were invaluable in guiding the practical application aspects of this work. In the end, I would like to thank my family for always supporting me and cheering me up during difficult times.

CONTENTS LIST

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER ONE: INTRODUCTION

Trains in today's railways are running at high capacity on the network to meet demand. Punctual and stable train services are sought by both train operators and passengers. However, unforeseen events from external influences such as major weather events, and internal causes such as infrastructure failures and rolling stock breakdown challenge dense train operations. Such disruptions can quickly spread over the network due to the strong interdependencies in a railway network. Depending on the scale of a disruption, timetable, rolling stock and crew may need to be recovered from a disrupted status as quickly as possible. Rescheduling timetable, rolling stock and crew usually are carried out sequentially and iteratively. A mutually compatible solution among all three steps needs to be found. Theoretically, the adjustments among the three steps can go back and forth for several rounds until a mutually compatible solution appears. Being one of the important steps in recovery of the railway service following a disruption, crew rescheduling is often neglected. However, if it is not properly considered when the timetable is revised, it can jeopardise the return to stable service.

This thesis mainly focuses on railway crew rescheduling for disruption management. Furthermore, integrated crew rescheduling with rolling stock rescheduling is also explored. Various kinds of crew work on trains, including drivers, train conductors, train managers and catering staff. In the context of this thesis, the terms drivers and crew are interchangeable since the model and approach to reschedule drivers can be easily modified and used for other kinds of train crew. This thesis limits the discussion to train drivers. Timetable, rolling stock and crew recovery are three closely linked steps in a disruption management process. A recovery solution from any step should comply with the other two. The terminologies used in this thesis are used in Great Britain and case studies are from Great Britain. Two types of trains are running on the same railways in Great Britain: freight trains and passenger trains. This thesis mainly focuses on the operations of passenger trains.

In the remainder of this chapter, Section 1.1 is the background. The motivation for this research is given in Section 1.2 and the research questions addressed are proposed in Section 1.3 and Section 1.4 is an overview of this thesis.

1.1. Background

The basis of passenger railway operations is the timetable which describes a set of train services. Each service is scheduled to run from one terminal station to another terminal station and call at a number of intermediate stops at specific times or pass other significant locations at specific times. To provide the train services, resources such as crew and rolling stock are needed. Detailed schedules are set for crew and rolling stock to perform daily train services. However, due to the inevitable occurrence of unplanned events, train services often cannot be provided on time as set out in a timetable. This further leads to the schedules set for crew and rolling stock become infeasible to follow.

Following the definition of disruption in the airline industry (Clausen, Larsen, Larsen, & Rezanova, 2010), in this thesis, disruption is defined as an event or a series of events that renders timetables and the planned schedules for rolling stock and crew infeasible.

Initiating event(s) cause a disrupted situation, which may be referred to as a disruption. When a disruption happens, a corresponding rescheduling process may need to be initiated depending on the scale of an incident.

A crew member's schedule is called a crew diagram, which is a list of activities starting with signing on and ending with signing off at a crew depot. The typical activities of a driver's diagram include driving tasks, breaks, taxi trips or boarding another train as a passenger to move from one place to another. After a disruption, the planned diagram for a driver may become impossible to follow due to cancelled or delayed trains. In this case, the diagram becomes infeasible. This infeasible diagram needs to be revised to a feasible diagram, called a recovery diagram, to be assigned to the driver. A special class of driver is spare drivers who are reserved at major stations. Spare drivers have an empty diagram which only contains signing on and signing off at a depot. They are reserved for covering tasks for other drivers when there is a disruption. Thus, the availability of spare drivers is important in disruption management. The number of spare drivers who are on duty at a major station on a day is limited due to cost.

For a relatively large incident, railway disruption management mainly contains three sub-problems: timetable, rolling stock and crew recovery. Usually, these three subproblems are solved in a sequential manner. The recovery period is the time length for recovery of the railway services after a disruption. The length of the recovery period is predicted and is important in solving the three subproblems after a disruption. After the recovery period, rolling stock and crew should be recovered back to their planned diagrams and perform tasks in the normal timetable. Using optimisation tools to solve railway rescheduling problems has been a rising topic in the operations research community. A large body of literature of applying operations research methods to solve timetable rescheduling problems exists. However, the latter two problems, rolling stock rescheduling and crew rescheduling lack the same level of attention.

For countries like Great Britain, the railway infrastructure provider is separated from railway services operators. They both have control rooms where a group of controllers are located. Controllers are responsible for managing disruption and recover train services as soon as possible after a disruption. Different railway service operators are assumed to have similar control arrangements and roles in a control room. Usually, a control manager is in charge of the control room. Train running managers cover the operation of the operator’s train services and are responsible for communicating with the infrastructure provider about the incident. Rolling stock controllers are responsible for rescheduling rolling stock. Crew rescheduling controllers are responsible for rescheduling crew and communicating with the crew about their changed diagrams. Sometimes the railway network of a train service operator is further split into parts and each crew controller is in charge of the crew in a part of the network. The detailed disruption management process is explained in Section 2.3.

1.2. Motivation for the Research

Although being a crucial step in railway disruption management, railway crew rescheduling in Great Britain is currently conducted manually. Crew controllers are under huge time pressure to reschedule crew after a disruption happens. The crew rescheduling problem is hard to solve for the following reasons:

(1) The size of the crew rescheduling problem. The size of the overall problem is directly decided by the number of drivers and driving tasks that need to be considered in the problem. It is straightforward that the directly affected drivers whose diagrams become infeasible should be included in the problem. However, other drivers should also be included so their feasible diagrams can be divided up and used in the new set of recovery diagrams. These driving tasks will be mixed with driving tasks from infeasible diagrams to be used to generate new feasible diagrams for all drivers considered in a problem. The number of driving tasks considered in a problem is also affected by the rescheduling period, during which the crew are rescheduled and after which the crew are recovered back to work on their planned diagrams. Since driving tasks are collected by the rule that they should commence within the rescheduling period, a longer recovery will lead to more driving tasks to be considered in a crew rescheduling problem. A crew rescheduling problem involving a large number of drivers and driving tasks may take a long time to solve.

(2) Fatigue rules need to be considered. Due to working safety requirements ((Rail Safety and Standard Board, 2012), (Office of Rail and Road, 2013)), drivers'

planned diagrams are carefully designed following labour rules, which within one shift are mainly shown by two aspects: (a) after a certain period of continuous driving, a driver needs to have a meal break; (b) the number of hours worked during a day is limited. There are also rules for crew rosters which may span consecutive days. For example, there is a minimum rest period of 12 hours between signing off from one shift to signing onto the next shift. During the rescheduling process, drivers may be rescheduled to work on different trains and fatigue rules need to be observed at all times. Guaranteeing that each driver can still have enough rest according to the regulations during the rescheduling process is difficult to achieve manually in pressured conditions.

(3) Retiming. As previously explained, crew rescheduling is usually the last important step in railway disruption management. It takes the rescheduling results of timetable recovery and rolling stock recovery as input. However, it may be necessary to cycle back and alter a former decision. For example, crew rescheduling may have a solution or better solution if some tasks are retimed for drivers to cover them in the rescheduled timetable. Retiming a task may lead to more tasks needing to be retimed if they are carried out subsequently by the same rolling stock. It is also necessary to consider if the rolling stock still has enough turnaround time at a terminal station if the last task is retimed. It is very challenging to consider retiming tasks and their potential impact on rescheduling.

(4) Robust solutions. A solution can fail in implementation due to not considering the real rescheduling environment or if a driver cannot be contacted in time. If a

solution fails in implementation, the solution is not robust enough. To have a robust solution, some constraints need to be considered. For example, the connection time needs to be considered so that drivers have enough time to change between two different activities, for example, two driving tasks on different trains. A driver not having enough time to change from one train to another could lead to their next train being delayed and further disruptions. This shows the solution is not robust enough and has implementation risk. Communication time needs to be considered for drivers to be contacted to receive their changed diagrams. Also, the solution needs to be robust since there is uncertainty about the accuracy of the available information when the rescheduling is made. When a disruption happens, the length of disruption is uncertain. Controllers only have information about the current disruption status (which may be fully up to date) and must estimate the time at which the disruption cause will be resolved. However, it may actually take a longer time than predicted for a disruption to be resolved and new information about the disruption may become known to controllers at any time. Thus, a strict solution may not be able to be implemented when new information about the disruption is known.

(5) Evaluating solutions. There may be several different solutions to reschedule crew after their diagrams become infeasible due to disruption. Each solution has implications for later operations. It is non-trivial, or even impossible, for a crew controller to evaluate each solution and find the most suitable one quickly. In the crew rescheduling process, consideration needs to be given to finding a

relatively low-cost solution in terms of cancellation and train delay penalties, overtime compensation, taxi fares, maintaining a low impact on passenger satisfaction, reducing further rescheduling, etc. One of the main challenges in rescheduling is to quickly propose a solution that achieves a good balance among these considerations.

To carefully consider each item discussed above and calculate the quality of a solution in a limited time, controllers are facing more and more challenges in today's railways with networks running at almost full capacity. A more reliable and quick decision-making system which can assist controllers to reliably find feasible and good enough solutions considering the balance between many factors and satisfying various constraints is needed. This is the first main motivation for this PhD research.

Second, a variety of parameters are used in solving the crew rescheduling problem. The initial values of the parameters are set by controllers. When parameters are set in a way that there is no feasible solution for rescheduling crew, controllers may be left uncertain about why, reducing trust in using a crew rescheduling decision support tool. Thus, it is necessary to understand how essential parameters affect crew rescheduling solutions and a feedback mechanism should be added to provide reasons for not being able to provide a solution. Controllers can adjust parameters according to feedback information and obtain potential solutions.

Third, generating more than one solution to a disruption may be desirable. The reasons for multiple solutions are threefold. (1) Many criteria can be used to evaluate a solution. It rarely happens that one solution is the best among all criteria; (2) A model for the crew rescheduling problem may not be able to include all the constraints that exist at the instant when a rescheduling process is initiated. There are various reasons that a model may not include all the constraints. First, a model could have very complex constraint structures after considering all constraints and make it impossible to solve in real-time. Second, some constraints cannot be foreseen. For example, a driver refuses to be rescheduled because they want to sign off on time at their depot for personal reasons. Thus, in reality controllers who have the most accurate information about the disruption should be able to use their knowledge and experience to choose the most appropriate solution. (3) One solution may fail in implementation due to lack of robustness or be rejected by drivers, thus other feasible solutions should be preserved as backup plans. Therefore, a model which can provide multiple optimal solutions should be studied.

Fourth, the common rescheduling strategies used by controllers in reality do not just involve crew. Sometimes, these rescheduling strategies involve both rolling stock and crew together. It needs an integrated rolling stock and crew model to give such solutions. When rescheduling rolling stock and crew in sequence, if the crew rescheduling solution cannot cover all the tasks that rolling stock covers, then a different rolling stock scheduling solution is needed. The adjustments between rolling stock and crew can go back and forth until a mutually feasible solution appears. Also, solving an integrated problem rather than two sub-problems can give no worse solutions and the solution is naturally feasible between the two sub problems. There is very limited literature in this area and the current research does not address the integrated rolling stock and crew problem fully. Thus, exploring a suitable model for the integrated rolling stock and crew rescheduling problem is one of the purposes of the thesis.

1.3. Research Questions

To fulfil the purposes of the thesis, the following questions about real-time railway crew rescheduling are posed:

1. *How can crew rescheduling problems be categorised and how does crew rescheduling fit into the railway disruption management process?*
2. *How can the crew rescheduling problem be modelled in each case using the categories determined in answer to question 1? How can various real-world constraints be set in the models, for example, fatigue rules, connection time and communication time? Which factors should be considered in evaluating the quality of crew rescheduling solutions and how can each factor be quantified?*
3. *How do parameters in the crew rescheduling problem affect the solutions and how can a feedback mechanism to adjust parameter values when there is no solution be set up?*
4. *How can the impact on train services of a disruption be modelled and the recovery period be predicted? The impact on different drivers from a particular*

disruption varies. Some drivers are directly affected by the disruption and others are not. It is not necessarily enough to find a good solution by just rescheduling directly affected drivers. Which drivers' schedules should be considered for rescheduling following a disruption?

5. *The crew rescheduling problem can take a long time to solve due to its problem size. Which techniques can be used to speed up the process of solving the crew rescheduling problem?*
6. *How can the integrated rolling stock and crew rescheduling problem with retiming possibilities be modelled and solved? How can multiple solutions be obtained in solving the integrated rolling stock and crew problem?*

1.4.Overview

The remainder of the thesis is organised as follows. Chapter 2 describes the planning and rescheduling process of passenger railway operations and the organisations involved in the two processes. Disruptions are categorised into three levels in this section. Two levels of disruptions are addressed in the thesis: minor disruptions and significant disruptions. Literature on railway crew planning and rescheduling is also reviewed in this chapter. Chapter 3 addresses the model and method, depth-first search crew recovery (DFSCR), for solving crew rescheduling during minor disruptions. Chapter 4 is about crew rescheduling during significant disruptions. The problem is formulated using integer linear programming and an efficient heuristic algorithm is proposed. Chapter 5 studies the integrated rolling stock and crew rescheduling problem with retiming possibilities. The model formulated for the problem has been successfully applied to several single delay and multiple delay scenarios. Chapter 6 is the conclusion.

CHAPTER TWO: PASSENGER RAILWAY OPERATIONS

The operations of the passenger railway are through a complicated system involving several parties with different responsibilities. This chapter lays out the background to passenger railway operations. The two most important railway parties: Infrastructure Manager (IM) and Train Operating Companies (TOCs) are first discussed in Section 2.1. The discussion is based on the situation in Great Britain. The main railway planning processes: timetable, rolling stock and crew planning are described against the four planning levels: strategic level, tactical level, operational level, and short-term level in Section 2.2. Section 2.3 is about railway disruption management. Commonly used rescheduling strategies in timetable, rolling stock and crew rescheduling are discussed. Section 2.4 and 2.5 review the existing publications in railway crew scheduling and rescheduling problems, respectively. This chapter answers research question 1 posed in Section 1.3. The main railway terminology used in this thesis is given first ((American Public Transportation Association, 2019) and (Nielsen L. K., 2011)).

Railway Terminology

Rolling stock: Rolling stock refers to railway vehicles, including both powered and unpowered vehicles. The typical rolling stock includes locomotives, powered and unpowered cars, wagons, multiple units, etc. A locomotive is a rail transport vehicle that provides the motive power.

Rolling stock unit: A unit is a self-propelled railway vehicle consisting of a fixed number of carriages. A multiple unit (MU) is a self-propelled entity composed of one or more carriages joined together, which can be coupled to another MU to form a composition to allow one driver to control.

Rolling stock composition: Rolling stock units can be combined with each other to form compositions. A rolling stock composition describes the number of each rolling stock unit, and in which order they appear in a train. Rolling stock compositions are frequently adapted during daily operations by uncoupling units from or coupling units to trains.

Train: A train is a railway vehicle or series of connected railway vehicles that is used or intended to be used in a train service.

Train service: A train service is a regularly scheduled transit service from a specified origin to a specified destination usually with several intermediate scheduled stops. A train service is usually associated with an identifying code. In GB, this code is a four-character code called the train reporting number or headcode.

Timetable: A timetable specifies the timings and stopping patterns of trains and other empty railway vehicles.

Train path: A train path is the infrastructure reserved to run a train between two places over a given period.

Relief station: Due to the facilities in a railway station some stations are relief stations where drivers can be scheduled to change trains.

Trips: For the planning process, the train services of a timetable are divided into smaller components called trips. The division into trips is performed in such a way that the assignment of a driver to a train cannot be changed during a trip. A service may contain one or more trips.

Diagram: A crew member’s schedule is called a crew diagram, which is a list of activities starting with sign on and ending with sign off at a crew depot. The typical activities of a driver’s diagram include sign on and off, driving tasks, breaks, taxi trips or boarding another train as a passenger to move from one place to another. There are also some activities arising from rolling stock diagramming including disposal, attachment, mobilisation, etc. A diagram of a rolling stock unit is a series of tasks performed by the unit on a day.

Driving task: A driving task is a special class of activity, which is specified between two major stations and has attributes: headcode (the GB term to identify a train service), and route and rolling stock knowledge requirements, indicating the required competencies of drivers for the specific route and rolling stock. A driving task usually consists of one trip, but sometimes it consists of more trips from one train service.

Recovery period: In this work, a recovery period is the length of time from rescheduling begins following disruption to the time when a normal timetable can be restored and rescheduling ends.

Disruption period: A disruption period is the time length when full infrastructure access is limited. The length of disruption period has uncertainty. If the disruption is caused by a technical fault, it can be evaluated as the expected time required to fix it. If the disruption is caused by unpredictable reason, say bad weather or missing crew, the disruption duration may need to be estimated by controllers.

2.1.Great Britain Railway Companies and Their Responsibilities

The smooth running of Britain’s railway relies on several parties’ cooperation. Typical industry organisations including Network Rail, Train operating companies (TOCs), Freight Operating company (FOCs) and rolling stock leasing companies (ROSCOs). TOCs run passenger rail services, leasing and managing stations (over 2,500) from Network Rail. Freight operating companies (FOCs) use the railway network to run trains that transport goods. ROSCOs own most of the coaches, locomotives and freight wagons which they lease to TOCs and FOCs, which together are called Railway Undertakings (RUs). In this Section 2.1, the railway infrastructure manager and railway operators in Great Britain are introduced.

2.1.1. Infrastructure Management by Network Rail

Like other European countries, Great Britain separates the infrastructure manager and train/freight operating companies. The government funds Network Rail (NR) to provide railway infrastructure which must satisfy a set of requirements like capacity and reliability, etc.

Network Rail (NR) is the owner, operator and infrastructure manager of Britain's main railway network. Great British Railways (GBR) shall replace NR in this respect from 2024. GBR also has other responsibilities, for example, contracting of passenger train services, the setting of fares and the collection of fare revenue, etc., see (Department for Transport, 2021). NR has the following four duties:

NR owns, repairs and develops the railway infrastructure in England, Scotland and Wales, including 20,000 miles of track, 30,000 bridges, tunnels and viaducts and the thousands of signals, level crossings and stations (biggest and busiest stations: 11 in London and 20 outside London). NR has designated the geographical area covered by railway network into five regions. Each region is further divided into routes. NR establishes and maintains a route business for each route area which takes primary responsibility for the part of the network and stations within the route area.

NR is responsible for understanding and shaping the future of Britain's railway through delivery of long-terms plans. A strategic network planning team is established and maintained to investigate the long-term railway contributions to national and regional economic growth and social well-being. The team is also responsible for understanding the railway capabilities and how it delivers railway passenger and freight services. It investigates the likely changes to passenger demands, patterns of train services and to the railway in future. Strategic advice is provided to railway funders to understand the complexities of the railway and to make informed investment decisions that could require potential changes in timetable, rolling stock and crew.

NR is responsible for setting the timetable. Various types of trains run on the railway network: metro, regional, intercity and high-speed trains. They differ from each other in journey time, maximum speed, etc. The timetable planning needs to balance the demands of stopping, non-stopping passenger and freight train services as well as taking lots of factors into account to keep the railway safe. In countries that separate railway infrastructure management from railway operations, the timetable is a collaborative effort between infrastructure provider and train/freight operating companies.

NR is responsible for directing service recovery. Control centres are set based on the principal routes to operate trains within the routes. In each control centre, there are a number of distinct roles which are responsible for collecting incident information, dealing with the incident itself, recovery timetable as quickly as possible and providing customer with consistent delay information. The detailed description can be seen in (BCRRE, NetworkRail, RFI, & TV, 2014). According to (RSSB, 2020), a NR manager oversees the disruption management for a specific NR route. NR controllers mainly interfaces with signallers to manage network capacity utilisation and service alterations.

2.1.2. Railway Operators in Great Britain

For train services, the government invites TOCs to tender. The TOCs bid for franchises based on minimum service requirements set out by the government. TOCs also liaise with NR to confirm that sufficient capacity is available to support the planned service. The passenger rail franchising system was created as part of the privatisation of British Rail in 1994. Railway franchises are awarded by the UK government's Department for Transport (DfT) to train operating companies through a process of competitive tendering. Franchises usually last from 7 to 10 years and cover a defined geographic area or service type. Over the years, the number of franchises in Great Britain reduced through a series of mergers. The current passenger rail franchising system will be replaced by a concession-based system in the government's latest plan to transform the railways by introducing GBR to replace NR in Great Britain (Department for Transport, 2021).

TOCs have existed since the privatisation of the network under the Railways Act 1993. Most TOCs hold franchises let by the DfT while a small number of open-access operators hold licenses to provide supplementary services on some routes. Some operators have been taken over by a government-owned operator of last resort, which operates a railway franchise on behalf of the government when a TOC is no longer able to do so. TOCs in Great Britain have been changing through years. Some TOCs have ceased to exist for reasons like withdrawal of the franchise, expired franchise, bankruptcy and merger. Many of the TOCs are in fact part of larger companies which operate multiple franchises.

TOCs pay NR to access tracks to run services and lease trains from ROSCOs. The proportion by which each cost element makes up an operator’s total cost is shown in Figure 1. There are 19 TOCs included in Figure 1: c2c, Chiltern Railways, CrossCountry, East Coast, East Midlands Trains, Abellio Greater Anglia, First Great Western, Northern, Southeastern, Southern, South West Trains, First Capital Connect, First TransPennine Express, Arriva Trains Wales, Virgin Trains West Coast, London Midland, London Overground, Merseyrail and First ScotRail. Staff costs accounts most in the total cost, followed by network rail charges or rolling stock charges, and energy fees occupy the least total cost.

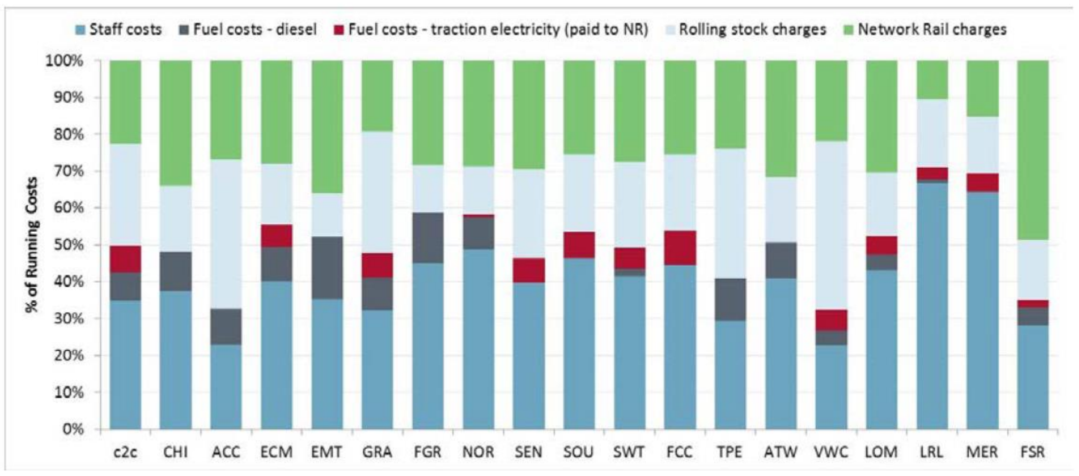


Figure 1 Total running cost per TOC (Office of Rail and Road, 2015)

2.2. Railway Planning

Railway planning is the scheduling of train movements and the allocation of resources to the timetable. Following (Huisman, Kroon, Lentink, & Vromans, 2005), different planning problems of a passenger railway operator can be classified under four hierarchies based on time horizon: strategic level, tactical level, operational level and short-term level. In this section, the three main planning problems are discussed. They relate to the planning of the timetable, the rolling stock and the crew. Strategic planning can start decades before the train operation. There are three main planning problems at the strategic level: line planning, rolling stock management and crew planning.

2.2.1. Timetable Planning

Timetable is the foundation of railway operations. The four levels of timetable planning are described below.

Line Planning at Strategic Level

A line is a direct railway connection between two terminal stations that is operated with a certain frequency and with a certain train type (Huisman, Kroon, Lentink, & Vromans, 2005). At the strategic level, lines are decided based on the estimated passenger and freight demand. The type and frequency of trains on each line are decided. Three train types may be categorised depending on stopping patterns. Intercity trains stop only at large stations. Interregional trains stop at a number of medium-sized stations and regional trains stop at nearly all stations they pass.

Timetable Planning at Tactical Level

At the tactical level, the timetable is designed using the train lines designed at the strategic level. Many European countries operate a cyclic timetable, a timetable in which the trains can be grouped into series such that trains in each series have the same routes and stop stations and the difference between the departure time of two successive trains is one cycle time (usually one hour), see (Cacchiani, 2008). Most cyclic timetable models are based on the Periodic Event Scheduling Problem, initially put forward by (Serafini & Ukovich, 1989). A general form is used to formulate running time, dwell time, passenger connection time and headway time constraints etc., between two events (arrival events and departure events for trains).

In GB, Long Term Planning (LTP) is at tactical level; this phase starts 16 months before the timetable’s first operational day. Each train and freight operating company develops the timetable that they would like to run in their area 14 months before the timetable operational day and Network Rail is responsible for coordinating and validating their timetables and developing a national timetable (NetworkRail, 2022). 6 months in advance of the timetable operational day, a national base timetable is provided to operators to enable them to start planning rolling stock and crew.

Timetable Planning at Operational Level

In GB, the term used to describe timetable planning at operational level is Short Term Planning (STP). 4 months in advance of operational day, operators can apply for readjustments to their timetable taking into account things like special events or weekend engineering work and NR need to work through these modifications to ensure there is no conflict and that trains can run smoothly. At the operational level, the tactical timetable is updated by adding new trains, modifying departure or arrival time. Eventually, 3 months in advance of timetable operations, a timetable for each week is finalised and the railway industry publishes the timetable for passengers. Also, a detailed timetable platform assignment plan is finalised (Narayanaswami & Rangaraj, 2012).

Timetable Planning at Short-term Level

In Great Britain, part of Very Short-Term Planning (VSTP) corresponds to timetable planning at the short-term level. VSTP covers the need to change timetables within 48 hours of operation and on the day of operation. In the before operation case, VSTP includes applying previously unplanned train paths in particular for freight trains and empty coaching stock. On the operational day, VSTP requires developing a basic plan for rapid implementation in response to a disruption ahead, for example, speed limitations in extreme weather. In the latter case, on operational day, the need to change timetable is explained in Section 2.3, Railway Disruption Management.

2.2.2. Rolling Stock Planning

Rolling stock is one of the resources that needs to be allocated to operate a timetable smoothly. Rolling stock planning at the four-time levels is discussed below.

Rolling Stock Planning at Strategic Level

Rolling stock management focuses on procuring and allocating the required rolling stock capacity. Relative aspects need to be considered when making a main decision on rolling stock capacity include the total demand for passenger railway services, the frequency of trains at peak and off-peak times, first-class and second-class train services. Decisions that need to be made include selection of rolling stock unit types, ordering new rolling stock units and rolling stock maintenance strategies.

Rolling Stock Management at Tactical Level

For rolling stock management, rolling stock allocations to the trains at peak time for a standard day of a week are decided at the tactical level. The reason behind this is that if it is possible to allocate rolling stock to trains during peak time, then the allocation will be appropriate during the other times of the day (see (Huisman, Kroon, Lentink, & Vromans, 2005)). To determine rolling stock allocations to the trains at peak time, one first needs to decide which types of rolling stock and how many units of each type need to be allocated to each line.

The goal of the rolling stock management at the tactical level is to match and provide the required rolling stock capacity for trains at peak time for each line. A rolling stock circulation for a standard day in a week is determined first using the rolling stock allocations to the trains at peak time. Then, modifications are made so that the rolling stock units ending at a station the previous evening can provide enough rolling stock units for train operations next morning. That is, a general week of rolling stock circulation has been determined. Conflicts may arise among rolling stock belonging to different lines at a station due to limited infrastructure capacity. These conflicts are resolved by local planners in shunting plans at each station.

Rolling Stock Management at Operational Level and Short-term Level

At operational level, modified timetables due to national events could destroy the feasibility of the planned rolling stock circulation. Thus, alterations to rolling stock plans are made afterward timetable modification. This is done in such a way that the perturbations to shunting plans are as small as possible.

Similar to the operational level, alternations to rolling stock schedules are made following the changes in timetable at the short-term level. Also, maintenance routing of rolling stock is planned at the short-term planning level.

2.2.3. Crew Planning

Similar to rolling stock, crew is another essential resource for running of a railway timetable.

Crew Planning at Strategic Level

Crew planning at strategic planning level deals with the location and capacity of depots and available number of drivers and conductors at each depot. Depots may need to open or close depending on the major changes introduced in the line planning. The required and available capacities of each crew depot are evaluated, and certain amounts of work can be shifted among depots to achieve a match between required and available capacities in the long term. To modify the capacity of a depot, major decisions like employing new crew, training new crew until they are fully operational and moving crew among depots are made.

Crew Planning at Tactical Level

The goal of crew planning at tactical level is to construct crew schedules. It is usually solved in two steps: crew scheduling and crew rostering. The crew scheduling problem is also called the diagram generation and selection problem. A diagram is a sequence of activities (e.g., driving task) which should start and end at the same depot. A driving task is defined as a trip between two relief stations assigned to a driver during which the driver cannot have a break or change trains. Crew scheduling generates anonymous daily working schedules (i.e., diagrams) which cover all required activities for the train services scheduled in the timetable for a defined period, e.g., a single workday. Station or depot-based activities (e.g., required rest breaks, signing off) are also included in crew diagrams. The problem is to select an optimal set of diagrams which can cover all the tasks from the unit diagrams, and which has the minimal cost. In the diagram generation step, feasible diagrams should be generated. In the diagram selection phase, diagrams should be chosen to minimise the operational cost, and to cover all the tasks from the timetable.

The crew rostering problem Is to use the diagrams generated from the crew scheduling step and then form them into a roster. A roster is sequences of diagrams with rest periods in between, usually overnight. (Sodhi & Norris, 2004) see creating a roster as two steps: creating a rest-day pattern and assigning specific diagrams to this pattern. A rest-day pattern consists of a sequence of weeks in which every day is assigned to a specific diagram type or rest day. The second step is to assign specific diagrams to this rest-day pattern. Usually, the roster is cyclically assigned to drivers. The length of the roster is the same as the number of train drivers in a depot. Crew rostering combines these diagrams into weekly or monthly sequences, which are subsequently assigned to individual crew members.

Crew Planning at Operational Level and Short-term Level

Crew schedules are adjusted for specific demands for particular weeks due to the changes of the timetable. Reasons for such demands can be national events, planned infrastructure maintenance work, and so on.

2.2.4. Summary

Management and co-ordination of railway operations is complicated since every single operational activity requires the compliance of several types of resources, infrastructure, vehicle and personnel, etc. The interdependencies among these resources are very complex. (Schiewe, 2020) presents a number of integrated railway planning models including integrating timetable and vehicle scheduling. The experimental results show that using the integrated timetable and vehicle scheduling model, the trade-offs between passenger travel time and operational costs can be found.

2.3. Railway Disruption Management

Railway disruption management faces some performance challenges in GB. In the 2020 Spring National Rail Passenger Survey (TransportFocus, 2020), 49% passengers state that their greatest cause of dissatisfaction is how train operators deal with delays.

Levels and Phases in Disruption Management

(RSSB, 2020) sets out the levels of disruption classification. *Minor disruption* sees delay and congestion, often without a specific incident occurring. *Significant disruption* sees a loss of or restricted access to parts of the railway network due to an incident, affecting the delivery of a normal timetable. *Severe disruption* occurs when incidents significantly restrict the use of the railway network and or last longer than one day (severe weather). The railway disruption management process is split into five phases: Phase 1 is identification, where the disruption is first identified and quantified, leading to the assessment of the level of the disruption and response. Phase 2a is selection, where the appropriate plans are selected and agreed. Phase 2b is deployment, where the plans are communicated and implemented. Phase 3 is operation, where the temporary service alterations are in place. Phase 4 is recovery, where the plans are removed, and service is gradually restored.

Types of Disruption Plans and Rescheduling Process

Different types of plans are prepared to handle different levels of disruptions. Train-by-train plans are created to standardise the rescheduling process concerning changes to individual trains or groups of trains for minor disruptions. Contingency plan templates for limited scenarios are made for significant disruptions. Very short-term planning, like short-term alterations are considered and prepared for typical severe disruption, like extreme weather.

Depending on the level of disruption, two processes are used. A *Train by Train Management Process* is used for minor disruptions with an associated set of train by train plans to be used as guidance. Train by train plans standardise the decision-making process regarding changes to individual trains or groups of trains. A *Service Recovery Framework Process* is used for significant disruptions and the immediate effect of severe disruptions with an associated set of contingency plans.

An optimised process for Train by Train Management is discussed and agreed by all key stakeholders in the IM and RUs. Train by train management is mainly conducted by the controllers from the RU who actually runs the train affected by the disruption. The controllers initially identify the disruption and assess its level. Then, the controllers select a train by train plan and agree it with the IM controllers. Then the plan is deployed. The disruption is monitored and if the disruption is determined no longer as minor in the future, a Service Recovery Framework Process is initiated. If delays are no longer occurring, the full service is on time.

Similarly, an optimised process for Service Recovery Framework for all RUs and IM operating in the designated area is discussed and agreed by all key stakeholders in the IM and RUs. An integrated part of the Service Recovery Framework process is the huddle, which describes a meeting of key stakeholders from all RUs and the IM for the area affected by the disruption. The result of a huddle is to trigger selection and deployment, operation and recovery in the Service Recovery Framework Process.

2.3.1. Service Recovery Framework Process

Service recovery is the process by which a normal timetable is restored, or a timetable for a degraded model is operated following disruption. The objective of train service recovery usually is to minimise overall disruption to passengers while returning to a normal timetable or an agreed degraded timetable (full timetable restored) as quickly as it is possible in practice, see (Mann & Panter, 2013) . This process is frequently substantially manual and implemented by controllers. In Europe and Japan, train service recovery is typically conducted by negotiations between two distinct entities: infrastructure manager and railway undertakings (see (Cacchiani, et al., 2014) and (Williams Rail Review, 2019)).

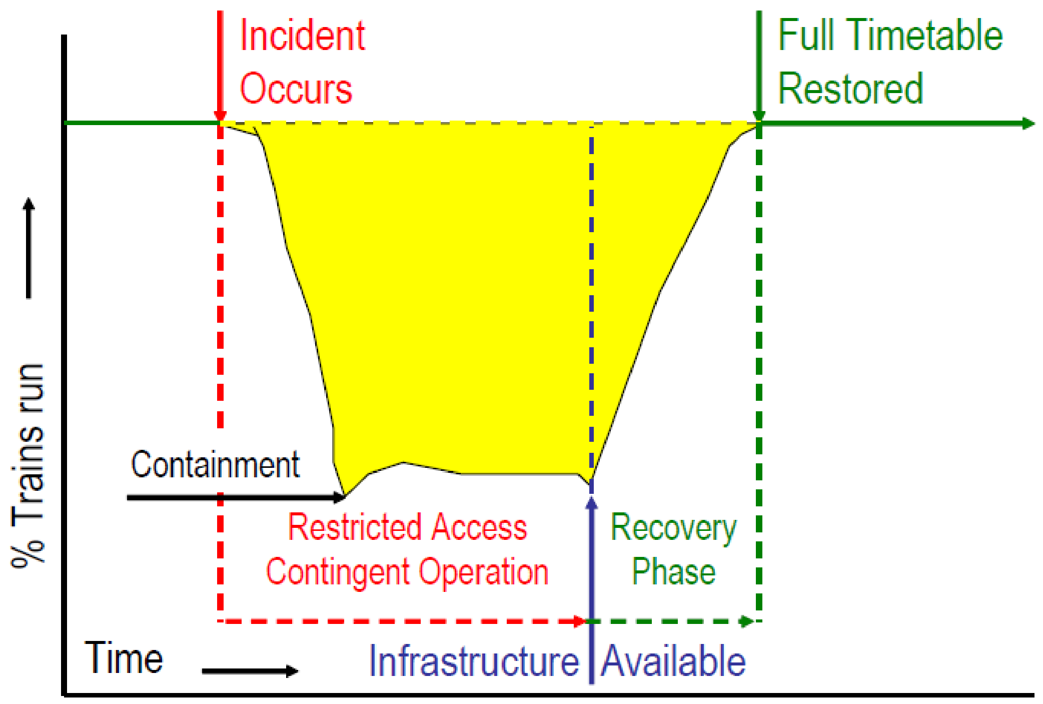


Figure 2 Train service recovery (Mann & Panter, 2013)

Figure 2 shows the number of operating trains and infrastructure status in a service recovery. Following a disruption, railway operation is at a restricted access state in which full infrastructure capacity cannot be provided to enable running of the normal timetable. A contingency plan, which is a pre-planned timetable that can still be used given the nature of the disruption and the restricted network, may be implemented. After the contingent operation, there will be a time during which the infrastructure for the normal timetable is available and the running of a timetable can be gradually restored.

Optimised Service Recovery Framework Process

In phase 1 (identification), a disruption is identified. The NR manager collects initial information from the disruption site and decides the level of disruption.

In phase 2 (selection and deployment), the NR manager holds a dialogue with affected parties. In this dialogue, they will consider the impact of the incident, discuss preliminary estimates about when normal infrastructure may be available, agree on contingency operation, arrange further conferences if necessary and declare a Service Recovery Commencement Time (SRCT). Component plans will be issued within parties and parties will commence timetable service recovery arrangements.

In phase 3 (operation), route controllers from RUs are responsible for rescheduling rolling stock and crew controllers are responsible for rescheduling crew. Then route controllers and crew controllers discuss with each other and propose a feasible solution which works both for rolling stock and crew. This solution will be provided to controllers in NR for agreement. If agreement is reached, controllers in NR and RUs will commence implementation of this solution.

In phase 4 (recovery), the full infrastructure access is available. The IM controller summarises current train performance and recommends any further changes to contingency plan operations. Further huddles are held if necessary. In the final step, a full timetable is restored, and dispensation provision applied by each RU and post-restoration review is instigated as appropriate by NR.

2.3.2. Timetable Rescheduling

The timetable is built with running time supplement and buffer time to increase the robustness. For minor disturbances, one can use this timetable slack by delaying trains, reducing running times, reducing headways, or reducing dwell time at the terminal station. For extensive re-scheduling, there are several methods to adjust the timetable during disruption management: a) Train overtaking: for example, breaking the predetermined order of lines and letting the fast train leave first, b) Inserting an on-time train in an intermediate station, c) Reroute: cancelling a train from departure, or skipping stations along the route or shortening the routes of the trains or cancelling the whole line.

How to revise the timetable to recover the services in a way that it minimises the disruption impact in both time and space horizons? A good solution should prevent disruption from propagating through the system and also recover train services quickly. There are some trade-off decisions to make, cancelling some services to make way for others, delaying some trains to let others through, etc. In practice, a revised timetable is designed to return to the planned timetable. The reason behind this is that the planned timetable is highly optimised and feasible.

2.3.3. Rolling Stock Rescheduling

This aim of rolling stock rescheduling is to provide capacity to the trains in a revised timetable. Due to disruption, some rolling stock units may end at the wrong positions. (Nielsen L. K., 2011) categorised the available options to reschedule rolling stock into three groups: changing shunting operations, adapting turning patterns and repositioning train units, which are described below.

Changing shunting operations. A planned shunting operation is made at each station at the connection between the rolling stock composition of an arrival trip and that of a departure trip. However, a different composition may be assigned to the arrival train that does not contain the right type of train unit that can be assigned to the departure trip, which may render the planned shunting operation of the composition invalid. A rescheduling strategy is to introduce a composition change at a connection where no composition change was planned. A newly introduced shunting operation may require different available tracks and yard capacity for the movements of the train units. The changes need to be communicated and agreed with local dispatchers.

Adapting turning patterns at terminal station. A planned turning pattern means that each incoming train is matched with an outgoing train. Applying a planned turning pattern can ease the task of local planning by repeating turning schemes. Locally changing the turning pattern by matching an incoming to a different outgoing train can require fewer shunting operations than strictly keeping the planned turning pattern in the rescheduling process. Changing a turning pattern may cause a train to depart from a different platform, which needs to be communicated with the controllers responsible for platform assignment.

Repositioning of rolling stock. Moving rolling stock units to another station can be an option if another station urgently needs them. During the day, trains run on the network at close to full capacity. It could be difficult to add an extra train. However, at night, the number of each rolling stock unit types may not match the required number for the next day’s operations due to disruptions during the day. Repositioning rolling stock is a frequent rescheduling strategy to tackle rolling stock off-balance at night.

There are other factors that controllers need to consider when they reschedule rolling stock. For example, the rolling stock which needs to undergo a maintenance check in a forthcoming couple of days should be monitored and controllers should make sure that these rolling stock units can get to the maintenance centre on time.

2.3.4. Crew Rescheduling

A disruption can render the planned crew diagrams infeasible by causing drivers on the affected trains to miss the start of one of their scheduled driving tasks. In this case, the planned diagram for the affected driver may become infeasible to follow. Thus, a recovery diagram needs to be assigned to the driver. When rescheduling crew during disruptions, it is preferable to reschedule as little crew as possible because implementation risks rise with the number of crew that need to be rescheduled. Also, after disruption, the crew should be recovered back to their planned diagrams since the planned diagram is already fully optimised and agreed by drivers. For small disruptions, it is preferable to solve it by swapping tasks between drivers rather than using the spare drivers who are reserved for big disruptions. If no driver can be found to cover a task, then this task needs to be cancelled, which is an undesirable solution since then the rolling stock circulation is disrupted and the timetable should also change.

Recovery Diagram

A recovery diagram is a revised diagram adjusted from a resource's planned diagram. All recovery diagrams assigned to all affected drivers together should ideally achieve an optimum depending on the objective set by a user, for example, to cover as many driving tasks as possible in an adjusted timetable with as few as possible changes to planned resource diagrams.

There are some rules about constructing a recovery diagram. A recovery diagram may not last more than a certain amount of time longer than the planned diagram. A recovery diagram should end at the same depot as in the planned diagram. A recovery diagram should consider breaks in compliance with the labour rules. A recovery diagram needs to have a certain amount of transfer time when a crew member transfers from one train to another. A recovery diagram should only contain driving tasks for which the crew member has corresponding route and traction knowledge.

In this work, focus is on minor disruptions and significant disruptions. Therefore, two methods of building recovery diagrams are proposed for the two kinds of disruptions, respectively.

2.3.5. Analysis of Crew Diagrams

To help readers understand crew diagrams better, in Section 2.3.5, some analysis of crew diagrams is presented. The three important characteristics of crew diagrams: the length of a diagram, continuous driving time and time required to change trains at a station are studied.

Figure 3 shows a day's work length of drivers of a TOC in the UK. There are 208 drivers on duty on a day, where depot A has the largest number of drivers, 60. The minimum, average and maximum work lengths are 5 hours, 7 hours 40 minutes and 10 hours, respectively.

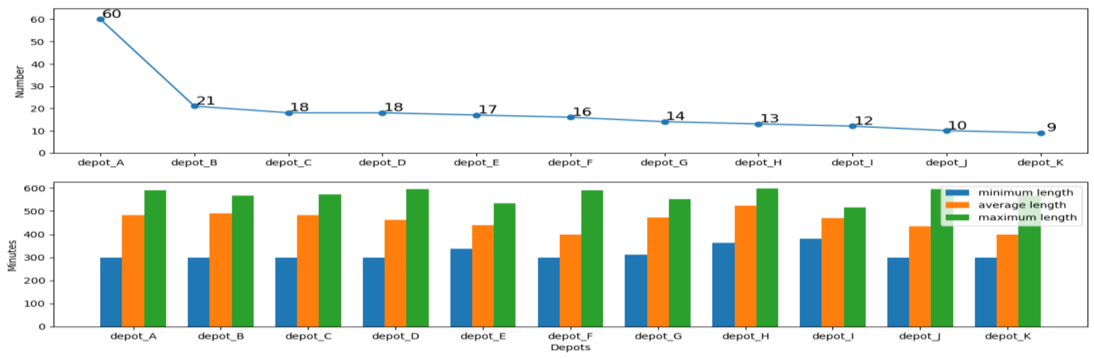


Figure 3 Number of diagrammed drivers and their work lengths in each depot

Among 208 drivers, 201 drivers have one meal break and 7 have two meal breaks in their diagrams. For drivers with one meal chance, Figure 4 shows the driving time before a meal break starts and after a meal break ends at different stations. The longest driving time for a meal break to appear is around 4 hours and 40 minutes and 4 hours and 10 minutes is the longest driving time after a meal to sign off.

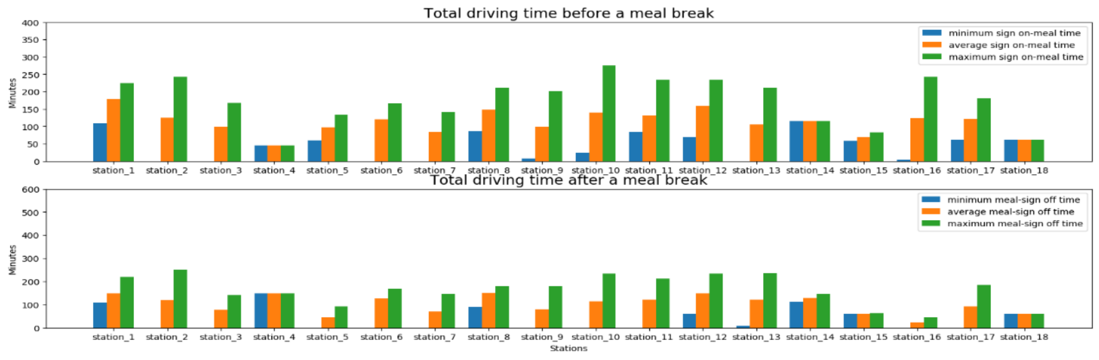


Figure 4 Total driving time before and after a meal break

It takes some time for drivers to change from one train to another at a relief station. Drivers need to completely stop the previous train and prepare the next train to depart. Change time depends on the scale of stations, platforms and trains, etc. We can get a general idea from studying the time difference for two successive driving tasks which belong to two trains on drivers' diagrams. Figure 5 shows various change times at different relief stations. The minimum change time, 7 minutes, happens at Station 2. The longest could be as long as 98 minutes, which is unlikely to reflect the real required change time. Since the statistics include idle time, the required times would be much smaller.

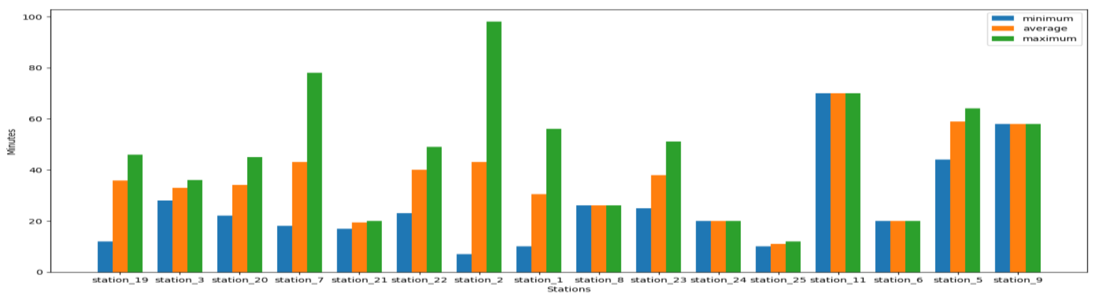


Figure 5 Change time at different stations

2.4. Literature Review on Crew Planning

The origin of the crew scheduling problem (CSP) in transportation can be found in airline industry works from the 1950s and 1960s. (Arabeyre, Fearnley, Steiger, & Teather, 1969) surveyed different approaches studied by some airlines in the 1960s to try to optimise the allocation of crew to flights. Using operations research methods, solving the CSP gained more attention in the 1980s and 1990s with the quickly advancing computer power. In the 1990s, the privatisation of railway operations in Europe required train operators to look for more productive and efficient crew scheduling solutions, which led to the rising interest of using computer power and mathematical algorithms in the railway industry for its potential of cost saving. In general, the CSP in the transportation industry: airline, bus, mass transit and railway etc., shares a wide range of similarities. In this work, we focus on research about the CSP in the railway industry.

Crew Scheduling Problem at Strategic and Tactical Level

Few works study crew scheduling at a strategic level. (Derigs, Malcherek, & Schäfer, 2010) present a system used in Germany that can analyse how working regulations will affect crew planning. (Sahin & Yuceoglu, 2011) consider a model to decide how many crew members are required to perform tasks in a given planning horizon under the working regulations.

Most of the existing research focuses on the tactical level, generating diagrams and forming diagrams into rosters optimally. (Caprara, Fischetti, Toth, Vigo, & Guida, 1997) define the crew management at tactical level as building the work schedules of crew needed to cover a planned timetable and its unit diagrams. The overall crew management problem is considered in two phases: crew scheduling and crew rostering. The main objective of crew management is to minimise the number of crew needed to perform all the daily occurrences of the trips in the timetable in the given period. Other objectives of the crew scheduling problem usually consider schedule efficiency (working time, idle time, break time, etc.), robustness (reducing train changes, adding buffer time, etc.) and employee satisfaction (safety, fairness and the popularity of the schedule).

Models for the Crew Scheduling Problem

Both crew scheduling and crew rostering problems require finding minimal cost sequences through given items. For crew scheduling, items like trips are sequenced to diagrams and for crew rostering, items like diagrams are sequenced to rosters. There are two basic integer linear program models to model both problems: generic set covering/partitioning problem (SCP/SPP) (e.g. (Caprara, Fischetti, Toth, Vigo, & Guida, 1997)) and generic minimal network flow problem (NFP) (e.g. (Vaidyanathan, Jha, & Ahuja, 2007)). Both formulation approaches in the literature use a space-time network to represent the problem. (Suyabatmaz & Şahin, 2015) used both formulations to solve a regional planning problem to minimise the number of crew members. They concluded that both formulations are capable of generating feasible solutions. Yet, the former formulation performs better not only in solution quality but also in computational time. SCP/SPP is a path-based approach which relies on identifying feasible paths (following labour and other scheduling rules) on a graph. NFP is an arc-based approach. Being a heavily restricted optimisation problem, the SCP/SPP formulation for the CSP usually has many fewer constraints compared to the NFP for the same CSP (Banihashemi & Haghani, 2001). The most commonly used formulation of the CSP is the set covering problem or its variations and the variables in the models represent feasible paths on a time-space network. The secondly widely used formulation is the network flow problem where variables represent arcs on a time-space network. Other customised formulations are also used by some researchers to satisfy a great variety of train operators' conditions. Here the set covering formulation is given.

Let T be the set of trips and J the set of diagrams. Moreover, let c_j be the cost associated with diagram j. Binary variable x_j takes 1 if diagram j is taken and 0 otherwise. The coefficient matrix A means if trip i covered by diagram j, that is, a_{ij} is 1 if diagram j covers trip i, 0 otherwise. The model for the CSP can be formulated as:

$$\text{Minimise } \textstyle\sum_{j \in J} c_j x_j$$

$$s.t. \sum_{j \in J} a_{ij} x_j \geq 1 \quad \forall\, i \in T$$

$$x_j \in \{0,1\}\ \forall\, j \in J$$

The objective is to minimise the costs of selected diagrams. The model is subject to the constraint: each trip i should be covered at least once. This problem is usually solved in two steps. The first step is generating a large enough number of diagrams. The second step is selecting a subset of diagrams which cover each trip at least once while achieving a minimum total diagram cost.

Methods for Solving the Crew Scheduling Problem

The crew scheduling problem is large with respect to the number of trips but also the explosive number of feasible diagrams constructed by trips. Additionally, many integer programming problems are known to be NP-hard and they require high computational effort. (Heil, Hoffmann, & Buschera, 2020) categorised the methods used in the literature since 2000 into four categories: integer programming methods, heuristics, column generation and meta-heuristics. (1) Integer programming methods: a straightforward method to solve the CSP is to enumerate all feasible diagrams and find the optimal combination of feasible diagrams using a commercial solver. However, such a method can only solve instances for small size and the performance is limited by the commercial solver. (2) Heuristics: based on the experience of integer programming, some heuristic rules are used to speed up the methods used to solve the CSP. Lagrangian relaxation of constraints in the CSP is especially widely used. A Lagrangian multiplier produced by optimising a Lagrangian relaxation problem can be used in directing how to choose a better feasible solution with respect to the objective of the CSP. Moreover, a fixing scheme is also frequently used in heuristics to reduce variable numbers in a model of the CSP. (3) Column generation: this is a powerful mathematical technique to solve large-scale optimisation problems. Feasible diagrams are not enumerated once but generated iteratively, which largely reduces the problem size. A master problem is usually solved to update the optimum and a pricing problem is solved to generate promising diagrams to be added into the master problem in the next iteration. (4) Meta-heuristics: meta-heuristics are problem-independent techniques, which do not take advantage of any specific form of the problem and hope to get a global optimum.

Genetic algorithms, ant colony optimisation and tabu search are commonly seen in literature.

Crew Scheduling Software

Railway crew scheduling applications are of interest to national governments or train operators. (Rezanova, 2009) listed some of the important railway applications for different companies. TRACS II in Great Britain (see (Kwan A. , Kwan, Parker, & Wren, 1996), (Kwan A. S., Kwan, Parker, & Wren, 1999), (Wren & Kwan, 1999), (Wren, et al., 2003), et al.), the Italian Railway Company (see (Caprara, Fischetti, Toth, Vigo, & Guida, 1997), (Caprara, Fischetti, Guida, Toth, & Vigo, 1999), (Caprara, Fischetti, & Toth, 1999) et al.), TURNI in Netherlands Railways (see (Kroon & Fischetti, 2000), (Kroon & Fischetti, 2001), et al.), Harmony Crew Duty Rostering in the Netherlands, Resource Management Solution - Rail Crew for Germany, etc. among others. We refer readers to (Rezanova, 2009) for each application and relevant literature.

2.5. Literature Review on Crew Rescheduling

These days, the focus of research is moving from scheduling to rescheduling crew in real-time. Similar to the CSP, crew rescheduling can be mathematically modelled as an optimisation problem of high complexity requiring a combination of heuristics and combinatorial search methods. The crew rescheduling problem has two kinds of constraints. (1) Covering task constraints: all driving tasks should be covered by at least one driver with the required route and rolling stock knowledge. A task can be covered by more than one driver. Then other drivers will be regarded as passengers. (2) Assigning feasible recovery diagram constraints: all drivers need to be assigned a feasible recovery diagram. It is natural to use binary variables to write these two types of constraints. If a task is covered (or a feasible recovery diagram is assigned to a driver), a binary variable should take one, or zero otherwise. For some drivers directly affected by a disruption, their planned recovery diagrams become infeasible due to the disruption. These infeasible planned diagrams need to be repaired to feasible diagrams. For drivers that are not directly affected by a disruption, their diagram is changed to another feasible recovery diagram because the necessary swapping of driving tasks is used to find solutions. For other drivers, their recovery diagram can be the same as their planned diagrams.

To the best of our knowledge, (Walker, Snowdon, & Ryan, 2005) were the first to study the railway driver rescheduling problem. They first built a model to generate timetable and crew diagrams from scratch. Then the model is modified to be used to solve an integrated timetable and crew rescheduling problem. The model is solved by branch and bound and tested on a small-scale network with 3 crew depots.

(Huisman, 2007) considered the driver rescheduling problem after changes are made in the underlying timetable due to infrastructure construction work. The duration of the infrastructure construction work is known, and the problem is solved prior to operation. He used an integer programming model solved by a heuristic method based on Lagrangian relaxation with column generation. To reduce the number of feasible diagrams, Huisman proposed the term “look like” diagrams, which start and end in the same crew depot and the start (end) times of the diagrams should not be much different from those of the original diagrams.

The current state-of-the-art for crew rescheduling for significant disruptions are from (Rezanova, 2009) and (Potthoff, 2010). The approach of (Rezanova, 2009) is based on branch and price and depth-first search (BnPCR). It first solves the linearised integer programming model and then the result from the linearised model is used to find an integer solution with a branch and bound technique. Overall, the branch and price method can be seen as using column generation at each node of the branch and bound tree. The branching strategy is the constraint branching proposed by (Ryan & Foster, 1981), which can guarantee an integer solution by forcing or forbidding a driver to perform a task. Rezanova introduced the terms recovery period and disruption neighbourhood to limit the size of the problem, which is similar to the idea of a "core problem" of (Potthoff, 2010). (Sato & Fukumura, 2010) used an algorithm framework similar to that of (Rezanova, 2009) to solve freight train driver rescheduling in disruption situations. However, they used Dijkstra's algorithm to find a candidate duty as a shortest path problem without considering resource constraints such as meal breaks.

(Potthoff, 2010) considered a real-time crew rescheduling problem as an integer programming problem when a disruption happens during operations. It uses a combined subgradient method to solve the Lagrangian dual problem and a greedy algorithm (GSLR) to choose the minimal cost recovery diagram for each driver. To generate a new recovery diagram by the pricing problem, dual values from the subgradient method are required. He proposed the term “core problem” to include the drivers whose diagrams need to be recovered. For tasks that are not covered in the core problem, a new core problem is built and explored to solve the problem again. (Veelenturf, Potthoff, Huisman, & Kroon, 2012) suggested railway crew rescheduling with retiming. The methodology is based on (Potthoff, 2010)’s core problem and heuristic method. An uncovered tasks list is maintained. In each iteration, an uncovered task is taken from this list and the core problem is expanded by adding retiming options. Another idea from (Potthoff, 2010) is the crew rescheduling under uncertainty. He considered quasi robust solution in the case of track blockage. The computation time for this model can be 4.5 times more than the basic model.

Besides using integer programming, constraint programming (CP) has been successfully used to solve complex combinatorial problems. Compared to integer programming having linear equations and inequalities, constraint programming has arithmetic constraints. CP is mainly solved by constraint satisfaction methods, such as backtracking, constraint propagation and local search. Backtracking is a general algorithm that finds solutions to constraint satisfaction problems. It examines partial candidate solutions, abandons candidates that cannot be developed to a global solution that is consistent and incrementally builds candidates for globally consistent solutions. The idea behind constraint propagation is to make the feasible region tighter so that backtracking search can commit to fewer candidate solutions that will be able to develop to a solution, see (Bessiere, 2006). Usually a conflict detection (CD) and conflict resolution (CR) approach is proposed to find solutions satisfying constraints in constraint programming problems. CD and CR have been applied to a lot of research in real-time timetable adjustments. (Chiang, Hau, Ming Chiang, Yun Kob, & Ho Hsieh, 1998) considered a knowledge-based system to solve railway scheduling using heuristic rules to solve conflicts in time order. (Oetting, Rittner, & Fey, 2013) developed a synchronal algorithm (KEKL) for real-time conflict detection and conflict resolution. As a decision-support approach, KEKL can provide multiple solutions for controllers to choose from and it is non-discriminatory and traceable in order to take the binding EU regulations into account. (Wegele & Schnieder, 2004) proposed a genetic algorithm for automated dispatching of train operations. Its optimisation based dispatching can be seen as a cyclic improvements process which cycles between independent conflict chains with trains involved in the conflict and solving conflicts. (Jacobs, 2004) proposed a new train-regulating procedure ASDIS that couples methods from computer-aided train-path management with an asynchronous approach. It allows an end-to-end line from planning to operation to be established. (D'Ariano, 2008) designed and implemented a decision support system called ROMA of which a conflict detection procedure checks whether the timetable is deadlock- free and detects potential conflicts in a given prediction period. A conflict resolution procedure computes a conflict-free timetable in real-time, compatible with the status of the network. A similar principle to CD and CR has also been used in solving crew rescheduling problems. It uses a repair-based approach that can be regarded as a kind of local search technique to repair infeasible crew diagrams.

(Abbink, 2014) designed a prototype system for real-time railway driver rescheduling by actor-agent techniques. The rescheduling principle is task-exchange. A driver-agent who is directly affected by the disruption acts as a team leader. A team leader starts a task exchange process to solve the conflicts in their diagram. Compared to the work of (Rezanova, 2009) and (Potthoff, 2010), which have an overall mechanism of including affected and possibly affected drivers in one problem, the solution of this prototype system is generated sequentially for each team.

(Verhaegh, Huisman, Fioole, & Vera, 2017), using an idea similar to (Abbink, 2014), created a heuristic algorithm DFID for crew rescheduling during minor disruptions. The idea is to insert an unplanned task into the previously feasible crew diagrams. The solution process can be seen as a depth-first iterative deepening search in a tree, with every node representing a crew schedule with unplanned tasks. The size of this tree is restricted by setting the maximum number of changed diagrams and the maximum number of unplanned tasks and fathoming branches that are not promising. The algorithm proposed by (Verhaegh, Huisman, Fioole, & Vera, 2017) inserts an uncovered task into the diagram without considering the feasibility of the connection with the commencing activity and terminating activity first. The infeasibility of the connection is further dealt with by adding passenger trips to the diagram. DFID from (Verhaegh, Huisman, Fioole, & Vera, 2017), BnPCR from (Rezanova, 2009) and GSLR from (Potthoff, 2010) are the three cutting-edge methods for crew rescheduling. A further comparison among them is made in Section 3.2.4. In general, BnPCR and GSLR are designed to provide a solution for significant disruptions in an overall mechanism. DFID addresses crew rescheduling for minor disruptions and solves the problem in a sequence manner.

None of the literature mentioned above addresses in detail the effects of parameters on the crew rescheduling problem. However, some parameters are crucial in finding a robust solution that can be applied in practice. For example, if communication time is not considered in a method, the solution may fail in implementation because a controller does not have enough time to communicate with drivers about their changed diagrams. Second, some rescheduling actions like work overtime and rescheduling a meal break are commonly used in reality. However, these actions and their effects on solutions are not fully explored in the current literature. Third, a feedback mechanism has not yet been considered: when there is no solution provided by an optimisation tool, no method exists to provide other useful information that may lead to a practical solution. As explained in Section 1.2, a feedback mechanism can be essential in obtaining potential solutions, which is further proved by the experiments in this thesis. Fourth, solving shortest path problems with resource constraints (SPPRC) is widely used to construct recovery diagrams in solving the crew rescheduling problem. However, SPPRC is an NP-hard problem, and it may take a long time to solve it to optimality. In the current literature, indications on how to solve SPPRC or how to speed up its solution process are not clearly explained. Fifth, studying an integrated crew and rolling stock rescheduling problem and obtaining multiple solutions are not comprehensively covered by the current literature. This research explores the integrated rolling stock and crew rescheduling problem and uses multicriteria methods to obtain multiple solutions.

CHAPTER THREE: CREW RESCHEDULING FOR MINOR DISRUPTIONS

In this chapter, the crew rescheduling problem for minor disruptions is addressed. This chapter answers the research questions 2 and 3 posed in Section 1.3. The modelling process of the problem and method to solve the problem is presented. The conditions to choose drivers to be rescheduled and detailed constraints like meal break requirements, working hours in the model are also explained. An objective considers four factors is proposed. Moreover, the impact of parameter values on solutions is explored and answered. A feedback mechanism is introduced to improve the solvability of the method.

As described in Section 2.3, minor disruption involves delay and congestion, often without a specific incident occurring. When minor disruption happens, the Train by Train Management process is initiated, which is mainly conducted by the controllers from the TOC who actually runs the train(s) affected by the disruption. First, the controllers identify the disruption and assess the level of it. Second, the controllers select a Train by Train Plan and agree it with the IM controllers. Then the plan is deployed. The disruption is monitored and if the disruption is expected to increase in severity a Service Recovery Framework Process is initiated.

A minor disruption can render the planned crew diagrams infeasible by causing drivers on affected trains to miss the start of one of their scheduled tasks, called an uncovered task in this situation. When a disruption happens, it can cause two types of crew diagram-related conflicts. One is a spatial conflict, where a driver cannot be at the origin of the next train-based activity of their diagram due to cancelled trains. Another conflict type is temporal, where a driver cannot begin their next train-based activity on time due to delayed trains, that is the train is not there for them to operate. In both cases, a crew controller can remove one or more successive train-based activities from a diagram that is infeasible due to disruption to make it feasible. The removed train-based activities, (i.e., uncovered tasks) will be tackled by the approach outlined in this chapter.

To request a driver to work overtime, delay a scheduled meal break and even delay a task by a few minutes is common to obtain an overall good solution in real practice which complies with regulations (Office of Rail and Road, 2013). Hence, as a decision support tool, an ideal method should be able to relax or remove some of the constraints, such as fixed departure and arrival times of train services in the timetable or the impossibility of overtime work. Also, an ideal method should be able to produce multiple solutions and allow crew controllers to choose the best practical solution. Therefore, a flexible method is proposed to solve the crew rescheduling problem for minor disruptions based on depth-first search and heuristic rules (DFSCR). DFSCR considers drivers working overtime, rescheduling breaks if the initial break opportunity is affected and delaying driving tasks within a given bound to obtain multiple solutions for crew controllers to choose from. It considers parameters such as connection time between activities (the time needed to change physical position to start a new activity), extra time for taxi journeys being required during heavy road traffic and maximum daily work duration to mimic the real rescheduling environment.

The third research question posed in Section 1.3, is to evaluate how the parameters in the crew rescheduling problem affect solutions and how to set a feedback mechanism to adjust parameter values when there is no solution. This question is addressed in this chapter. (Christian & Hanno, 2019) described some reasons why project collaborations between academics and industry may fail to implement a mathematical optimisation-based solution or product. For example, considering important parameters such as working hours and overtime, setting a high value for these parameters may not be acceptable and setting them at a low level may not reflect the flexibility of the real situation. In addition, a practically feasible solution that is often used in reality may not be obtainable from the optimisation model because some constraints are strictly set in the optimisation model. Therefore, it is better to leave some parameters to the user to adjust. However, due to lack of experience or detailed understanding of an optimisation decision support tool, some combinations of parameters set by the user may cause the model to give no solutions or unreasonable solutions. A sensitivity test for this study is conducted on five parameters to explore the effects and bounds that give reasonable solutions. DFSCR has a feedback mechanism to generate feedback from one model run, analyse the reasons for not having a solution and initiate the corresponding relaxation to adjust the parameters to rerun the model automatically.

The structure of the remainder of the chapter is as follows. Section 3.1 outlines a method (DFSCR) to solve the crew rescheduling problem for minor disruptions. In Section 3.2 DFSCR is applied to a realistic scenario and the outcomes are compared against existing state-of-the-art methods. Section 3.2 also describes the results of sensitivity tests on parameters used in DFSCR. Section 3.3 reports the results of testing 382 scenarios using DFSCR and its feedback mechanism. Section 3.4 is the conclusion.

3.1. Method: Depth-First Search Crew Recovery Method (DFSCR)

The DFSCR approach is inspired by the method of (Verhaegh, Huisman, Fioole, & Vera, 2017) using the idea of swapping tasks among a list of drivers to solve an uncovered task. In their work, the feasibility of inserting an uncovered task into a diagram is considered later in the algorithm. The present work addresses the feasibility of inserting an uncovered task into a diagram at the beginning of the algorithm to reduce the computational requirements. This is achieved by introducing the ideas of a commencing activity (the activity immediately before an uncovered task) and a terminating activity (the activity directly after an uncovered task). Finding a driver to cover the uncovered task means being able to find suitable commencing and terminating activities in a driver's diagram. By finding the commencing activity and terminating activity on a driver's diagram for the uncovered task, the DFSCR approach can filter drivers and find feasible drivers that have a higher possibility to cover the uncovered task. Finding a solution is a process of adjustment where inserting a task into a diagram might result in further uncovered tasks. In the algorithm ApplyDFID shown in (Verhaegh, Huisman, Fioole, & Vera, 2017), it is unclear if there are still tasks left uncovered when a solution is saved. Thus, a solution with conflicts may be saved but such a solution may not solve the rescheduling problem in practice. To address this problem, an uncovered task list is maintained in DFSCR, and solutions are saved when the list is empty to guarantee there are no tasks left uncovered. Moreover, some other important constraints are considered in DFSCR. The communication time to contact drivers is considered, as the possibility of delaying a driving task and to specify some rules that allow a planned break to be rescheduled if it is affected by inserting an uncovered task. Enough communication time ensures that drivers can receive the changes in their diagrams and retiming is useful when there are no other solutions. Sensitivity tests are conducted to explore the effects of parameter values on solutions. Such sensitivity tests can be used to develop guidance to help controllers to set suitable parameter values that give the best chance of obtaining solutions and to give a better understanding of the changes in solutions for the same problem when some changes are made in parameter values. Also, a feedback mechanism is added to improve the probability of finding a solution. Overall, DFSCR is designed to provide conflict-free solutions to assist controllers in rescheduling crew. Various parameters are considered, and controllers can set initial values for them to mimic a real rescheduling environment. The results obtained from sensitivity tests and the feedback mechanism can further help controllers to understand and use the optimisation tool. Developing and testing a set of guidelines derived from the sensitivity tests collaboratively between controllers and academics is recommended.

3.1.1. Overview of DFSCR

As explained in Section 2.5, crew rescheduling problems have two types of constraints specifying that: (1) every task should be covered, (2) every driver should be assigned a feasible diagram. Corresponding to the two types of constraints are two types of conflicts that may arise.

The first type of conflict with these constraints is if there is a driving task that is not covered by any driver. This conflict can in reality be detected by crew controllers in three ways.

- First, uncovered tasks are simply generated during the very short-term planning adjustment phase before operations. Usually, it is a list of uncovered tasks for the next day’s operations.
- Second, uncovered tasks can be generated by converting an infeasible diagram to a feasible one. If a driver is late for their next driving task, a crew controller can remove the remaining driving tasks from their diagram. These removed driving tasks should be considered as uncovered tasks.
- Third is that if the railway is on a limited operation due to restrictions following disruption, such as a speed restriction or an unavailable track section. Once the restriction is lifted, railway operation will gradually return to normal by adding more trains into operation. Each newly added train service needs to be split and assigned to drivers as driving tasks. These driving tasks need to be inserted into the current crew schedule.

The second type of conflict is when a driver’s diagram becomes infeasible to implement due to disruption. Crew controllers can capture this conflict by assessing disruption impact on crew planned diagrams which may become infeasible in real-time operations by communicating with drivers that are in disrupted trains or mapping real train running against the planned timetable. Note that the second type of conflict can be converted to the first type by a crew controller who converts an infeasible diagram to a feasible one by creating more uncovered tasks.

A series of disruptions can happen on an operational day. For each disruption, a number of uncovered tasks that happen at different stations and times can be generated. There are three basic priority orders in which to solve them (see (Chiang, Hau, Ming Chiang, Yun Kob, & Ho Hsieh, 1998)). (1) station ordering, where uncovered tasks are solved station by station. At each station, uncovered tasks are solved by the departure time. (2) train order, where uncovered tasks are solved train after train and (3) time order, where uncovered tasks are solved by their departure time.

The DFSCR approach addresses resolving the first type of conflict by accepting one uncovered task as input. This work does not design a specific conflict detection system to find conflicts or study the best priority order to solve uncovered tasks for one disruption. It is assumed that crew controllers can find initial conflicts, find uncovered tasks from conflicts with or without the use of operations support software and input uncovered tasks to DFSCR in their chosen order. DFSCR focuses on giving conflict-free solutions to each problem of inserting an uncovered task in turn.

For each uncovered task, DFSCR uses a combined local search and backtracking techniques to insert the uncovered task into the current crew schedule. DFSCR starts with a complete crew schedule with one uncovered task. First DFSCR uses heuristic rules to search for suitable drivers diagrams that have a high probability of being able to accept the uncovered task. Inserting the uncovered task into one driver's diagram may lead to more uncovered tasks, which can be seen as follow-up conflicts. Then each newly generated uncovered task will be inserted into an appropriate diagram based on their departure time order until no uncovered task is left. Thus, a solution to the problem of inserting an uncovered task into a current crew schedule is a series of drivers 'diagrams that have been revised to feasible by swapping tasks and one of the drivers 'revised diagrams contains the initial uncovered task and no uncovered task is left. To produce multiple solutions, a depth-first search is utilised by DFSCR. Depth-first search is a specific form of backtracking related to searching the tree structure. It starts at the root and explores as far as possible along a branch before backtracking. A branch represents a solution to inserting the uncovered task into the current crew schedule.

The cost function f for a solution in the DFSCR model considers four factors: total taxi time (TT), total work overtime (WO), number of used drivers (UD) and total planned task delay time (PTDT). Taxi time, working overtime and planned task delay time are measured as the overall time length of taxi trips, overtime and planned task delay time required in a solution, respectively. Number of used drivers is the overall number of drivers used in a solution. All four factors are summed together with weights as shown in Equation (3.1).

$$f = \alpha_1 TT + \alpha_2 WO + \alpha_3 UD + \alpha_4 PTDT \qquad (3.1)$$

Weights α_1, α_2, α_3, α_4 can be easily set by users. This study sets α_1, α_2, and α_4 to 1 and α_3 to 300 to demonstrate the method. These four factors are chosen in the work for their importance based on discussions with controllers for a GB TOC. Taxi time and work overtime are chosen because they bring financial costs. Number of used drivers is penalised because rescheduling more drivers brings higher implementation risks. Delay is penalised because delaying one train may cause further trains to be delayed. Other factors can also be used to assess a solution. For example, the change of the starting time of a meal break can also be considered. It can be easily added to the cost function but is not deemed essential by controllers.

To summarise, DFSCR can fit into the following decision system flow. Once a conflict is identified by a crew controller, uncovered tasks are found by controllers as explained in the beginning of Section 3.1.1. These uncovered tasks will be input into DFSCR with order of priority. For an uncovered task and the current crew diagrams, DFSCR will give multiple conflict-free solutions. One solution can be chosen by a crew controller and then the solution will be written into the current crew diagrams. Then the next uncovered task is solved with the updated crew diagrams.

3.1.2. Inserting One Uncovered Task into a Diagram

To insert an uncovered task into a diagram, this uncovered task needs to be consistent with the activity before and after. The driver should have enough time to arrive at the uncovered task after their previous activity and enough time to arrive at the next activity after the uncovered task. The connection between activities depends on the geographical and time difference relationship. There are nine different connection types used in the method. They are shown in Table 1. These nine connection types are extracted from planned crew diagrams. They cover all significant activity combinations. To decide about a connection possibility, the geography and time of two activities are considered. A suitable connection type is used depending on whether the previous activity ends and the next activity starts at the same station and whether the time between the end of the previous activity and the start of the next activity is big enough to have a break opportunity. Some connection types contain a break opportunity if the time difference is suitable for a break. However, having a break opportunity is not considered as a separate problem. For an infeasible diagram involving an affected break opportunity, DFSCR repairs this infeasible diagram and then moves onto another infeasible diagram.

Table 1 Connection Types

Type	Description
break	connect two activities with time interval enough for a break
change	connect two activities with time interval not enough for a break but bigger than connection time

imm	connect two driving tasks within one train with time interval bigger than 0
pass	take a driving task as a passenger ride
breakPass	take a driving task as a passenger ride with time interval (excluding the passenger ride) enough for a break
breakDoublePass	take two driving tasks as a passenger ride with time interval (excluding the passenger ride) enough for a break
changeDoublePass	take two driving tasks from different trains as a passenger ride
doublePass	take two driving tasks from the same train as a passenger ride
taxi	take a taxi to relocate

In the remaining part of Section 3.1.2, two important concepts: commencing activity and terminating activity are defined. Using these two concepts, a method for finding a set of possible drivers that can take up an uncovered task is described. There can be three results of inserting an uncovered task into a driver's diagram, depending on how the meal breaks are affected and on the newly generated uncovered tasks. These results are: infeasible, conditionally feasible and unconditionally feasible. If a meal break is affected by the insertion, certain rules of resetting an adjusted break are used to resolve this.

Commencing Activity

Suppose that there is an uncovered task that must be inserted into a driver's diagram. Then for each of the currently working drivers 'existing diagram, a suitable commencing activity is searched, defined as the last activity on the diagram to which the uncovered task could be appended. The commencing activity for a given driver is determined in three steps, as follows.

First, the activity of the driver when the rescheduling process starts (labelled the current activity) is identified. There are two types of current activity. One is that the rescheduling start time lies between two activities on the driver's diagram, then the current activity is defined as the last activity which took place before the rescheduling start time. The other is that the rescheduling start time lies inside the duration of an activity, which in this case is defined as the current activity.

Second, the communication time required to contact the driver to inform them about their changed diagram is considered. From the rescheduling start time, a driver must remain at a station longer than the communication time to get the changed diagram message and communicate whether they either accept or refuse it.

So from the end of the current activity to the start of the last activity on the driver's diagram, two consecutive activities A and B are looked for, such that between the end of activity A and the start of activity B, the driver stays at the station longer than the communication time. If such a pair of activities can be found, then activity A is called the initial commencing activity. It is permitted to delay the uncovered task by less than a given maximum planned task delay time to find an initial commencing activity for a driver. The above two steps are shown in Algorithm 1.

The initial commencing activity is the earliest possible choice of commencing activity and will be updated later (after the termination activity is identified, see section "Terminating Activity") to find the optimal commencing activity. In general, the optimal time for the commencing activity is as late as possible so as to minimise the total disruption to the driver's diagram. The commencing activity can be found by looping between the initial commencing activity and the terminating activity.

```
Algorithm 1 find initial commencing activity(driver,uncoverTask)
currActivity=get activity at rescheduling start time(driver)
for ( activity=currActivity, activity<the last activity in diagram) do
    informTime= max{rescheduling start time, activity arrival}
    startMove=informTime+communication time
    if (startMove≤next activity departure) then
        if (connection exists between activity and uncoverTask) then
            return activity
        end if
    end if
end for
```

Algorithm 1 Finding the initial commencing activity

Terminating Activity

Following the insertion of a new uncovered task into a diagram, an algorithm searches from the end of the diagram for the terminating activity, defined as the earliest activity that has a connection from the uncovered task to it. It is found in two steps. First, the terminating activity is set as the sign off of this driver if there is a feasible connection from the uncovered task to sign off.

Second, the terminating activity is updated by looping between the initial commencing activity and sign off to find an activity that happens as early as possible. The commencing activity is then identified by looping backwards from the terminating activity towards the initial commencing activity until the algorithm finds an activity which has a suitable connection to the uncovered task. The time between the commencing activity and terminating activity will be as small as possible.

The activities between the commencing activity and terminating activity on this driver's diagram are called the affected activities. Different actions are taken depending on the type of affected activity. The first type is a driving task, which will be added to the uncovered task list. The second type is a break, which should be considered for replacement by an adjusted break activity. The third type is a passenger trip, for which no action is needed. DFSCR looks for a commencing activity and a terminating activity that are as close as possible to an uncovered task to minimise the number of affected activities, targeting the rescheduling of fewer drivers. It is worth noting that a preceding activity or succeeding activity rather than the nearest activities may lead to a better solution and DFSCR may need to consider this in future.

Finding a Set of Possible Drivers for an Uncovered Task

Two conditions should be present for a driver to be able to cover an uncovered task. First, the driver should have the route and rolling stock knowledge required by the uncovered task. Second, both a valid commencing activity and terminating activity in the driver's diagram should be found. After finding the possible drivers, they are sorted by a score computed using the method of (Verhaegh, Huisman, Fioole, & Vera, 2017). Based on this method, it is intuitively clear that if a driver has a lower score, it means that this driver has a higher chance of taking the uncovered task.

Three Possible Results of Inserting an Uncovered Task into a Driver's Diagram

There are three possible results of inserting an uncovered task into a driver's diagram: an infeasible, a conditionally feasible, or an unconditionally feasible diagram. There are three reasons for an infeasible solution: a) a break is affected, and it cannot be replaced with an adjusted break, b) the uncovered task to be inserted was removed from this driver's diagram previously, c) inserting this uncovered task will cause a previously inserted task to be removed. The last two conditions are to prevent a cycle being formed in the solution process. If there is at least one driving task that is affected, the insertion result is said to be conditionally feasible. It means that by inserting one uncovered task into a crew diagram, some other driving tasks need to be removed from this diagram. If there is no driving task affected, the result is unconditionally feasible.

Resetting an Adjusted Break

If a driver's break is affected by the insertion of an uncovered task, there are two ways to adjust the break opportunity. One is that there may be enough time for a driver to have a break between the commencing activity and the uncovered task or between the uncovered task and the terminating activity. That is, the connection between commencing activity and uncovered task or uncovered task and terminating activity contains a break opportunity. The other way is that after the terminating activity there is enough time to insert a break. However, a break must be arranged before a driver works more than the maximum working time without a break.

3.1.3. Algorithm for DFSCR

The algorithm for the DFSCR approach is shown in Algorithm *2*. The input consists of the planned crew diagrams, adjusted timetable, rolling stock diagrams and an initial uncovered task. The output of DFSCR is a set of solutions, each of which inserts the uncovered task into current crew diagrams in some way.

An uncovered task list is created and initialised with the initial uncovered task. All the possible drivers for this task are found and ordered with a score in step 1. The driver with the lowest score from the possible driver list in step 3 is chosen. The uncovered task is inserted into this driver's diagram in step 4. Three results are possible: infeasible, unconditionally feasible and conditionally feasible.

If the result is unconditionally feasible in step 5, the uncovered task is removed from the list in step 6. In step 7 if there is no task left uncovered, then a solution is saved in step 8. If there is a task left uncovered and the tree size is not exceeded in step 9, then in step 10 the uncovered task is updated as the first task in the uncovered task list and solveDFSCR is called again. The parameter maxDisruptDrivers is used in steps 9 and 15 to check if the tree size is exceeded. The depth of the tree represents the number of drivers whose diagrams are revised to solve the disruption. This depth should be no more than the parameter, maxDisruptDrivers.

If the result is conditionally feasible, in step 14 the newly generated uncovered tasks are added into the uncovered task list and this list is sorted by increasing order of departure time of all elements. If the tree size limit is not exceeded in step 15, the uncovered task is updated as the first task in the uncovered task list and the problem is solved for this uncovered task and the solveDFSCR is called again in step 17.

After each insertion, an undo step is performed in step 20, the changes made to diagrams in the insertion in step 4 are undone. The newly inserted task into the diagram will be removed and the removed tasks will be added back. Since as many solutions as possible are preferred to be found, an undo step will reverse the changes in the drivers' diagrams even if a solution is found and the algorithm will check the next possible driver (step 2 in Algorithm *2*) until there is no possible driver left for this uncovered task.

Algorithm 2 solveDFSCR(uncoverTask, maxDisruptDrivers)

Input: planned crew diagrams, adjusted timetable and rolling stock diagrams, uncovered task

```
 1: drivers=find all possible drivers for uncoverTask and rank them by scores
 2: while at least one driver in drivers do
 3:   driver=get a driver with the lowest score; remove this driver from the
      driver list
 4:   insertRes=insertTaskToDiagram(uncoverTask, driver)
 5:   if insertRes=UNCONDITIONAL then
 6:     remove uncoverTask from uncoverTaskList
 7:     if (uncoverTaskList is empty) then
 8:       save a solution
 9:     else if (not exceed tree size or cut due to fathoming) then
10:       update uncoverTask, maxDisruptDrivers
11:       solveDFSCR(uncoverTask,maxDisruptDrivers)
12:     end if
13:   else if insertRes=CONDITIONAL then
14:     remove uncoverTask from uncoverTaskList, add new generated uncov-
        ered tasks into uncoverTaskList
15:     if (not exceed tree size or cut due to fathoming) then
16:       update uncoverTask, maxDisruptDrivers
17:       solveDFSCR(uncoverTask,maxDisruptDrivers)
18:     end if
19:   end if
20:   undo(insertRes)
21: end while
```

Output: a number of solutions to the problem of inserting an uncovered task into the current crew diagrams

Algorithm 2 Solving DFSCR

It is addressed that first a solution is obtained after checking whether the list containing uncovered tasks is empty, which guarantees that the solution is conflict free. Second, the rules for resetting an affected meal break are given. Third, the term commencing activity and terminating activity is proposed to find the drivers with a higher possibility to cover a task. Here it is emphasised that DFSCR (Algorithm 2) is designed to find multiple solutions, so when the algorithm reaches a leaf node, it takes one or more steps backwards to develop a new branch to find another solution. It searches for solutions until all possible solutions are found. Two critical aspects of Algorithm 2 are discussed: defining and limiting the tree size and ranking the solutions.

Limiting the Tree Size and Fathoming

The depth of the search tree is limited by the number of drivers allowed to be rescheduled. This condition is checked in steps 9 and 15 in Algorithm 2. Rescheduling many drivers to solve one uncovered task is not practical. So a maximum number of affected drivers is set to control it. A partial solution that affects more than the maximum number of drivers should be abandoned. Also, tree size is limited by fathoming which means pruning some solutions. An upper bound of the method is updated as a new lower solution cost appears. When a partial solution has a bigger cost than the current upper bound, it is pruned to limit the tree size.

Ranking the Solutions

Multiple solutions can be ranked based on total cost, total taxi fee, total work overtime, number of used drivers and total delay time. Note that a solution involving significant retiming of a task is not preferable by DFSCR and will most probably be ranked last based on the total delay time. Practically then, where there is no better solution, crew controllers have a chance to evaluate this solution and see if its induced conflicts can be easily solved manually. Then such a solution can be helpful.

3.1.4. Generating Feedback from the Model Run

When there is no solution returned, it may be the case that minor adjustments to some parameters yield solutions. Feedback can provide hints as to which parameters should be adjusted to obtain solutions potentially. Thus, the feedback generation ability from the DFSCR approach is described in Section 3.1.4. There are three kinds of feedback generated by the model:

(1) The requirement to delay the uncovered task by more than the maximum task delay time.

(2) A driver needs to work overtime to the extent that exceeds the maximum work overtime or maximum work length.

(3) More drivers' diagrams need to be rescheduled to find a solution.

Feedback types (1) and (2) can be generated from finding possible drivers for an uncovered task (step 1 in Algorithm 2). Feedback type (3) can be obtained by checking if the partial solution exceeds the tree size (steps 9 and 15 in Algorithm 2).

Adjusting Parameters According to Feedback

The following relaxation types correspond to feedback types (1) – (3).

Relaxation A Suppose that an uncovered task must be delayed by more than the maximum task delay time. Then feedback information of type (1) is generated. This means that this driver is still late for the delayed uncovered task when the rescheduling starts, considering the communication time and connection time. Since delaying a task could lead to further delays, adjusting the communication and connection times is preferable.Therefore, in Relaxation A, communication time and connection time are reduced. In the experiments, the communication time is reduced from 20 to 10 minutes, and the connection time is reduced from 10 to 3 minutes. Crew controllers suggested the minimum values of these parameters. These minimum values may vary from one operator to another.

Relaxation B Suppose a driver could work more than the overtime or work length limit that has been set. Then feedback information of type (2) is generated. The appropriate relaxation is to increase maximum work overtime and maximum work length. In the experiments, the maximum work overtime is increased from 2 hours to 2 hours and 20 minutes and the maximum work length is increased from 10 hours to 10 hours and 20 minutes. Crew controllers suggested the default values (2 hours and 10 hours, respectively). Relaxing both of them by 20 minutes may give the controllers new solutions that can be discussed with drivers. It is not likely to violate the fatigue rules (but these rules vary from one operator to another).

Relaxation C This is imposed when more drivers need to be used (feedback information of type 3). Then the appropriate relaxation is to increase the maximum number of affected drivers. In the experiments, the maximum number of affected drivers is increased from 2 to 3.

Two combined ways to relax parameter values are also utilised to test the feedback mechanism of DFSCR.

Relaxation AC After Relaxation A, if the number of solutions is less than 3, a further Relaxation C is added.

Relaxation BC After Relaxation B, if the number of solutions is less than 3, a further Relaxation C is added.

Note that there is no particular reason why these combinations of relaxation types are chosen and why a further relaxation is added if the number of solutions is less than 3. They are just reasonable examples of combinations of relaxations that may be tried. Other parameter values and combinations of relaxation types can be used. Further testing is needed to determine the optimal parameter values and combinations of relaxations.

3.2. Single Case Study and Sensitivity Tests

Section 3.2 uses the DFSCR approach to resolve a late inbound train scenario that disrupts the crew diagrams. Later, the results with two modified versions of the methods of (Rezanova, 2009) and (Potthoff, 2010) applied to the same scenario are compared.

3.2.1. Late Inbound Train Scenario

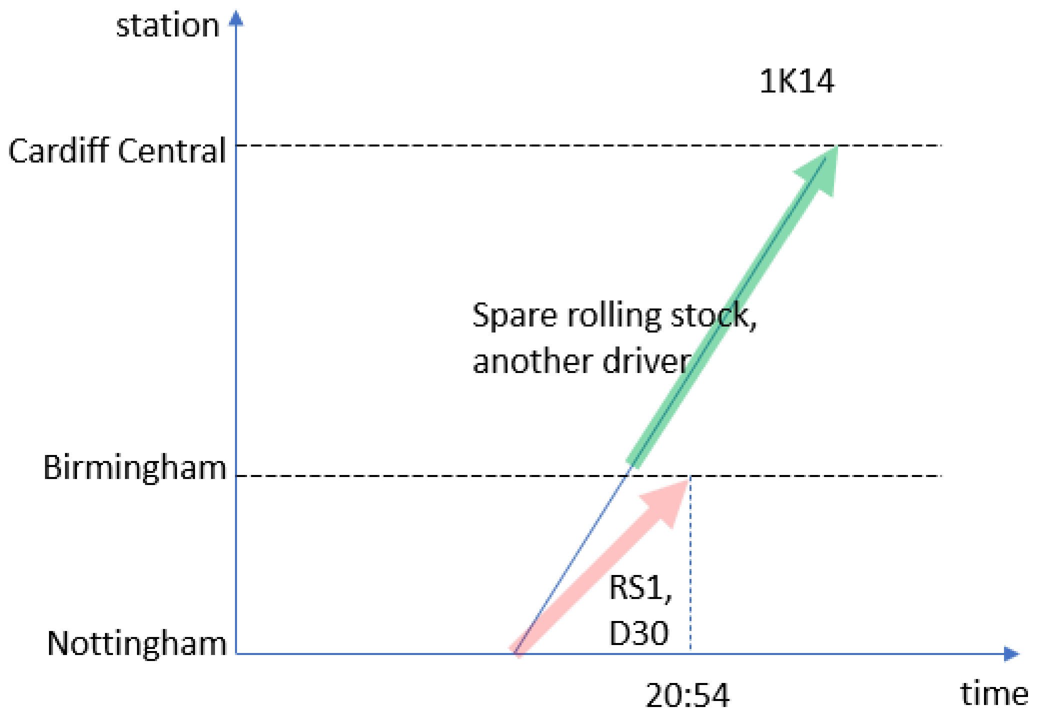


Figure 6 Late inbound train scenario

The scenario is illustrated in Figure *6*. In this scenario, a train (1K14) departs from Nottingham and travels towards Cardiff Central via Birmingham New Street. It is scheduled to be driven by two different drivers (D30 and another driver) who swap at Birmingham New Street. When the train arrives late at Birmingham New Street, both the first leg driver (D30) and rolling stock (RS1) are late. Spare rolling stock is typically available in Birmingham New Street and can be used by the second leg driver to start the train leg from Birmingham New Street on time. The arrived rolling stock RS1 can be treated as the new spare rolling stock. The problem is that the late first leg driver (D30) is late for the next driving task on their diagram. A common solution is to find another driver to cover all the remaining driving tasks on this late driver's diagram.

Assume at 20:09 it is observed that the train 1K14 is late and is estimated to arrive at 20:54 at Birmingham New Street. Table 2 shows the driver of the affected section's diagram. Task ID 23 represents the driving task of the late train service leg on the driver's diagram. Since their next driving task departs at 20:49, driver D30 is unable to perform this driving task from Birmingham New Street to Nottingham, which are Task ID 24 and the following task ID 25. Here tasks 24 and 25 are combined and treated as one uncovered task, 24-25, which must now be covered.

Table 2 Driver diagram: D30

ID	origin	departure	destination	arrival	activity type	headcode	driver ID
19	LESTER	15:24	LESTER	15:24	signOn		D30
20	LESTER	15:50	BHAMNWS	16:46	Driving	1L43	D30
21	BHAMNWS	17:49	DRBY	18:36	TAXI		D30
22	DRBY	18:39	DRBY	19:19	Break		D30
23	DRBY	19:39	BHAMNWS	20:24	Driving	1K14	D30
24	BHAMNWS	20:49	NTNG	22:14	Driving	1H22	D30
25	NTNG	22:26	LESTER	23:00	Driving	5M75	D30

26	LESTER	23:43	LESTER	23:43	signOff		D30

In the solution process, there are nine parameters whose default values are in brackets.

1. **connection time to non-sign off activity** (10 minutes): a driver needs time to cross the platform to start the next driving task or go to the crew depot to have a break. This time is necessary and should be considered to connect two activities.
2. **maximum work length** (10 hours): the maximum permitted time of a driver's diagram for the day.
3. **maximum continuous work length without break** (3.5 hours): the maximum time a driver can work without having a break.
4. **maximum work overtime** (2 hours): the maximum time by which a driver can exceed the scheduled working time.
5. **maximum planned task delay** (10 minutes): the maximum amount of time by which a task may be delayed.
6. **rescheduling start time** (20:09): the time when the rescheduling process starts.
7. **communication time** (20 minutes): a necessary time allocated to contact the driver to inform them about their changed diagram.

8. **taxi variation** (0 minutes): variations in the taxi travel time required between any two stations.
9. **maximum number of affected drivers** (2): the maximum number of drivers whose diagrams are permitted to be rescheduled to solve the problem.

The default values of all these parameters were discussed and confirmed with the experts from train operating companies. Also note that five of these parameters (connection time to non-sign off activity, maximum work length, maximum work overtime, rescheduling start time and taxi variation) are varied in the sensitivity tests within the ranges shown in the figures in Section 3.2.5.

3.2.2. Solution Analysis

The solutions obtained using the same parameter settings as in Section 3.2.1 to this problem are shown in Table 3. Taxi duration estimates are obtained from Google Maps. Work overtime is calculated as the difference between the new sign off and old sign off times. DFSCR finds all 9 solutions in 5.6s. The cost of a solution is the sum of work overtime, taxi duration, the penalty for drivers that have an adjusted diagram and planned task delay minutes.

None of the solutions require a delayed task. The first solution uses the available driver D219 to take task 24-25. Other solutions request driver D25 to take task 24-25 and, in exchange, another driver needs to take task 569 for D25.

Table 3 Results obtained from default values

solution index	driver1	insert task1	driver2	insert task 2	total work overtime	taxi duration	adjusted diagram penalty	solution cost
1	D219	24-25			75	58	300	433
2	D25	24-25	D219	569	75	71	600	746
3	D25	24-25	D209	569	97	71	600	768
4	D25	24-25	D30	569	75	118	600	793
5	D25	24-25	D153	569	122	71	600	793
6	D25	24-25	D127	569	75	118	600	793
7	D25	24-25	D115	569	89	118	600	807
8	D25	24-25	D69	569	149	71	600	820
9	D25	24-25	D125	569	149	118	600	867

Some details of solution 6 from Table 3 are shown in Table 4. Driver D25 takes the initial uncovered task 24-25 and then takes a taxi trip to sign off. This driver needs to work overtime. In exchange, D127 takes driving task 569 from D25 and takes a taxi to sign off. This driver does not need to work overtime.

Table 4 A typical solution (6): recovery diagrams and cost breakdown

driver ID	ID	origin	departure	destination	arrival	activity type	headcode	overtime	taxi	total cost
D25	564	BHAMNWS	15:35	BHAMNWS	15:35	signOn		0	0	0
D25	565	BHAMNWS	16:09	LESTER	17:08	Driving	1O11	0	0	0
D25	566	LESTER	17:18	TYSLSDG	18:53	Driving	1M24	0	0	0
D25	567	TYSLSDG	19:08	BHAMNWS	19:33	TAXI		0	0	0
D25	568	BHAMNWS	19:36	BHAMNWS	20:16	Break		0	0	0
D25	24-25	BHAMNWS	20:49	LESTER	23:00	Driving	1H22/5M75	0	0	0
D25		LESTER	23:10	BHAMNWS	24:08	taxiAdded		0	58	58
D25	571	BHAMNWS	24:11	BHAMNWS	24:11	signOff		75	0	75
penalty for using the driver										300
D25 cost										433
D127	1271	LESTER	14:57	LESTER	14:57	signOn		0	0	0
D127	133	LESTER	20:50	BHAMNWS	21:45	passengerAdded	1M55	0	0	0
D127	569	BHAMNWS	21:55	TYSLSDG	22:16	Driving	5P87	0	0	0
D127		TYSLSDG	22:26	LESTER	23:26	taxiAdded		0	60	60

D127	1272	LESTER	24:38	LESTER	24:38	signOff		0	0	0
penalty for using the driver										300
D127 cost										360
total cost										793

taxiAdded/PassengerAdded means that it is added by us to this driver's diagram to relocate the driver.

3.2.3. Mathematical Optimisation for the Crew Rescheduling Problem

As mentioned in Section 2.5, BnPCR and GSLR are the current state-of-the-art methods for solving the crew rescheduling problem during significant disruption. Both methods are applied to solve the late inbound train scenario presented in Section 3.2.1. The models, to which the methods are applied, slightly differ from each other since (Potthoff, 2010) considers a penalty for task cancellation. A work overtime function is added to both methods and the maximum work overtime and maximum work length are set to have the same default values as in Section 3.2.1. Further, BnPCR and GSLR solve the problem in a disruption neighbourhood (core problem), which is a neighbourhood that includes critical drivers and their tasks during the recovery period. However, the first disruption neighbourhood (core problem) built by both methods cannot solve the problem even when permitting working overtime. Expanding disruption neighbourhood (core problem) rules are added by extending the recovery period by two hours and adding more drivers who appear in the same station as the uncovered task around the same time automatically when there is no solution. With these modifications, both methods give solution 1 in Table 3 after expanding once and allowing drivers to work overtime. BnPCR finds the solution in 3.8s and it finds an integer to the linearised model directly. GSLR finds the solution in 1.7s.

3.2.4. Comparing Methods: DFSCR, DFID, BnPCR and GSLR

In this section, DFSCR is compared to the three cutting-edge methods: DFID from (Verhaegh, Huisman, Fioole, & Vera, 2017), BnPCR from (Rezanova, 2009) and GSLR from (Potthoff, 2010) according to the following aspects.

1. **Number of disruptions**. DFSCR can process multiple uncovered tasks produced by multiple disruptions. In this study, it only takes one initial uncovered task as input. However, it can be easily modified to process multiple uncovered tasks, by initialising the uncovered task list in Algorithm 2 with multiple uncovered tasks. Note that multiple uncovered tasks are not considered at the same time, but each uncovered task is considered singly in sequence. DFID can process multiple uncovered tasks in a similar manner. BnPCR and GSLR can process multiple disruptions since they build a set partitioning problem / set covering problem with all affected drivers as constraints. BnPCR and GSLR aim to find a recovery diagram for any driver that has an infeasible diagram due to disruption.
2. **Number of solutions**. DFSCR and DFID can generate multiple solutions because it builds a tree structure to search for solutions and each leaf of the tree represents a solution. BnPCR first solves a linearised crew rescheduling model.

When the optimal solution to the linearised model is fractional, BnPCR uses a branch and bound tree to develop multiple integer solutions by forcing or forbidding a driver to take a task. However, the appearance of fractional results depends on the scenario and costs of connection types set by the user. It is not easy to control by the user. Also, fractional results appear rarely in the experiments of (Rezanova, 2009). GSLR solves the problem with an advanced Lagrangian heuristic algorithm iteratively, it generates one solution at every iteration. Each objective is the same during the solution process, so the solution with the minimal objective value is the final solution.

3. **Delay a single task**. Delaying any task is not preferable because it will change the real-time timetable. If a delay happens at a busy station or peak time, it can affect a number of trains and quickly propagate through the network. If a delay happens at a rural station or off-peak time, it most probably will not lead to further disruption. DFSCR can consider delaying a task by less than the maximum planned delay time. DFID does not consider delaying a task. However, some modifications can be added to change this in DFID. BnPCR and GSLR do not consider the possibility of this. They both treat tasks as fixed nodes in a graph and a diagram for any driver is a path in this graph. So, if a node is moved, this change is the same for any driver, which is not realistic. If a driver requests this task to be delayed to cover it, this does not mean that another driver would request the same. (Veelenturf, Potthoff, Huisman, & Kroon, 2012) deals with a retiming extension of the GSLR method. The model and method proposed by

them considers the possible retiming copies of a task and such copies can be used to produce solutions.

4. **Consideration of overtime**. DFSCR considers the possibility of drivers working over- time and limits it by introducing two parameters: maximum work overtime and maximum work length. DFID, BnPCR and GSLR can also consider the driver working overtime to cover a task. BnPCR and GSLR both use a resource-constrained path finding algorithm to find feasible recovery diagrams. An extra condition can be added to disallow recovery diagrams that violate maximum work overtime and maximum work length constraints.
5. **Reset a break for the driver**. Drivers need to have a break after a certain period of continuous working time. A break is scheduled for the minimum required time according to working rules in a driver's diagram. However, drivers may have their breaks at times that differ from what is exactly scheduled. DFSCR can restore an appropriate break with predefined rules as in Section 3.1.2 if the planned break is affected. DFID can allow a break to be changed when inserting a task into a diagram. BnPCR and GSLR can also allow a break to be changed. Similar to overtime, extra constraints about break time can be considered when generating recovery diagrams using a resource-constrained path finding algorithm based on dynamic programming.
6. **Having a feasible solution before the algorithm finds an optimal solution**. DFSCR, DFID and GSLR can have a feasible solution in this case, but BnPCR cannot

guarantee this because there may be a fractional feasible solution, from which an integer solution cannot be easily recovered.

7. **Considering the feedback mechanism**. DFSCR can generate feedback as explained in Section 3.1.4. DFID, BnPCR and GSLR do not have this feature.
8. **Method running time**. All three methods solve the problem described in Section 3.2.1 in a few seconds. GSLR (1.7s) is quicker than BnPCR (3.8s) in solving this scenario. DFSCR finds 9 solutions in 5.6s.

3.2.5. Sensitivity Tests

One of the reasons that mathematical optimisation techniques may fail to be transferred into practical use in decision support tools is that, due to the lack of transparency of the optimisation tool, the user does not fully understand the optimisation tool or the impact of adjusting its parameters. A user may unknowingly adjust parameters in such a way as to not fully utilise the capabilities of the optimisation model.

Thus, in Section 3.2.5, some parameter values are varied to evaluate how they will affect the solutions for the scenario in Section 3.2.1. Five parameters are adjusted: rescheduling start time, extra taxi time, maximum work overtime, connection time to non-sign off activity and maximum work length. Every time, one parameter is varied in a realistic range and the rest of the parameters are fixed as their default values shown in Section 3.2.1.

Rescheduling Start Time

The rescheduling start time varies from 19:00 to 21:00 with a one-minute step size. How the number of solutions and solution costs change with the rescheduling start time is tested.

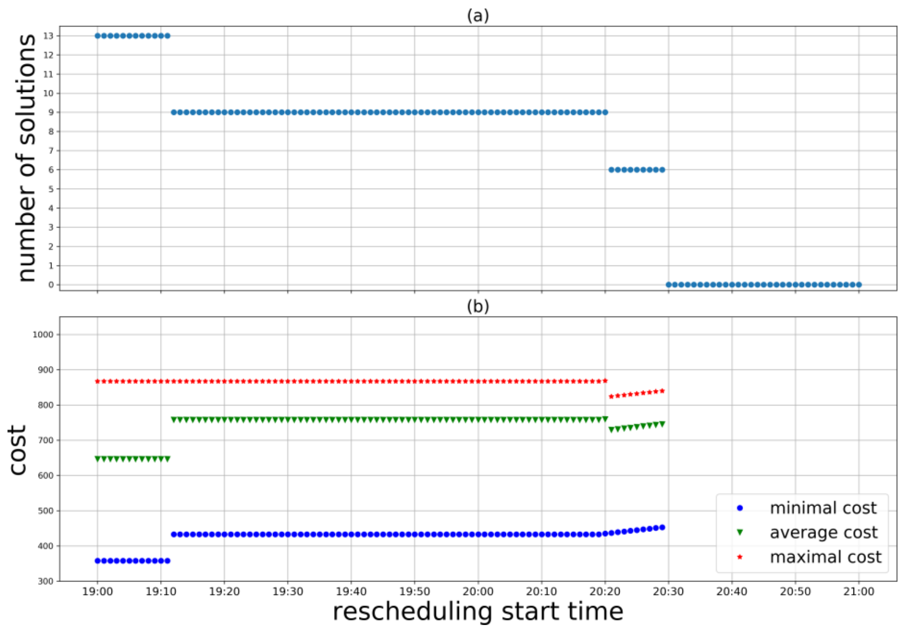


Figure 7 (a) number of solutions, and (b) cost, with respect to rescheduling start time

In Figure *7*(a), the number of solutions drops from 13 as rescheduling starts at 19:00 to 9 at 19:12 and to 6 at 20:21. When the rescheduling starts after 20:29, there are no solutions. The number of solutions drops from 13 to 9 because 4 available drivers in Leicester depot can no longer arrive at Birmingham New Street by taxi in time to take the uncovered task 24-25. Similarly, the drop from 9 to 6 occurs as 3 drivers from Leicester depot cannot arrive at Birmingham New Street to take task 569 from driver D25 in time. When the rescheduling starts at 20:29, task 24-25 is forcibly delayed by the maximum planned task delay time of 10 minutes for all solutions. After 20:29, task 24-

25 needs to be delayed by longer than the maximum planned task delay time to obtain a solution.

Later rescheduling start times affect the solution cost in two ways. One is that it may make some solutions infeasible and the other is that it may require a task to be delayed further. From Figure *7*(b), before 19:12, the smallest minimal cost is 358, which uses a driver from Leicester depot. This driver does not need to work overtime, which reduces the cost.

Extra Taxi Time

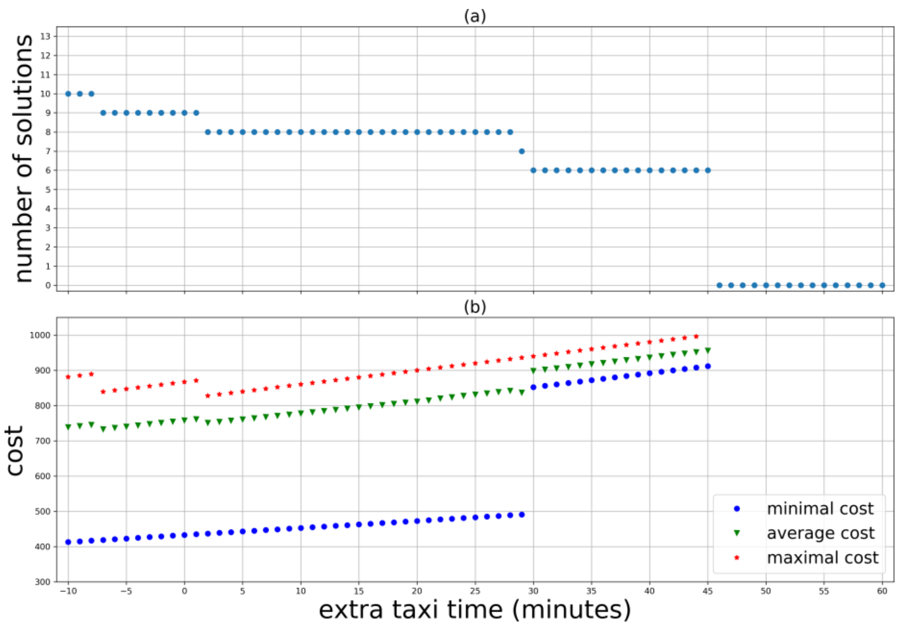


Figure 8 (a) number of solutions, and (b) cost, with respect to extra taxi time

Estimated taxi travel times were obtained from Google Maps. These stored times do not necessarily reflect dynamically changing real road conditions. The number of solutions and solution costs are tested for variable additional time added to the base taxi travel times. The test range is from -10 to 60 with step size 1. The increased taxi times will prevent some drivers from arriving at their scheduled or rescheduled tasks on time. The increased taxi times will also increase work duration. So with the increase in extra taxi time, the number of solutions is decreasing (Figure 8 (a)). When the extra taxi time is greater than or equal to 46, there is no solution. The number of solutions drops from 10 to 9, 8 and 6 because work length or work overtime exceeds the maximum work length or maximum work overtime, respectively.

In Figure *8*(b), when the extra taxi time is less than or equal to 29, the minimal cost solution is using the same available driver. The minimal costs are increasing because the taxi fee is calculated based on the taxi duration. The sudden increase of minimal cost at +30 minutes duration is because this available driver cannot take this uncovered task 24-25 anymore. The maximal costs are not necessarily increasing with increasing taxi duration because some drivers are not available due to the long taxi duration. The maximal cost solutions at every point do not necessarily use the same drivers.

Maximum Work Overtime

The default value for maximum work overtime is 2 hours. Nine solutions appear when the maximum work overtime is greater than 1 hour and 20 minutes. There are no other solutions even when overtime increases up to 5 hours. Correspondingly, the costs also do not change.

Connection Time to Activity (excluding Sign Off)

Connection time will affect the total cost in two ways. One is that longer connection time will cause more overtime work that is part of the total cost. The other way is that longer connection time will cause some drivers to miss their respective passenger trips, thus generating extra taxi fees. Also, longer connection time will make some solutions infeasible due to maximum work overtime and maximum work length constraints.

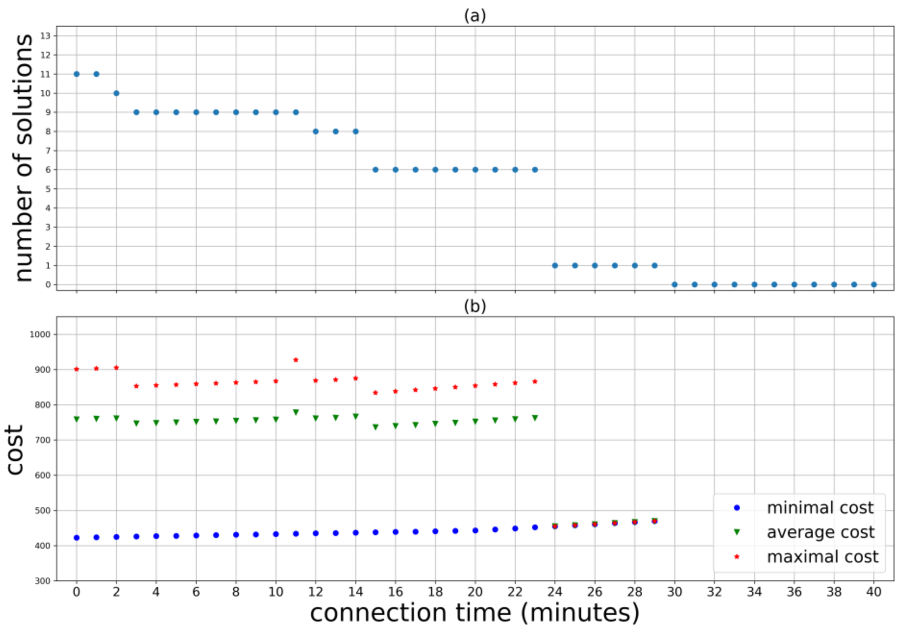


Figure 9 (a) number of solutions, and (b) cost, with respect to connection time

In Figure *9*(a), the number of solutions drops from 11 to 9 because one driver working overtime exceeds the maximum work overtime of 2 hours and another driver cannot take task 569 in time due to increased connection time. The number of solutions drops further from 9 to 6 because 3 available drivers from Leicester depot cannot take task 569 in time. It further drops from 6 to 1 because driver D25 cannot take task 24-25 in time even if this task is delayed by the maximum planned delay time.

The minimal, average and maximal costs in Figure *9*(b) have a stable change at all test points since connection time will affect the work overtime cost. However, a long connection time causes some solutions to become infeasible, so the cost is not always increasing.

Maximum Work Length

In Figure 10, the work length increases from 8 hours to 12 hours, with a 10-minute step size. There is no solution in Figure 10(a) when the maximum work length is lower than 8 hours 40 minutes. When the maximum work length increases, more solutions appear and the smallest minimal cost solution in Figure 10(b) appears when the maximum work time is set at 9 hours and 40 minutes. At most 13 solutions are obtained after the maximum work length is extended. When the maximum work length increases, more solutions appear until the maximum work length is set as 10 hours and 40 minutes. Since the maximum work length is bounded by the maximum permitted overtime work, the availability of solutions is also bounded by this condition.

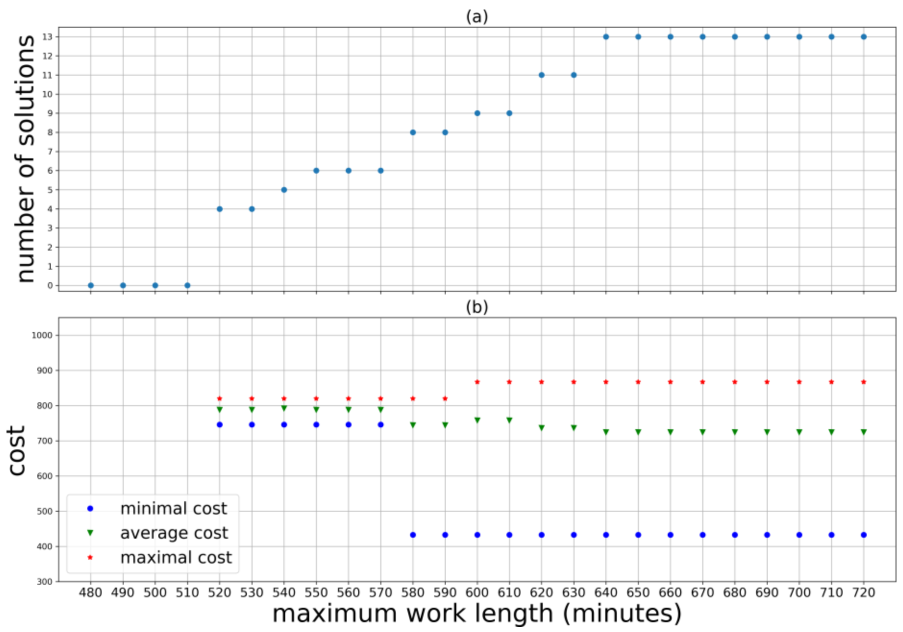


Figure 10 (a) number of solutions, and (b) cost, with respect to maximum work length

3.3. Multiple Experiments

To further test DFSCR's performance over the period of an entire day, DFSCR is applied to all driving tasks of a train operator's one day operation. The dataset used for testing has 124 diagrammed drivers and 24 spare drivers. These spare drivers are located in 3 depots during the day. There are 382 driving tasks in the dataset. For each test, one driving task is duplicated and this duplicated task is assumed as the uncovered task, so DFSCR is applied to 382 tests. Figure *11* shows the effect of inserting these duplicated driving tasks into the current crew diagrams during a day. In general, DFSCR can find solutions for 150 of the tests and the average number of solutions is 4.48. For tests where DFSCR cannot find solutions, the main reason is that the initial uncovered task of a test starts very early or late at a station away from major stations. The chance of finding a chain of suitable drivers swapping tasks among them is low. For each test, the average solution time is 0.75s. The number of tasks that occur every hour increases from early morning to reach its peak during 6:00-7:00. Then it fluctuates along the day until reaching another peak during 16:00-20:00. Average solution time has a similar shape to the number of driving tasks. When there is a big number of tasks, a corresponding number of drivers are on duty. When an uncovered task happens at a peak time, more suitable drivers are also considered in DFSCR to find solutions. Thus, the computational time for this test is also large. In general, DFSCR can generate around 2 solutions for each test from 09:00 to 21:00 with a smaller number of solutions at early or late times.

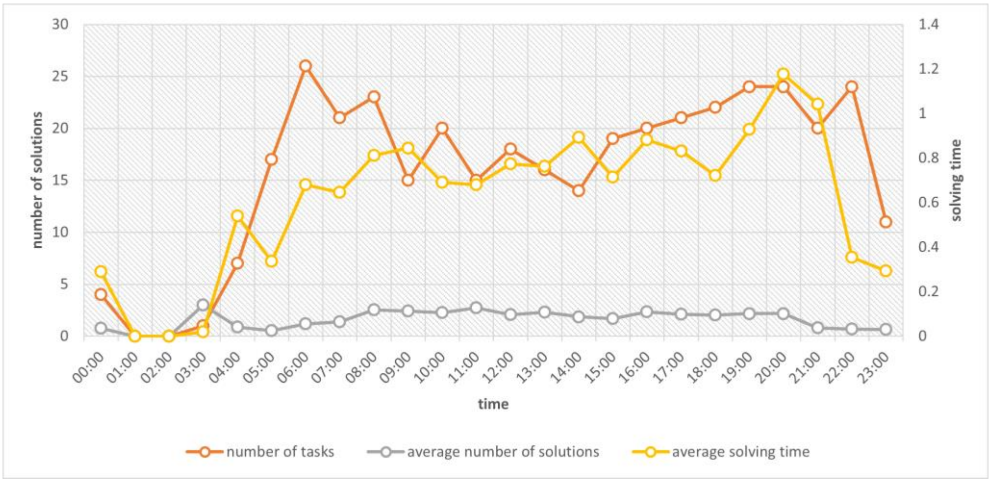


Figure 11 Average number of solutions and solution time against task departure time

To test how the maximum number of affected drivers (MNAD) affects solutions, MNAD is increased from 1 to 5. Later the results from increasing MNAD are compared to the results from using the feedback mechanism. Table 5 shows how solutions can change with respect to the increasing of MNAD. If MNAD increases from 1 to 5, the solution number increases from 4.21 to 4.95 for tests that can find solutions and the success rate increases by 2.1% for all tests. The average costs of the minimal, average and maximal costs of all tests that have solutions are shown in Table 5. So, the costs, especially the minimal costs, shown in Table 5 may change with respect to MNAD. The minimal, average and maximal costs increase with the increase of parameter values. One reason is that with a bigger MNAD, some tests that do not have a solution with a lower MNAD now tend to have solutions. The other reason is that with a bigger MNAD, solutions that involve more drivers appear and these solutions have a higher cost according to our cost function. The solution time increases gradually until it jumps from 0.88 to 1.35 when the MNAD rises from 4 to 5. Based on the nature of DFSCR, the solution time will increase exponentially with the increasing MNAD. However, the solution time is still very small due to fathoming. Fathoming is to prune solutions that have bigger costs than the current lowest cost as in (Verhaegh, Huisman, Fioole, & Vera, 2017), which is explained in Section 3.1.3.

Table 5 Solution changes with increasing maximum number of affected drivers

MNAD	solNum	min cost	ave cost	max cost	solTime(s)	success rate
1	4.21	308.70	325.92	352.01	0.65	38.0%
2	4.48	318.68	346.84	382.67	0.75	39.3%
3	4.81	323.09	359.50	399.02	0.82	39.5%
4	4.93	327.68	365.87	408.20	0.88	39.8%
5	4.95	331.82	372.35	418.52	1.35	40.1%

Multiple Testing on Feedback Mechanism

To test the feedback mechanism, the DFSCR approach is built first with the default parameter values in Section 3.2.1 except the rescheduling start time, which is updated to the relevant time for each testing task. The rescheduling start time is set at 30 minutes before the testing task departs.

The results of the DFSCR approach solving all tests with default parameter values and relaxed parameter values are shown in Table 6. After the relaxation, the number of tests successfully solved increases by 6%. The average number of solutions increases by 1.13. In Table *5*, success rates increase by 0.2% when the maximum number of affected drivers is increased from 2 to 3. In Table 6, success rate is increased by 6% using a feedback and relaxation scheme. It is worth noting that increasing MNAD may not be as good as an approach of adjusting parameter values based on the feedback mechanism since the solution time rises sharply in the former case.

Table 6 Comparison between testing results before and after relaxation

	number of tasks(tests)	success rate	average number of solutions	average solution time
before relaxation	382	39.3%	4.48	0.75s

after relaxation	382	45.3%	5.61	1.58s

The detailed relaxation types of all the relaxed tests are shown in Table 7(a). 144 tests are suitable to be relaxed. Among them, 108 tests are relaxed by Relaxation A. 238 tests (Table 7(b)) with no relaxation are tests where the relaxation does not help. However, in 233 tests, drivers require the task to be delayed by more than 30 minutes. In the other 5 tests, the uncovered task happens in the early morning or late at night when no other driver is on duty. The detailed explanation of relaxations can be seen in Section 3.1.4: Relaxation A: reduce communication time and connection time; Relaxation B: increase maximum work overtime and length; Relaxation C: increase the maximum number of affected drivers. Relaxation AC and BC are combinations of the corresponding relaxation types.

Table 7 (a) tests with relaxation, and (b) tests without relaxation

(a) Tests with relaxation					
number of tests with relaxation	A	B	C	AC	BC
144	108	14	6	16	0
(b) Tests without relaxation					
number of tests with no relaxation	delay >30 minutes	no driver is on duty			

238	233	5

3.4.Conclusion

In this section, a new approach was described for solving the crew rescheduling problem for minor disruptions (DFSCR). This new approach aims to produce multiple solutions for inserting an uncovered task into the current crew schedule. To illustrate the applicability of this method, a case study based on a real-world network and crew diagrams has been used. DFSCR has been able to solve the problem and give multiple solutions as expected. To study the influence of parameters used in the model and to find their effective bounds, a sensitivity test has helped explain the reasons for various trends in the results for the same case study. DFSCR was also tested on 382 scenarios to give various statistical information about performance, including the number of tests that can be solved and the running time. The feedback mechanism of DFSCR was also tested and results show that more solutions can be obtained after using the feedback information to relax constraints. DFSCR has a number of limitations, which should be addressed in future work.

First, the biggest limitation is to recognise the scale of disruption impact that can be efficiently fixed by DFSCR. DFSCR relies on crew controllers to recognise uncovered tasks caused by disruption and input these uncovered tasks into DFSCR one after another. This process is not straightforward to crew controllers when many trains are disrupted.

Second, further work is needed to address the follow up conflicts induced by a solution if the solution retimes a driving task. Note that this can have a direct impact on the schedule of a train journey. The knock-on effect of one train's delay can spread over a railway network. A solution containing a shifted driving task should be further examined by a controller to better understand its impact on the remaining operation. The severity of the impact brought by solving an uncovered task is not sure in this way because it depends on the railway rescheduling environment at that moment. If a task is required to be retimed at a major station during peak time, this solution may not be chosen. However, if the same situation happens at a rural station late at night, it may be useful if there is no better solution available. Further, a retimed task may affect rolling stock circulation. However, if this delay time can be absorbed due to robust rolling stock circulation, this solution can be considered. Future work should address the whole delay impact on the model.

Third, the order of various uncovered tasks fed into DFSCR may influence the overall crew rescheduling cost. Optimising the order of feeding disruptions into DFSCR will require further testing.

Fourth, in this work, a feedback mechanism was proposed and proved that it can help to find more solutions. This work did not try to find the optimal parameter values used in relaxations A, B and C, the order of using different relaxation types or the complete combination of relaxation types. This can be a further study point.

In the wider picture of schedule adjustment during disruptions, how to put DFSCR into practice in the current railway operation system is another challenge. Crew rescheduling is manually processed in Great Britain now. DFSCR requires planned crew diagrams and an uncovered task as input. Planned crew diagrams are stored in crew management software and usually crew controllers need to use crew management software to recognise uncovered tasks. Further study is needed to integrate crew management software, traffic management software, DFSCR and crew controllers' actions. Note that this could result in development of a system that would automatically detect conflicts of all kinds and provide dispatchers and crew controllers with automated advice on their resolution, in the spirit of (Stelzer, 2016).

CHAPTER FOUR: CREW RESCHEDULING FOR SIGNIFICANT DISRUPTIONS

This chapter addresses the crew rescheduling problem during significant disruptions. Section 2.3 explains that significant disruptions lead to a loss of or restricted access to parts of the railway network due to an incident, affecting the delivery of a normal timetable. After a significant disruption is detected, a service recovery process is initiated. The IM discusses with affected TOCs. In this dialogue, they will consider the impact of the incident, estimate when normal infrastructure may be available and agree on contingency operation. For significant disruption management, the timetable should be recovered first. The main aim of crew rescheduling is to cover as many train trips as possible in a revised timetable. Since the timetable is recovered to its normal level, the crew should also return to their planned diagrams. The planned crew diagrams are designed to cover the regular timetable. This chapter answers the research questions 2, 4 and 5 posed in Section 1.3.

Since solving the crew rescheduling problem requires a revised timetable as input, using a model to reschedule the timetable after a disruption is first considered. A typical significant disruption is a line blockage, which limits the normal access to a part of the railway network and affects trains that are planned to run over the blockage site. Another typical significant disruptions include signalling failures on busy routes, maintenance overrun, overhead line equipment failures, etc. This section builds a timetable rescheduling model for the specific disruption - line blockage since a variety of unexpected events can lead to line blockage, e.g., damaged bridges, failed rolling stock on track, landslips. Then, the crew rescheduling problem is modelled as a set covering problem to reschedule the crew after the disruption.

The structure of the remainder of the section is as follows. Section 4.1 describes the model for solving the timetable rescheduling problem and presents the results of several experimental tests. Sections 4.2 and 4.3 introduce the mathematical programming and techniques used in the following sections. Section 4.4 describes the model and method for solving the crew rescheduling problem for significant disruptions. Experimental results of using the model and method are presented. Moreover, three different dynamic programming search techniques are used in the method and compared for solving the crew rescheduling problem. Section 4.5 is the conclusion.

4.1.Timetable Rescheduling Problem

As explained in Section 2, the basis of passenger railway operations is the timetable which describes a set of train services scheduled to run from one terminal station to another and call or pass at several intermediate stops at specific times. Due to unplanned events, trains often cannot run on time as set out in the timetable. This section aims to provide a model to recover the timetable for complete line blockage, for which a train service recovery process needs to be initiated.

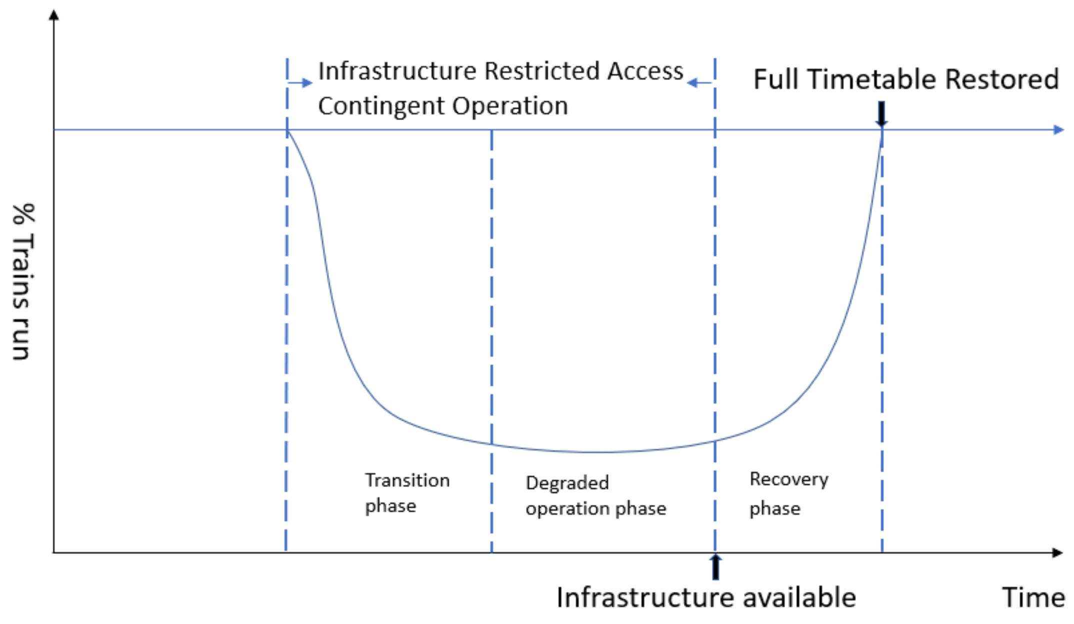


Figure 12 Three stages of the train service recovery process

In Section 2.3.1, the train service recovery framework process has been described from the perspective of the involved organisations and actors. Here, the network status (infrastructure availability and the number of trains running) of the train service recovery process is expressed. Figure 12 shows the three stages of the train service recovery process: the transition phase, degraded operation phase and recovery phase. Following a disruption, railway operation is at a restricted access state in which total infrastructure capacity cannot be provided to enable the running of a regular timetable. A reduced timetable is used during the degraded operation phase. After a certain amount of time, the infrastructure for the regular timetable will again become available and normal running can be gradually restored. In the transition phase, some trains are directly cancelled or delayed due to the initial disruption. Trains start to queue up, and a quick decision is needed to prevent delay propagation through the network. In the degraded operation phase, a revised timetable based on a contingency plan adapted for the specific circumstances may be used. In this phase, the number of trains running on the network is the lowest. During the recovery phase, the regular running of train service is gradually restored, and the number of trains running on the network rises.

This section aims to provide a revised timetable which covers the three phases of train service recovery. Two time periods are defined for use in the work. One is the recovery period during which the disrupted timetable is recovered. The recovery period is regulated by rescheduling start and end times. After the recovery period, the method guarantees that operations can return to a normal timetable as long as the infrastructure and rolling stock required to run a full timetable are available. After the recovery period, train operations on the network should be at the point “full timetable restored”, shown in Figure 12, with all initially scheduled services running. How to set the rescheduling start and end time is explained in Section 4.1.2. The other is the blockage period during which no train can pass the blockage area, and the infrastructure is at a restricted access state, as shown in Figure 12, infrastructure restricted access. The blockage period is regulated by blockage start and end times, which are usually taken as an assumption. Usually, the recovery period starts as the blockage period starts and ends later than the blockage period ends.

An integer programming model is introduced to revise a timetable in case of a complete blockage taking into account infrastructure and rolling stock availability. Solving the model requires a revised timetable covering the three phases of train service recovery. Operations are guaranteed to return to normal after rescheduling. Keeping the planned rolling stock circulation patterns is preferred, by penalising solutions that do not use the designed patterns. Finally, the model is successfully applied to the busiest line of a TOC in Great Britain. Later the results of the revised timetable will be used as input to solve the crew rescheduling problem in Section 4.4.6.

The structure of the remainder of the section is as follows. Section 4.1.1 reviews the research on timetable rescheduling. Section 4.1.2 is the problem description. Section 4.1.3 introduces how trains running on a network can be modelled. Section 4.1.4 proposes a timetable rescheduling model in case of a complete blockage. Section 4.1.5 is a simple example, and Section 4.1.6 applies the model to 14 scenarios using actual data from a TOC in Great Britain. Section 4.1.7 concludes the timetable rescheduling topic.

4.1.1. Related Works

A large body of literature has investigated timetable rescheduling for railway disruption management. The literature closely linked to the timetable rescheduling model proposed is first reviewed in this section. Note that in the literature reviewed here, the trains running on a railway network are modelled by event-activity graphs.

(Louwerse & Huisman, 2014) investigated the rescheduling of a railway timetable in cases of partial or complete blockage. They used an event-activity graph to describe trains running on a double-track line at a macroscopic level. Rolling stock inventory and circulation were considered so trains could take rolling stock compositions from excess inventory or an early arriving train at the same station. An integer programming model was built to minimise train delays and cancellations and balance trains in both directions and over the operation period. Two real-world cases on the Dutch railway were tested to prove that all instances can be solved optimally within one minute. (Veelenturf, Kidd, Cacchiani, Kroon, & Toth, 2016) also used an event-activity graph to solve the railway timetable rescheduling problem during blockages based on (Louwerse & Huisman, 2014). In their paper, the disruption management process takes place from the start of the disruption until the operation is fully restored. They considered a more complex railway network with open track sections, which can be single-tracked, double-tracked or contain more parallel tracks. The tracks can be used in both directions. (Veelenturf, Kidd, Cacchiani, Kroon, & Toth, 2016) took into account track, rolling stock composition capacity and train routings. The objective is to minimise train cancellations and delays. The model was tested successfully on many complete and partial blockage scenarios.

The model in this section is based on the model used in (Louwerse & Huisman, 2014). In the present work, an event-activity graph is also used to model trains running on a double-track line. (Louwerse & Huisman, 2014) focused on the second phase (degraded operation phase) of the train service recovery process and aimed to provide a degraded timetable for this phase. They assumed the network was empty when the disruption started and aimed to provide a new cyclic timetable for the degraded operation phase. They do not consider modifying the timetable in the transition phase. To use their solution for the degraded operation phase, some manual checks and further modifications may be required to implement the solution. Only after the necessary manual modifications to the revised timetable, can it be used as input to reschedule rolling stock and crew. In the present work, rescheduling the timetable for the three stages of the train service recovery process is considered. The railway network is not assumed to be empty when the rescheduling starts. When the rescheduling ends, the number of rolling stock compositions at each station must be the same as what is required for running a regular timetable afterwards. Thus, the running of a regular timetable can be recovered, as shown in Figure 12. The output, a revised timetable, can be directly used to reschedule rolling stock and crew. Moreover, the model proposed in this chapter minimises train cancellations, delays, and changes to the rolling stock circulation patterns. Thus, a solution with the fewest changes to the rolling stock circulation patterns while minimising cancellations and delays is preferred. Such solutions are easier to communicate, and more likely to be accepted and implemented by local dispatchers.

Passenger-oriented timetable revision is also becoming increasingly popular. Its objective is usually to minimise passengers' travel time and the operation cost of train operators. (Zhu & Goverde, 2020) proposed a mixed integer linear programming model which applies to dispatch measures of retiming, reordering, cancelling, flexible stopping, short turning trains, handles stock circulation at short-turn and terminal stations and takes into account the station capacity. An adapted fix-and-optimisation algorithm is developed to minimise the travel time and the number of transfers. (Zhan, Wong, Shang,

& Lo, 2021) simultaneously rescheduled trains and passenger routes from both operators’ and passengers' perspectives. The integrated train rescheduling and passenger routing problem is formulated as a mixed integer linear programming problem. Then the integrated problem is decomposed into two subproblems using the alternating direction method of multipliers (ADMM). Each subproblem can be solved effectively with a dynamic programming algorithm.

4.1.2. Problem Description

A railway network consisting of stations and double-track sections is considered in the problem. A blockage occurs on both tracks between two stations, and the tracks are entirely blocked. It is assumed that the duration of the blockage period is known, that is, the exact times for blockage start and end, t_{block_start} and t_{block_end} are known. A train uses a rolling stock composition for its full service, after which the rolling stock composition is moved to a shunting yard or used by another train. A rolling stock composition describes the number of individual rolling stock units and in which order they appear in a train. It is worth noting that rolling stock composition is considered as a whole in the problem. The compositions are not split during the day. Trains that have already passed their last scheduled stop before the blocked segment at the moment when the disruption occurs are assumed to continue to run as planned. The rescheduling start time $t_{reschedule\_start}$ is set the same as t_{block_start} , and the rescheduling end time $t_{reschedule\_end}$ is set for a fixed period, such as two hours after t_{block_end}.

The rescheduling end time should be set later than the expected blockage end time so that the infrastructure can restore a normal timetable. It is tricky to guarantee that rolling stock is available at the recovery period's end. During the train service recovery process, the rolling stock may be assigned to run different trains compared to the original plan due to the disruption. The number of rolling stock composition at each station may differ from the planned number. Rolling stock balance is introduced to describe the difference between the actual number of rolling stock compositions and the scheduled number of rolling stock compositions at the end of the recovery period at each station. The rolling stock balance at each station should be zero to guarantee the availability of rolling stock for the operations after the recovery period. Also, services run by rolling stock at the rescheduling end time must be completed on time.

A rolling stock turning pattern is planned at each terminal station. It means that each incoming train is matched with an outgoing train. Applying a designed turning pattern can ease the task of local planning. Locally changing the turning pattern by matching an incoming train to a different outgoing train can require shunting operations and needs to be communicated and agreed upon with the local controllers. Thus, a penalty is added to the objective if such changes to the planned rolling stock pattern happen. The rescheduling timetable model proposed aims to minimise train cancellation and delays, and violations of planned rolling stock circulation.

Overall, this section solves the timetable rescheduling problem at the macroscopic level. It uses rescheduling strategies, including train cancellations and retiming, while considering infrastructure capacity and rolling stock availability constraints to recover train services for the whole process.

4.1.3. Event-activity Graph

The network model of trains running on the network is first introduced to formulate the timetable rescheduling problem. An event-activity graph is used to model trains running on a network. Each train runs from one terminal station to another, passing or stopping at intermediate stations. Trains that are planned to run through the blockage site during the blockage period should stop at a turnaround station before approaching the blockage site. Rolling stock used by the trains can be used for other trains that depart from the turnaround station. A turnaround station is defined as the last stop station before approaching the blockage site, where there are the necessary tracks for trains to change direction. The events and activities used in the event-activity graph are described in what follows.

Event

There are three types of events: train departure events, train arrival events and inventory events. A ***train departure/arrival event***, denoted as e, is one movement record from a journey. It has attributes including station, time and the associated service. The set of all such train departure/arrival events is denoted by E.

An ***inventory event*** at the station s, denoted i_s, represents the number of stationary rolling stock compositions that can be used for services that depart from this station at $t_{reschedule\_start}$. It contains attributes like the station and the number of rolling stock compositions. The set of all such inventory events is denoted by I.

Activity

There are five kinds of activities: train activities, headway activities, inventory activities, circulation activities, and dummy activities.

Two consecutive train events from one train form a ***train activity***. A train activity starting with a train departure event and ending with a train arrival event represents a movement from one station to another. A train activity starting with a train arrival event and ending with a train departure event represents the train dwelling at a station. The set of all train activities is denoted by A_{train}.

A ***headway activity*** is a period reserved between two train events to guarantee safe running. The set of all headway activities is denoted by A_{head}. It models the minimum safety time between two trains running on the same track or dwelling on the same platform.

An ***inventory activity*** is connected by (1) an inventory event to a train departure event; (2) a train arrival event to an inventory event. (1) means that services departing from a terminal station can take stationary rolling stock in the station of the inventory event. (2) means that rolling stock used for trains arriving at a terminal station can become stationary in the station of the inventory event.

A ***circulation activity*** is formed by connecting a train arrival event to a train departure event at turnaround or terminal stations. Note that circulation activities should be created for two trains that are from different train groups. Each train group contains trains that depart and end at the same station. Trains from the same group depart with a specific frequency (for example, one train per hour) with exceptions during peak time. The set of all turnaround activities is denoted by $A_{circuit}$.

Dummy activities are created for trains that run over the disrupted area. For train v that is scheduled to run over the disrupted area, let k be the train v's last stop before the blockage site and let l be the train's first stop after the blockage site. Let e_k^{arr} and e_l^{arr} denote the train arrival events at the station k, l respectively. Let e_k^{dep} and e_l^{dep} , denote the train departure events at the station k, l respectively. e^{dep} and e^{arr} are the train departure and arrival events for the train v from its first and last station. Thus, the train v can be partitioned into three trains, v^{α}, v^{β} and v^{γ}, where v^{α} has events from e^{dep} to e_k^{arr}, v^{β} has events from e_k^{dep} to e_l^{arr} and v^{γ} has events from e_l^{dep} to e^{arr}. A dummy activity is created from e_k^{arr} to e_k^{dep} and e_l^{arr} to e_l^{dep}. Note that a dummy activity is not only created between trains v^{α} and v^{β}, $v^{\beta}$and v^{γ} from the same train, but also if v^{α} and v^{β}, v^{β} and v^{γ} are defined as parts of different trains but from the same train group. The set of all dummy activities is denoted by A_{dummy}.

A train event $e \in E$ has incoming activities and outgoing activities. Using A_{in}^{e} $=\{a$: a is an incoming activity for train event $e\}$ to denote incoming activities. Similarly, A_{out}^{e} $=\{a$: a is an outgoing activity for a train event $e\}$. Notations $A_{in}^{i_s}$ and $A_{out}^{i_s}$ are defined similarly for inventory events i_s at the station s. The essential elements used in an event-activity graph to simulate trains running on a network are summarised in Table 8.

Table 8 Basic elements used in an event-activity graph

Parameters	Explanation
v	train v
e	train event e
i_s	inventory event i_s at station s
a	train / headway / inventory / circulation / dummy activity a

The illustration of a simple event-activity graph on a double track line with two terminal stations is given in Figure 13. It is worth noting that inventory events can be created for any station if there is stationary rolling stock at the station between the rescheduling start and end time.

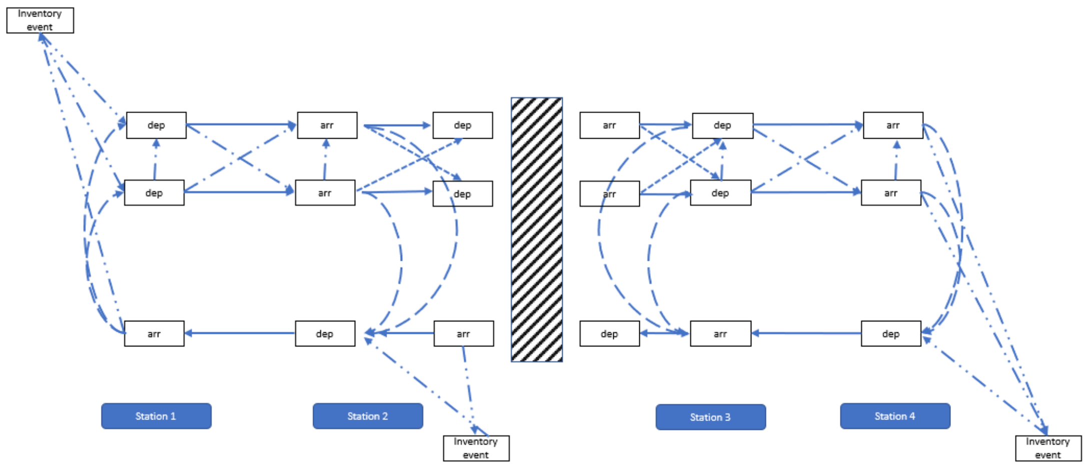


Figure 13 Event-activity graph built for a complete blockage

In Figure 13, rectangles correspond to events. Train activities are shown with solid lines. Dashed lines with single dots represent headway activities. Inventory activities are dashed lines with double dots. Dashed lines show rolling stock circulation activities. Dummy activities are shown with dotted lines. The shaded rectangle indicates the blockage site.

4.1.4. Model

For train event e, an integer variable x_e is introduced. x_e represents the replanned occurrence time of the train event e and takes an integer value that regulates the train event time to minutes. For train v, let y_v be 1 if a train v is cancelled and 0 otherwise. For each activity a, let z_a be 1 if the activity a is cancelled; 0 otherwise. Here a can be any of the activity types mentioned above. The variables used to formulate a timetable rescheduling model are listed in Table 9.

Table 9 Variables used in a timetable rescheduling model

variable	Explanation
x_e	the occurrence time of train event e
y_v	It takes 1 if train v is cancelled, 0 otherwise
z_a	It takes 1 if train / headway / inventory / circulation / dummy activity a is cancelled, 0 otherwise

Constraints

(a) Event time constraints

Trains should depart and arrive at a given time. The event e, t_e is the planned event occurrence time. A train should not be replanned more than t_{delay} minutes later than initially scheduled, a fixed number set by controllers, or depart earlier than planned.

$$t_{delay} \geq x_e - t_e \geq 0 \quad \forall e \in E \tag{4.1.1}$$

(b) Duration constraints

Notation $|a|$ is used to show the length of an activity a. Generally, it can take different values depending on a. For train activity a, the planned duration $|a|$ can be obtained from the timetable; the value $|a|$ should be at least the same as the scheduled train activity duration, see constraint (4.1.2) below. However, for headway, inventory, circulation and dummy activities a, $|a|$ depends only on the type of the activity and has a fixed value. For headway, inventory, circulation and dummy activities, $a = (e, f)$. If the activity is cancelled ($z_a = 1$), there is no constraint on the occurrence time of events e and f. However, if the activity is taken ($z_a = 0$), the two events take place. The occurrence time of the event f should be later than the occurrence time of the event e plus $|a|$: see constraint (4.1.3). M stands for a big enough constant number, which guarantees that constraint (4.1.3) holds if $z_a = 1$.

$$x_f - x_e \geq |a| \quad \forall a = (e, f) \in A_{train} \tag{4.1.2}$$

$$x_f - x_e + Mz_a \geq |a| \quad \forall a = (e, f) \in A_{circuit} \cup A_{inv} \cup A_{dum} \tag{4.1.3}$$

(c) Headway constraints

To formulate headway constraints, for each headway activity $a = (e, f)$, variables λ_{ef} and λ_{fe} are introduced. Variable λ_{ef} is defined as taking 1 if the event e happens before f and 0 otherwise. Similarly, λ_{fe} takes 1 if the event f happens before e and 0 otherwise. Let v_e denote the train which contains the train event e and v_f denote the train which includes the train event f. If an event e happens before f, f can only occur after e plus the minimum headway time $|a|$: see constraint (4.1.4). Suppose an event f

happens before the event e, the event e can only occur after the event f plus minimum headway time $|a|$; see constraint (4.1.5). If neither train v_e and v_f is cancelled, λ_{ef} or λ_{fe} should take 1: see constraint (4.1.6).

$$x_f - x_e + M\left(1 - \lambda_{ef}\right) \geq |a| \quad \forall a = (e, f) \in A_{head} \tag{4.1.4}$$

$$x_e - x_f + M\left(1 - \lambda_{fe}\right) \geq |a| \quad \forall a = (e, f) \in A_{head} \tag{4.1.5}$$

$$\lambda_{ef} + \lambda_{fe} + y_{v_e} + y_{v_f} \geq 1 \quad \forall a = (e, f) \in A_{head} \tag{4.1.6}$$

(d) Inventory constraints

The number of stationary rolling stock compositions at the station limits the number of inventory activities taken at a station.

$$\sum_{a \in A_{out}^{i_s}} (1 - z_a) \leq n_s \quad \forall i_s \in I \tag{4.1.7}$$

(e) Rolling stock circulation constraints

When the train v_e arrives at a terminal or a turnaround station s, its rolling stock may be used for another train v_f or become stationary at this station. For the train v_e to depart from s, it should have one rolling stock composition to support its service. Otherwise, it should be cancelled. This rolling stock can come from another train v_f or rolling stock stationary at station s. Denoting by E_s^{arr} the set of all train arrival events at a station s and by E_s^{dep} the set of all train departure events from a station s, the following constraints are obtained:

$$\sum_{a=(e,f)\in A_{circuit}\cup A_{inv}\cup A_{dum}} (1-z_a) = 1 - y_{v_e} \quad \forall e \in E_s^{arr} \qquad (4.1.8)$$

$$\sum_{a=(f,e)\in A_{circuit}\cup A_{inv}\cup A_{dum}} (1-z_a) = 1 - y_{v_e} \quad \forall e \in E_s^{dep} \qquad (4.1.9)$$

Equation (4.1.8) means that for the train associated with the train event e arriving at the station s, its rolling stock can be used for another train when a rolling stock circulation or dummy activity is taken, or its rolling stock can become stationary when an inventory activity is taken. Equation (4.1.9) means that the train associated with the train event e departing from the station s can obtain its rolling stock from another train when a rolling stock circulation or dummy activity is taken, or rolling stock is stationary when an inventory activity is taken.

(f) Cancelling trains during blockage constraints

The set of trains planned to pass the blockage site during the blockage period is denoted by V_{block}. Each train v in V_{block} should be cancelled, that is:

$$y_v = 1 \ \forall \ \ v \in V_{block} \qquad (4.1.10)$$

(g) Departed trains cannot be cancelled.

A train that has already departed when rescheduling starts, that is $e^{dep} \leq t_{reschedule\_start}$, cannot be cancelled. V_{early_dep} stands for the set of such trains.

$$y_v = 0 \ \ \forall v \in V_{early\_dep} \qquad (4.1.11)$$

(h) Rolling stock balance and timetable recovery

Rolling stock balance means that at the end of the rescheduling period, the number of rolling stock compositions at stations should be the same as the required number from the initial timetable. Rolling stock balance guarantees enough rolling stock compositions at each station to run a timetable for the remainder of the day after rescheduling ends. It implies that operations can return to the normal level, and a regular timetable can be recovered. Let m_s be the required number of rolling stock compositions at station s.

$$\sum_{a \in A_{in}^{i_s}} (1 - z_a) = m_s \quad \forall i_s \in I \tag{4.1.12}$$

The left part of the equation (4.1.12) means that the trains arriving at station s and their rolling stock remain stationary. The accumulated number of such rolling stock should equal the number required at this station to perform a regular timetable after the rescheduling end time. Note that if trains are cancelled in “pairs”, the rolling stock composition is balanced. The term “cancelled in pairs”, means that the number of trains cancelled in both directions is equal between any two stations.

Objective

As discussed before, keeping the planned rolling stock circulation patterns is preferred, which can reduce unnecessary shunting operations. In the objective, a penalty is imposed for circulation, inventory and dummy activities if they violate the planned circulation patterns. A_{wp} is the set of circulation, inventory and dummy activities that contravene the planned circulation patterns. The objective function for the timetable rescheduling problem is the sum of weighted costs of train cancellations, delays and unplanned rolling stock circulations. Variables α, β are the weights for cancellation and delay, respectively. Variable γ_a is the weight for choosing an activity that violates the rolling stock circulation pattern and depends on the activity type. Thus, the model for the timetable rescheduling problem during a complete blockage is:

$$min \sum_{v \in V} \alpha y_v + \sum_{e \in E} \beta (x_e - t_e) + \sum_{a \in A_{wp}} \gamma_a (1 - z_a)$$

$$subject\ \ to\ (4.1.1) - (4.1.12)$$

4.1.5. an Illustrative Example

Figure 14 shows an example of timetable rescheduling in case of a one-hour blockage. Trains are running between stations S1 and S4. A one-hour stoppage (11:00-12:00) appears between S2 and S2B. Trains from both directions short turn at S2 and S2B, respectively. The grey shaded area between S2 and S2B is the blockage site. Four trains are disrupted due to the blockage, and they can be rescheduled as the arrow shows. Since the rescheduling does not affect other trains at a macro level, operations can return to normal at 12:30.

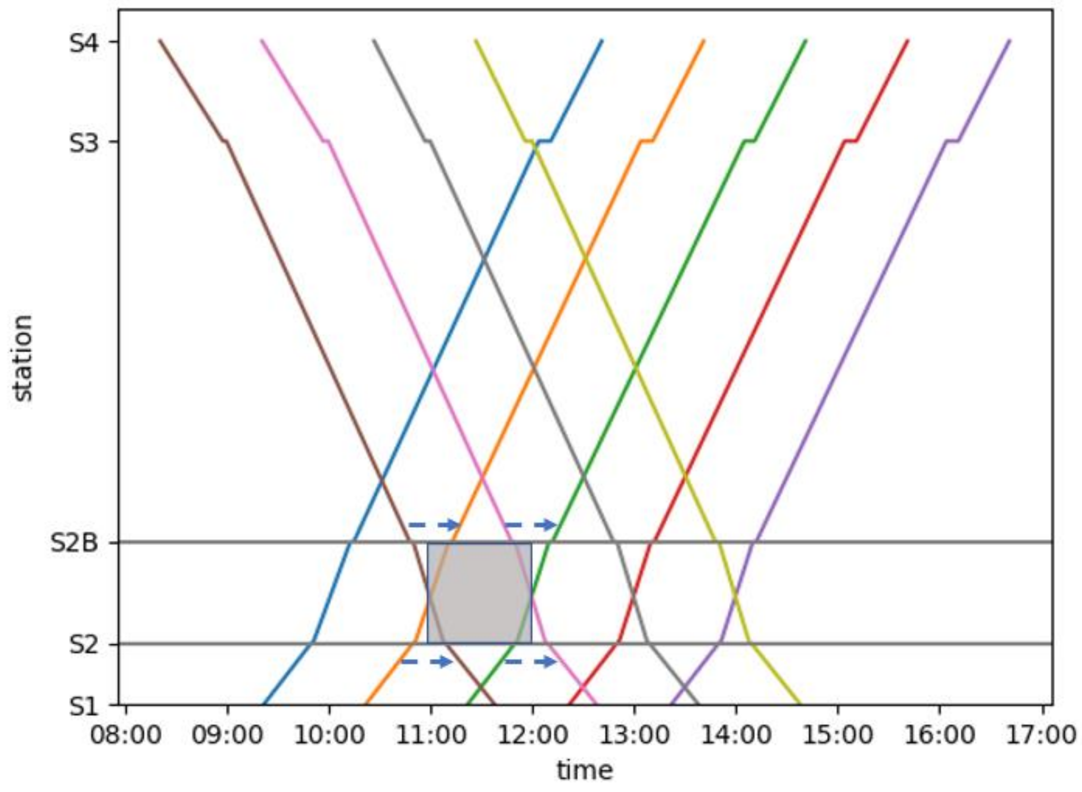


Figure 14 Timetable rescheduling in case of one-hour complete blockage

4.1.6. Experimental Tests

Experiments are carried out using one TOC network in Great Britain. Compared to many European countries, a non-cyclic timetable is used in Great Britain. Extra trains are added on some lines during a day's operations. Thus, a blockage with the same duration may affect trains differently if it occurs at different times on a day. Also, the same rescheduling solution may not be feasible to implement for the same blockage occurring at different times. Thus, a blockage with the same duration occurring at different times of a day is tested. The disruption site is set on its busiest line, which runs between Stations S1 and S2 with 15 trains running from S1 to S4 stopping at S2 hourly and 16 trains running from S4 to S1 stopping at S2 hourly. 18 trains running from S1 to S2 hourly and 16 trains running from S2 to S1 hourly. Most of them depart from the terminal station at the same minutes past each hour with a few exceptions. There are 2 extra trains running from S1 to S3 during afternoon peak time and 1 extra train running from S3 to S1 in the early morning. 1 extra train runs from S2 to S1 during morning peak time and in the evening. 4 trains run from S3 to S4 in the early morning and 3 trains run from S4 to S3 at late night. The trains running on the line are shown in Figure 15.

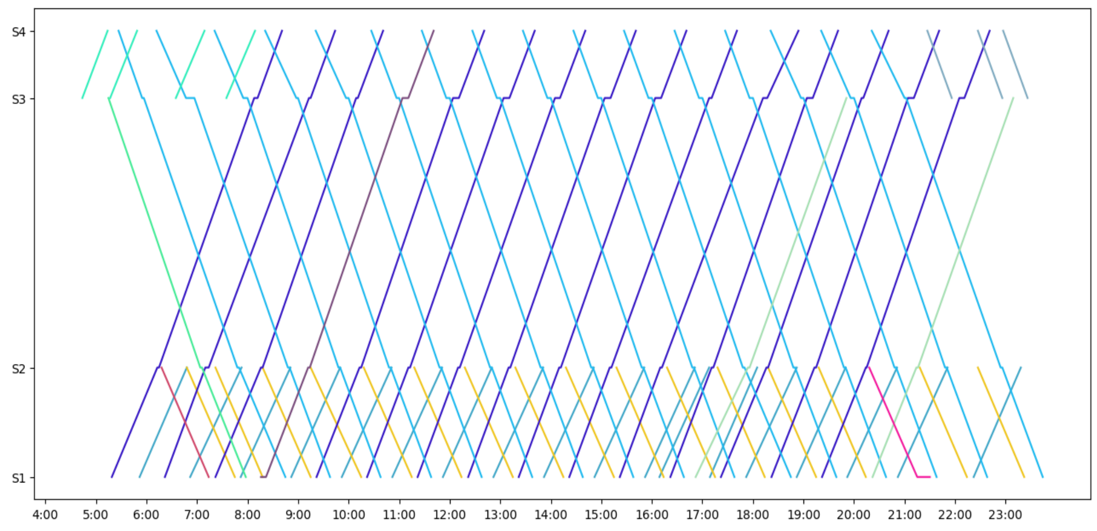


Figure 15 Trains running on the busiest line of a TOC in Great Britain

Trains Considered in the Model

Directly and potentially affected trains by the disruption during the recovery period are selected to build the timetable rescheduling model. Directly affected trains are trains that run over the blockage area and need to turn around at a station due to the blockage during the blockage period. Trains scheduled to run during the recovery period and use the same rolling stock used by the directly affected trains are potentially affected trains.

Experimental Set up

After studying the planned timetable and discussing with controllers, in the tests, the period for circulation activities is set as 5 minutes. The headway, dummy, and inventory activities period is set as 2 minutes. The cancellation weight α is set as 1000, and the delay weight β is 1. For penalising an inventory or dummy activity which violates the rolling stock pattern, γ_a is set as 10. For penalising a rolling stock circulation activity which infringes the rolling stock pattern, γ_a is set as 50. A rolling stock circulation activity violation has a higher cost due to our experience during experimental tests. It is noticed that in a station where the inventory event exists, keeping the rolling stock turning pattern is prioritised over using spare stationary rolling stock. The model is solved with CPLEX V12.10.0 on a computer with 16 GB RAM and 1.99GHz. The main aim of the experiments is to test if the model and solver can solve line blockage problems. Also, two extra questions need to be answered: how a longer turnaround time affects solutions and how the maximum allowed delay time affects solutions. As shown in Figure 15, most trains run on the line with a fixed one-hour period between them. However, extra trains run on the line during morning and afternoon peak times. Thus, the same delay time and turnaround setting may affect trains differently at different times in a day. During busy times, more trains may need to be delayed or cancelled for the same delay time or turnaround time.

14 blockage (from 6 am to 7 pm) scenarios on the line between S1 and S4 are tested. The blockage site is at a station between S1 and S2. Trains approaching from both sides

short turn before approaching the blockage site. For each test, the blockage period is set as 3 hours, and the recovery period is set as 5 hours. For each scenario, a model is built as in Section 4.1.4. The results of 14 scenarios are shown in Tables 10 and 11 with the different turnaround time settings. A blockage id identifies each scenario. NT stands for the number of trains considered in the model. TC stands for the number of cancelled trains, and TD stands for the number of delayed trains. The experimental tests are conducted in two categories: In one category, the maximum delay t_{delay} is set as 10, and 20 in another.

Experimental Tests 1

Table 10 Timetable rescheduling with the turnaround time (5 minutes) at blockage sites

Scenarios			$t_{delay} = 10$			$t_{delay} = 20$		
Id	Blockage time	NT	TC	TD	Objective	TC	TD	Objective
AA_06	06:00-09:00	76	10	0	10400	10	0	10400
AA_07	07:00-10:00	73	14	0	14400	12	2	12490
AA_08	08:00-11:00	78	18	0	18450	16	4	16584
AA_09	09:00-12:00	78	18	0	18450	16	4	16578
AA_10	10:00-13:00	78	18	0	18450	16	4	16592

AA_11	11:00-14:00	78	18	0	18450	16	4	16578
AA_12	12:00-15:00	78	18	0	18450	16	4	16594
AA_13	13:00-16:00	82	18	0	18450	16	4	16578
AA_14	14:00-17:00	81	22	1	22606	20	5	20784
AA_15	15:00-20:00	88	20	2	20710	18	6	18872
AA_16	16:00-19:00	88	15	5	15638	15	5	15638
AA_17	17:00-22:00	80	17	1	17726	17	1	17726
AA_18	18:00-21:00	76	18	1	18454	16	5	16570
AA_19	19:00-22:00	64	12	1	12514	12	1	12514

The results for timetable rescheduling with a smaller turnaround time (5 minutes) at a turnaround station are shown in Table 10. Each scenario is solved within 2 seconds. Table 10 shows that 64 to 88 trains are considered in the problem scenarios. Overall, the 14 scenarios can be further grouped into (AA_06 - AA_07), (AA_08 - AA_12), (AA_13 - AA_17) and (AA_18 - AA_19) based on the number of trains involved in each group. Fewer trains are considered in the early morning (76 trains for scenario AA_06 and 73 trains for scenario AA_07). The number of trains considered in models for AA_08 -

AA_12 is constant at 78. The number of trains considered in models for AA_13 - AA_17 is higher (80-88). For the last two scenarios (AA_18 - AA_19), fewer trains are considered in the models (76,64).

When t_{delay} is set as both 10 or 20, the maximum and minimum values of the objective are obtained for the scenarios starting at 14:00 and 06:00, respectively. When $t_{delay} = 10$, the maximum and minimum objectives are 22606 and 10400, respectively. When $t_{delay} = 20$, the maximum and minimum values of the objective are 20784 and 10400, respectively. When $t_{delay} = 10$, the scenario starting at 16:00 has the most significant number of delayed trains, 5. When $t_{delay} = 20$, the scenario starting at 15:00 has the most significant number of delayed trains, 6. Compared to the maximum delay set as 10, more trains are delayed in some scenarios (AA_07 - AA_15 and AA_18) when the maximum delay is set as 20. In exchange, fewer trains are cancelled. Solutions have lower or equal cost when the maximum delay is 20.

For the blockage starting at 06:00, no trains are required to be delayed, and 10 trains are cancelled when $t_{delay} = 10$ or 20. For the blockage scenario starting at 07:00, 14 trains are cancelled when $t_{delay} = 10$ while 12 trains are cancelled, and two trains are delayed when $t_{delay} = 20$.

For the blockage from 08:00 to 13:00, 18 trains must be cancelled, and no train is delayed when $t_{delay} = 10$. Fewer trains (16) are cancelled, and more trains (4) are delayed when $t_{delay} = 10$.

For the blockages starting in the afternoon (14:00 - 17:00), between 15 and 22 trains are cancelled, and between 1 and 5 trains are delayed when $t_{delay} = 10$. With $t_{dealy} = 20$, there are between 15 and 20 cancelled trains and between 1 and 6 delayed trains.

For the last two scenarios, when the blockage starts at 18:00, 18 trains are cancelled, and one train is delayed when $t_{delay} = 10$, compared to 16 trains that are cancelled and five trains delayed when $t_{delay} = 20$. When the blockage starts at 19:00, there are 12 train cancellations and one train delay for both $t_{delay} = 10$ or 20.

Experimental Tests 2

Table 11 Timetable rescheduling with the significant turnaround time (15 minutes) at blockage sites

Scenario			$t_{delay} = 10$			$t_{delay} = 20$		
Id	Blockage time	NT	TC	TD	Objective	TC	TD	Objective
AA_06	06:00-09:00	76	10	2	10420	10	2	10420
AA_07	07:00-10:00	73	14	2	14420	14	2	14420

AA_08	08:00-11:00	78	18	0	18450	18	0	18450
AA_09	09:00-12:00	78	18	0	18450	18	0	18450
AA_10	10:00-13:00	78	18	0	18450	18	0	18450
AA_11	11:00-14:00	78	18	0	18450	18	0	18450
AA_12	12:00-15:00	78	18	0	18450	18	0	18450
AA_13	13:00-16:00	82	18	0	18450	18	0	18450
AA_14	14:00-17:00	81	22	2	22678	22	3	22638
AA_15	15:00-20:00	88	20	4	20794	20	4	20794
AA_16	16:00-19:00	88	19	4	19532	17	6	17616
AA_17	17:00-22:00	80	20	1	20618	18	5	18780
AA_18	18:00-21:00	76	18	1	18454	18	1	18454
AA_19	19:00-22:00	64	12	1	12514	12	1	12514

When a longer turnaround time is required at a turnaround station (15 minutes), the rescheduling results are shown in Table 11. Compared to the results obtained in Table 10 with a lower turnaround time (5 minutes), more trains are required to be cancelled, no matter if the maximum delay is set as 10 or 20 minutes. Also, the solution costs are higher than in the scenarios solved with a shorter turnaround time. A more considerable delay time can decrease the number of cancelled trains and improve the solution cost. However, train cancellation may be opted for if the delay causes the next train to catch up.

In conclusion, a longer maximum delay can decrease the number of cancelled trains and lead to better solutions. There is a trade-off between delays and cancellations. Some trains can be delayed rather than cancelled when the maximum delay time is extended. Also, a longer turnaround time at the turnaround stations can cause more trains to be cancelled and delayed, leading to worse results. Since the model is guaranteed to return to normal operations after rescheduling time, delays and cancellations will not affect trains outside the recovery period.

4.1.7. Conclusion

In this section, a model is proposed for the timetable rescheduling problem in case of a complete blockage. It gives a revised timetable which covers the three phases of a train service recovery process: the transition phase, degraded operation phase and recovery phase. The railway network is not assumed empty when rescheduling starts. The railway services are planned to return to their normal level when rescheduling ends. The model uses rescheduling strategies of cancellation and retiming and considers constraints, including infrastructure, rolling stock availability, and rolling stock balance. It minimises train cancellations, delays and changes to rolling stock circulation patterns. A solution that requires changes to the current rolling stock circulation patterns without decreasing cancellations and delays is not preferred. It has been successfully tested on 14 scenarios of a complete blockage on a double-tracked line using actual data from a TOC in Great Britain. Timetable rescheduling is the first step in railway disruption management. In Section 4.4, the crew rescheduling, is explained.

4.2. Optimisation Problems

In this section, some basic mathematical programming concepts are presented, which will be used in the remainder of the thesis.

4.2.1. Linear Programming, Duality and Reduced Cost

Linear programming is to optimise objective function linear in terms of decision variables while a set of linear constraints and sign restrictions are imposed on these decision variables, see (Kantorovich, 1939), (Fang & Puthenpura., 1993) and (Dantzig & Thapa., 2003). Consider a linear programming problem in standard form:

$$min\, c^T x \qquad\qquad LP$$

$$Ax = b$$

$$x \geq 0$$

Here $A \in R^{m\times n}, c, x \in R^n$ and $b \in R^m$

The dual of the above LP is:

$$max\, b^T \pi \qquad\qquad DP$$

$$A^T\pi \leq c, \;\; where\; \pi \in R^m$$

Theorem 1 Strong Duality: If the LP problem has an optimal solution, then so does its dual and their optimal values are equal.

The reduced cost vector is defined as $c - A^T\pi$, with each element corresponding to a variable x. For a minimisation problem, a negative reduced cost associated with the corresponding variable component x_i can be used to construct a better solution for the LP problem.

4.2.2. Integer Programming and Lagrangian Relaxation

A standard integer linear problem (ILP) is stated as follows.

$$\min_x cx \qquad\qquad (ILP)$$

$$Ax \geq b$$

$$Bx = d$$

$$x \geq 0$$

$$x_i\; integer\;\; for\;\; i \in I$$

Where b, c and d are vectors, A and B are matrices of conformable dimensions and the index set I denotes the variables required to be an integer. The theory of Lagrangian relaxation for integer programming is explored in (Geoffrion, 1974).

Lagrangian relaxation is a technique to relax some constraints in an optimisation problem. The idea is to remove the constraint and add a penalty in the objective function to penalise any solution that violates the constraint. Thus, a Lagrangian Relaxation Problem (LRP) can be obtained by bringing the explicit constraints into the objective function multiplied by the associated Lagrangian multiplier vector λ. For example, a LRP for a given λ can be formulated as follows:

$$L(\lambda) = \min_{x} c(x) + \lambda(b - Ax) \qquad (LRP)$$

$$s.t. Bx = d$$

$$x \geq 0$$

$$x_j \; integer, j \in I$$

For any given $\lambda \geq 0$, $L(\lambda)$ is the lower bound of the optimal value of the ILP.

Theorem 2 Let $v(LRP)$ be the optimal value of LRP and $v(ILP)$ is the optimal value of ILP, $v(LRP) \leq v(ILP)$.

Proof:

Assuming x^* is an optimal solution to the Problem (ILP) and $\lambda \geq 0$, then $\lambda(b - Ax^*) \leq 0$

$$v(LRP) \leq cx^* + \lambda(b - Ax^*) \leq v(ILP)$$

Also, since the constraints in the ILP are taken into the objective in Lagrangian relaxation, the feasible set of the ILP is a subset of the feasible set of the Lagrangian relaxation. Thus, for the ILP, the optimal value of the corresponding Lagrangian relaxation (LRP) is lower than or equal to the optimal value of ILP.

Lagrangian dual problem

Since $L(\lambda)$ can provide a lower bound for a given ILP, it is more interesting to obtain the best lower bound. Taking the maximum of the Lagrangian relaxation gives the following Lagrangian dual problem (LDP):

$$max\{L(\lambda)\} = \max_{\lambda}\{\min_{x}\{(cx + \lambda(b - Ax) : Bx = d, x \geq 0, x_i\ integer, i \in I\}\}$$

To solve the Lagrangian dual problem, the subgradient method is used. Lagrangian and linear relaxation can provide a lower bound to the ILP problem; the relation between these lower bounds in Theorem 3 can be found in (Geoffrion, 1974). A linear relaxation of the integer program is the problem with the same objective and the same constraints except for the integrality restrictions, which are removed.

Theorem 3 Let ILP be an integer linear program. Then the optimal value achieved by the Lagrangian relaxation of ILP is greater than or equal to the optimal value achieved by the linear relaxation of ILP.

4.2.3. Set Covering Problem

The minimum set covering problem can be formulated as the following integer linear program.

$$min \sum_{j=1}^{n} x_j$$

$$Ax \geq 1$$

$$x \in \{0,1\}^n$$

Here A is a $m \times n$ matrix, $a_{ij} \in \{0,1\}$. The set covering problem is NP-hard of combinatorial optimisation. Given a collection of elements, the set covering problem aims to find the minimum number of sets that incorporate (cover) all of these elements (Grossman & Wool, 1997). Many problems in industry can be formulated as set covering problems, for example, job-machine problems, scheduling problems, and picking up the best locations to cover the maximum number of customers. Crew rescheduling can be formulated using a variety of set covering problems.

4.3. Optimisation Techniques

The following mathematical techniques are used in the algorithm to solve the crew rescheduling problem for significant disruptions. Before presenting the algorithm, a general description of these mathematical techniques is provided.

4.3.1. Column Generation

In large linear programs with many more variables than the number of constraints, it is usually impossible to consider all the variables explicitly. Also, only a small subset of variables will likely be used in the optimal solution, and most of the variables will be assigned to 0. Based on this idea, column generation was first proposed by (Ford Jr & Fulkerson, 1958).

The linear program is formulated as two parts, restricted master problem (RMP) and subproblem. RMP has a subset of variables, and its objective function is the same as that of the linear problem, where only these variables are nonzero. The subproblem is to find a new variable of negative reduced cost for minimisation problems using the current dual values that are generated from the RMP and bring it to the variable pool of RMP. Usually, the subproblem is referred to as a pricing problem. A general column generation scheme can be seen in Algorithm 3.

```
Algorithm 3 A general column generation scheme
1: The RMP problem is solved based on a small set of variables.
2: The dual value for each constraint is obtained for the RMP.
3: The dual values are used in the subproblem to find new variables with a
   negative reduced cost.
4: If new variables are generated, new variables are added to the RMP problem,
   go back to step 1. If no new variable is generated, the solution to the current
   RMP is optimal.
```

Algorithm 3 Column generation scheme

Various column generation strategies can be designed to improve speed. In every iteration of the subproblem, more than one new variable with a negative reduced cost can be added to the RMP. It will increase the time required for one iteration since RMP needs to solve a bigger scale problem, but it may decrease the number of iterations. The reader is referred to (Vanderbeck, 1994) for the ways to choose a ``good'' subset of columns.

4.3.2. Subgradient Method

For solving the crew rescheduling problem (see Section 4.4), the Lagrangian dual problem is created for the crew rescheduling model. The subgradient method is commonly used to solve a Lagrangian dual problem. The subgradient method is an iterative algorithm for obtaining the optimal value of non-differentiable convex minimisation problems. Shor first developed it in his book (Shor, 1985). Compared to the gradient used in the steepest descent method, the subgradient does not require a function to be differentiable, so it always exists for a convex or concave function.

Definition A function $f: D \to \mathcal{R}$ is convex if its domain D is a convex set and for all x, y in its domain, and all $\lambda \in [0,1]$, it satisfies $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$. A function f is concave if $-f$ is convex.

Consider a problem for convex f:

$$min\, f(x)$$

To minimise f, the subgradient method uses the iteration

$$x^{k+1} = x^k - \Delta_k g^k,$$

where x^k is the value of x at kth iteration. The variable g^k is a subgradient vector and Δ_k is the step size at the kth iteration. Since the subgradient method does not have the descent property of the gradient method, it is necessary to track the best result f_{best}^k so far in every iteration k, $f_{best}^k = min\{f_{best}^{k-1}, f(x^k)\}$ for all k.

4.3.3. Shortest Path Problems with Resource Constraints

The shortest path problem with resource constraints (SPPRC) is widely used as a subproblem to contribute to the success of the column generation method. It constitutes a flexible tool to model complex cost structures and a wide variety of rules that define the feasibility of a path. It was introduced by (Desrochers, 1986) in his PhD thesis as a subproblem of a bus driver scheduling problem. Solving SPPRC means finding the shortest path among all feasible paths that start from a source node and end at a sink node. Several varieties of SPPRC are studied in literature and can be classified by resource accumulation ways, additional path structural constraints, objective and underlying network. Readers are referred to (Irnich & Desaulniers, 2005) to read the detailed descriptions of various SPPRC problems.

In this work, SPPRC is considered on an acyclic graph where paths are all elementary paths, in which all nodes are pairwise different. A digraph $G = (V, A)$ is defined, where V and A are the set of nodes and arcs, respectively. A path $P = (v_0, v_1, \dots v_p)$ is a finite sequence of nodes where each node pair $(v_i, v_{i+1}), i = 0, \dots, p-1$ is connected by an arc $a \in A$. $v(P) = v_p$ is the end node of the path P. A resource extension function is defined for every arc in a graph and every resource. Let k be the number of resources; a vector $T_i = \left(T_i^1, \dots, T_i^k\right) \in R^k$ is called the resource vector corresponding to the node i. A non-decreasing resource extension function (REF) is of the form: $f_{ii+1}^k = f_{i-1i}^k + t_{ii+1}^k$. A path $P = (v_0, v_1, \dots v_p)$ is feasible if there exist T_i for all positions $i = 0, \dots, p-1$ such that $f_{v_i, v_{i+1}} \leq T_{i+1}$.

Finding a recovery diagram is solved as SPPRC in the crew rescheduling problem. The application of SPPRC in solving the crew rescheduling problem is explained in Section 4.4.3.

4.4. Crew Rescheduling Problem

Crew rescheduling is the next important step in railway disruption management, following timetable and rolling stock rescheduling. As the introduction (Chapter 1) explains, a driver's daily work is regulated as a diagram. When a disruption happens, some drivers' diagrams become infeasible, which means the drivers cannot follow them. Thus, each driver whose diagram is affected by a disruption should be assigned a recovery diagram. The definitions for a commencing activity and a terminating activity used in DFSCR for minor disruptions are given in Section 3.1.2. Here the definitions for these two terms are modified for crew rescheduling in significant disruptions.

A commencing activity is set as the activity that a driver is performing or just finished performing when rescheduling starts. A terminating activity is set as the activity that a driver will perform when rescheduling ends. Between the commencing and terminating activities, a recovery diagram has different tasks than the planned diagram. However, a recovery diagram has the same activities from the sign-on to the commencing activity as a planned diagram. The reason is that when a disruption happens, these activities already happened. A recovery diagram has the same activities from terminating activity to the sign-off as in the planned diagram because a recovery diagram is supposed to be recovered back to the scheduled diagram. The main goal of the crew rescheduling problem is to reschedule crew to cover the train services in a revised timetable as much as possible with changes to planned crew diagrams as little as possible.

The work in this section is based on (Potthoff, 2010) following the idea of applying Lagrangian relaxation to the same crew rescheduling problem and using a greedy algorithm to solve a crew rescheduling model. In (Potthoff, 2010)'s work, it is not clear how a recovery diagram is found by solving SPPRC to optimality since SPPRC is an NP-hard problem and it is not explained in (Potthoff, 2010) how they solve it. In this section, we give a general scheme to solve SPPRC. However, in some of the scenarios we tested, solving SPPRC to optimality using this general scheme takes too long time (more than 500 seconds), which makes the method not suitable to be used in real-time. Then we proposed three search methods to speed up the process. Later, how these three search methods affect solutions is compared and analysed. A disruption neighbourhood is proposed to limit the problem size. The method for building an initial disruption neighbourhood and expanding a disruption neighbourhood is described. Also, (Potthoff, 2010) did not elaborate on various connection types and described only the basic ones. This section gives detailed algorithms to construct different connection types between two tasks. These connection types and their costs are essential in building recovery diagrams that are very close to the diagrams used in practice and calculating recovery diagram costs. Such recovery diagrams are easier to be accepted by drivers.

The model for the crew rescheduling problem for significant disruption is first given in Section 4.4.1. Then how to construct the graph to show the tasks considered in a crew rescheduling model and their connections are explained in Section 4.4.2. A recovery diagram can be found as SPPRC using the graph constructed, which is explained in Section 4.4.3. Using this basic knowledge, the method LRCG for solving the crew rescheduling model is given in Section 4.4.4. Then a didactic example is given to demonstrate the model and method in Section 4.4.5. Several examples using actual data are presented and analysed in Sections 4.4.6 and 4.4.7.

4.4.1. Problem Description and Model

The crew rescheduling problem (CRP) can be mathematically modelled as follows. An idea, the disruption neighbourhood is introduced to help build the model for CRP. A disruption neighbourhood is used to select drivers that should be considered in the rescheduling problem. How to construct a disruption neighbourhood is explained in Section 4.4.5. For now, let D be the set of all drivers in a disruption neighbourhood. For each driver $d \in D$, a recovery diagram r should be generated and assigned to this driver. The set of recovery diagrams for the driver d is denoted by R^d. If the recovery diagram r of the driver d covers a driving task t, then a_{tr}^d is 1, 0 otherwise. Binary variables x_r^d and f_t are introduced. x_r^d equals 1 if recovery diagram r is chosen for the driver d, 0 otherwise. f_t equals 1 if driving task t is not covered, 0 otherwise. The crew rescheduling problem can be modelled as below:

$$min \sum_{d \in D} \sum_{r \in R^d} c_r^d x_r^d + \sum_{t \in T} p f_t \qquad \text{CRP}$$

$$\sum_{d \in D} \sum_{r \in R^d} a_{tr}^d \, x_r^d + f_t \geq 1, \forall t \in T$$

$$\sum_{r \in R^d} x_r^d = 1, \forall d \in D$$

$$x_r^d \in \{0,1\}\ \forall d \in D, \forall r \in R^d\ and\ f_t \in \{0,1\}\ \forall t \in T$$

Let c_r^d be the cost of a recovery diagram and p be the penalty for task cancellation. The objective of CRP is to minimise the cost of the selected recovery diagrams and task cancellations. The first type of constraint means that each task should be covered at least by a recovery diagram or cancelled. The second type of constraint means that each driver should be assigned a recovery diagram.

4.4.2. The Graph used to find Recovery Diagrams

To build a model for CRP, recovery diagrams should be generated. Recovery diagrams can be generated by solving SPPRC. This section introduces the graph used to solve SPPRC.

A directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ is built to find recovery diagrams of all drivers. $\mathcal{N}$ denotes the nodes representing all driving tasks, commencing activity and terminating activity for each driver in a disruption neighbourhood. The commencing or terminating activity represents a driver's activity when rescheduling starts or ends, respectively. The node representing a commencing or terminating activity is called the source node or sink node.

The set of arcs $\mathcal{A}$ satisfies the following conditions. Two nodes are connected with one arc at most, and each arc has its direction. If there is an arc from $a \in \mathcal{N}$ to $b \in \mathcal{N}$, then there is no arc from b to a. Each arc has its type and weight. It is present in the graph if a driver can consecutively cover the corresponding two driving tasks. Table 12 shows all the arc types used in the work. Arc weights will be used to calculate the cost of a recovery diagram. Besides these costs, if a driver takes up a new task that is not in their planned diagrams, an extra cost of 300 is added. If this driver is spare, an additional cost of 80 is added.

Table 12 Connection types used to construct a recovery diagram

No	Arc name	Connection type	Weight	Description
1	immArc	immediate	0	immediate connection between two tasks on one train
2	changeArc	change	10	change trains
3	mealArc	meal	5	meal break
4	passArc	passenger trip	20	take a passenger trip
5	mealPassArc	meal passenger trip	15	take a passenger trip and meal break
6	douPassArc	double passenger trips	30	take double passenger trips
7	mealDouPassArc	meal double passenger trips	25	take double passenger trips and a meal break
8	taxiArc	taxi	50	take a taxi trip

9	taxiOverTimeArc	taxi overtime	60	take a taxi trip to a late sign off
10	taxiMealArc	taxi meal	40	take a taxi trip and a meal break
11	overTimeArc	overtime	40	late sign off
12	oldArc	original connection	0	same connect as in planned diagrams

Determining Arc Type

Twelve different arc types are used in the graph built to produce recovery diagrams. The geographical relationship between two tasks is considered first to connect two driving tasks. If the previous task arrives at the same station that the next task departs, there are three types of arcs to consider, *mealArc*, *changeArc* and *immArc*.

A passenger or taxi trip is considered if a previous task arrives at a different station than the next task departs. Thus, these two types *passArc* and *mealPassArc* can be used. At most, two passenger trips are considered to connect two tasks since building this graph is computationally time-consuming. Then there are two types *douPassArc* and *mealDouPassArc*. There could be more than one possible passenger trip between two tasks to connect them. Taxi is also considered. There are types *taxiArc* and *taxiMealArc*. When drivers work overtime, there are types *overtimeArc* and *taxiOvertimeArc*. To reduce the deviations from the planned diagrams, if two tasks can be connected as in the initial diagram, *oldArc* is used. Given two tasks, the arc type between them can be decided with the following algorithms (Algorithm 4 - Algorithm 6).

It is important to distinguish if the next task to be connected is “sign off” because overtime may be required. Algorithm 4 explains how to decide the arc type between task1 and task2 if task2 is not signed off. At first, the connection type, oldArc*,* between two tasks is considered if the two tasks can be connected as in the planned data. If task1 ends at the station where task2 starts, then mealArc is chosen in step 8 if the time difference between the two tasks is long enough to have a meal break. Otherwise, changeArc or immArc is considered in steps 11 and 13. If task1 ends at a different station where task2 starts, Algorithm 5 is used. In this case, a driver must be transferred from one station to another to perform task2 after task1.

Algorithm 4 Determine the type of arc between task1 and task2 if task2 is not "sign off"

```
Input: task1,task2
 1: if task1 and task2 are carried out by the same driver in planned data then
 2:     if task1 and task2 can still be connected as in this driver's planned dia-
        gram then
 3:         return oldArc
 4:     end if
 5: end if
 6: if task1 arrives at the station where task2 starts then
 7:     if the time difference between two tasks is enough for a meal then
 8:         return mealArc
 9:     end if
10:     if task1 and task2 are on different trains then
11:         return changeArc
12:     end if
13:     return immArc
14: else
        determine the type of arc between task1 and task2 requiring transferring
        driver (Algorithm 5)
15: end if
Output: arc type
```

Algorithm 4 Determining the arc type between task1 and task2.

For transferring drivers, Algorithm 5 is used. A passenger trip is first considered if the driver can board a train to relocate in step 1. If such a passenger trip exists, a meal break opportunity is further considered in step 2. If there is enough time for a meal break, mealPassArc is chosen in step 3. Otherwise, passArc is selected in step 5. At most, two passenger trips can be used. If no passenger trip is possible, using a taxi to relocate drivers is considered in step 13.

```
Algorithm 5 Determine the type of arc between task1 and task2 requiring transferring driver
Input: task1,task2
 1: if task1 and task2 can be connected with a task_pass (task_pass stands for
    a passenger trip) then
 2:   if time difference between task1 and task_pass or task_pass and task2 is
      enough for a meal break then
 3:     return mealPassArc
 4:   end if
 5:   return passArc
 6: end if
 7: if task1 and task2 can be connected with a task_pass1 and task_pass2
    (task_pass1 and task_pass2 stand for two passenger trips) then
 8:   if time difference between task1 and task_pass1 or task_pass1 and
      task_pass2 or task_pass2 and task2 is enough for a meal break then
 9:     return mealDouPassArc
10:   end if
11:   return douPassArc
12: end if
13: if time difference is enough for a taxi trip and a meal then
14:   return taxiMealArc
15: end if
16: if time difference is enough for a taxi trip then
17:   return taxiArc
18: end if
Output: arc type
```

Algorithm 5 Determining the arc type requiring transferring a driver

Algorithm 6 explains how to determine an arc type between task1 and task2 if task2 is sign off. As in Algorithm 4, oldArc, mealArc and immArc are considered first in steps 2,7 and 9. If a driver needs overtime to cover task1, overTimeArc is considered in step 11. A passenger trip required to relocate the driver to sign off is called in step 14. If a driver is required to work overtime and must be relocated, taxiOvertimeArc is used.

Algorithm 6 Determine the type of arc between task1 and task2 when task2 is "sign off"

```
Input: task1,task2
 1: if old arc exists between task1 and task2 based on step 1-4 in Algorithm 4
    then
 2:   return oldArc
 3: end if
 4: if task1 arrives at the station where task2 starts then
 5:   if task1 ends before task2 starts then
 6:     if time difference between task1 and task2 is enough for a meal break
        then
 7:       return mealArc
 8:     end if
 9:     return immArc
10:   else if task1 ends before task2 starts plus a maximum work overtime
      then
11:     return overtimeArc
12:   end if
13: else
14:   if passenger trips exist based on rules in step 1-12 in Algorithm 5 then
15:     return the arc type between task1 and task2 requiring transferring
        driver based on step 1-12
16:   end if
17:   newSignOffTime=task1 ending time+taxi duration
18:   if newSignOffTime is before task2 happens then
19:     return taxiArc
20:   else if newSignOffTime ends before task2 happens plus a maximum work
      overtime then
21:     return taxiOvertimeArc
22:   end if
23: end if
Output: arc type
```

Algorithm 6 Determining the arc type between task1 and sign-off

Creating a Subgraph for a Driver

Since different drivers can share parts of the graph, an integrated graph $\mathcal{G}$ is built with all tasks for all drivers in a disruption neighbourhood. A subgraph is then extracted for each driver instead of building a graph for each driver.

The integrated graph is stored as a node-list structure. To get a subgraph for a given driver, a depth-first search is used to find the set $\mathcal{N}'$ of all nodes that this driver can cover with a given source node and sink node. Then, the nodes in $\mathcal{N}'$ and lists of arcs that start at these nodes form a subgraph for this driver, denoted by $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$, which is used to find recovery diagrams just for the driver.

4.4.3. Solving SPPRC

SPPRC is NP-hard in the strong sense, but efficient algorithms exist for solving some crucial variants of SPPRC (Irnich & Desaulniers, 2005). Dynamic programming combined with the labelling algorithm is one of the most widely used techniques. Some SPPRC, for example, constrained shortest path problem (CSPP), can be solved with Lagrangian Relaxation. Constraint Programming solves SPPRC using a broad spectrum of constraints, like path structural constraints. Also, some heuristics algorithms like direct search verify that no negative reduced cost paths exist in the pricing step in a column generation approach. Dynamic programming combined with the labelling algorithm is used in this work to generate recovery diagrams for drivers. Here this method is explained in more detail.

Starting from source node s, the dynamic programming approach for the SPPRC extends the path $P = \{s\}$ one by one into all possible directions. For efficiency, paths in the dynamic programming approach are encoded by labels. A label for a path $P = (s, v_1, v_2, \dots, v_p)$ is directly linked to the label of the prefix path $(s, v_1, v_2, \dots, v_{p-1})$. Paths with the same prefix path would have the same chain of labels for their common paths. Besides the nodes visited by a path, a label for a path also stores the resource vector of this path, $f(P)$, which is calculated using the resource extension function on the nodes visited.

UP is the set of unprocessed paths to be extended, PP is the set of processed paths. A is the graph built for SPPRC. In the labelling algorithm, one unprocessed path from UP is chosen and all feasible extensions (Q, v) with $v \in V$ are constructed and added to UP, while Q is removed from UP and added to PP.

Algorithm 7 solve SPPRC

```
1: initialise: $p_0 = (s)$, $UP = \{p_0\}$, $PP = \{\}$
2: Path extension step
3: while UP != empty do
4:     choose a path Q from UP and remove Q from UP
5:     for each arc (v(Q),w) $\in$ A do
6:         if the extension path (Q,w) is feasible then
7:             add (Q,w) to UP
8:         end if
9:     end for
10:    add Q to PP
11: end while
```

Algorithm 7 Solving SPPRC

Solving SPPRC using Forwards, Backwards and Bi-directional Search

In Algorithm 7, the starting point to construct a path is the source node, which is extended using dynamic programming in the direction of the source node to the sink node. This search method is called forwards search. The algorithm can be slightly modified to construct paths with two different search methods: backwards search and bi-directional search.

For backwards search, in step 1 in Algorithm 7, the path $p0$ is initiated with a sink node, and the path is extended in the direction of the sink node to the source in step 5. For bi-directional search, one path $p0_{forwards}$ is initiated with the source node and stored in $UP_{forwards}$, and another $p0_{backwards}$ is initiated with the sink node and stored in $UP_{backwards}$. A random number is used in step 4 to decide the search direction, forwards or backwards. Once the search direction is determined, a path will be chosen from a corresponding set and extended accordingly.

Recall that a source node represents a commencing activity for a driver, and a sink node represents a terminating activity for a driver. Thus, the forwards search builds all recovery diagrams from a commencing activity to a terminating activity. The backwards search creates recovery diagrams from a terminating activity to a commencing activity. The bi-directional search produces some recovery diagrams from a commencing activity to a terminating activity and others from the opposite direction. Using different search methods in solving SPPRC can cause different solution behaviours. (Righini & Salani,

2006) illustrated how basic dynamic programming can be enhanced by using the bi-directional technique for solving the capacitated vehicle routing problem. Later in Section 4.4.7, the effect of using three search methods on obtaining crew rescheduling solutions are compared and analysed.

Example

Due to the safety working regulations, there is a limit for drivers' continuous working without a break called the maximum continuous working time without a break. Here is an example to show how SPPRC can model the meal break rules in the crew rescheduling problem.

Figure 16 shows an example that involves break opportunities and considers one resource constraint: continuous working time without breaks. The source and sink nodes are denoted by s and t, respectively. The source node and sink node represent the current activity a driver needs to perform when rescheduling starts and ends. Nodes 1,2, and 3 represent three different driving tasks. Parameter t_{diff} is defined between two nodes as the time difference between the arrival time of a previous task and the departure time of the next task. Each arc (i, j) has one attribute and one resource of using the arc. The attribute is the arc type $(type_{ij})$, which shows the nature of the connection between two nodes. In this example, there are arc types "change" and "break". "change" means that a driver needs to transfer trains between two nodes and t_{diff} is not smaller than 10 minutes for a driver to transfer trains. "break" means that t_{diff} is enough for a driver to have a 40-minute break.

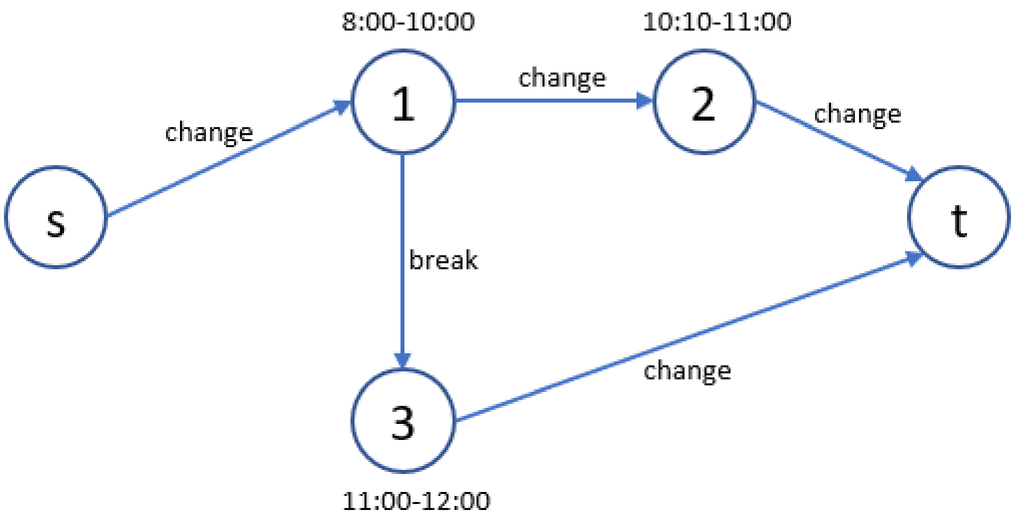


Figure 16 Example of finding feasible recovery diagrams

In the example of Figure 16, given driver d, at the source node, this driver has already worked for 110 minutes. The maximum continuous working time without a break (T_{max}) is 4 hours (240 minutes). The path $P_1 = (s, 1, 2, t)$ is not resource feasible. Indeed, at node 1, driver d has worked 230 minutes (110+120) without a break. From node 1, driver d cannot visit node 2 since the arc does not contain a break opportunity, and at node 2, driver d has worked more than T_{max} without a break. The second path $P_2 = (s, 1, 3, t)$ is feasible because the arc between node 1 and 3 has a break opportunity. Then at the beginning of node 3, the continuous driving time of driver d is 0. Using this graph, only one recovery diagram P_2 can be generated.

4.4.4. The LRCG Method

CRP can be solved as an integer programming problem if all model variables are precisely known, particularly the size of D and R^d for each driver d. The number of recovery diagrams generated for drivers in a model for CRP increases exponentially with respect to the number of tasks considered in the model. Thus, CRP is a large-scale integer programming problem, requiring heuristic rules and mathematical techniques to limit the problem size.

Two techniques are used to limit the problem size: (1) iterative expanding disruption neighbourhood. When a disruption happens, it needs to be clarified when and within which edges of the railway network this problem can be solved. Thus, it is not exactly known how many drivers d are included in the model. Therefore, solving this problem uses an iterative expanding approach. (2) using column generation to generate new variables, representing recovery diagrams gradually. The size of R^d grows exponentially with the increasing size of driving tasks. Meanwhile, generating a recovery diagram is unnecessary if the diagram is no better than the already generated diagrams for a driver. Thus, one needs to dynamically generate recovery diagrams and add them to the CRP model as variables to improve the solution speed.

The LRCG method to solve the CRP model consists of several parts. Figure 17 gives an overview of the whole procedure. As shown in Figure 17, the inputs for the LRCG method include a revised timetable, revised rolling stock diagrams, planned crew diagrams and rescheduling start and end times.

(1) Step 1: in the preprocessing step, an initial disruption neighbourhood Δ is built with Algorithm 8. A subset of recovery diagrams R^d is generated for each driver d in the disruption neighbourhood Δ. It is time-consuming and unnecessary to create all recovery diagrams for each driver. At this step, a certain number of recovery diagrams are first generated for each driver. Later in the algorithm, more promising recovery diagrams are generated and added to the current R^d

(2) Step 2: a CRP model is built with the disruption neighbourhood Δ and recovery diagrams R^d. The CRP model is solved with a greedy algorithm. Lagrangian multipliers are obtained from solving the Lagrangian dual problem

(3) Step 3: a pricing problem is built using Lagrangian multipliers as input to generate promising recovery diagrams. If such recovery diagrams exist, these new recovery diagrams are used to create new variables x_r^d that are added to the current CRP model; go to step 2. Otherwise, proceed to step 4

(4) Step 4: If no new recovery diagram is generated and there is task left uncovered, the current disruption neighbourhood Δ is expanded; go to step 1. Otherwise, proceed to step 5

(5) Step 5: stop. The output of this algorithm is a recovery diagram for each driver in the final disruption neighbourhood and cancelled tasks

Each step of the LRCG method is explained in the following subsections.

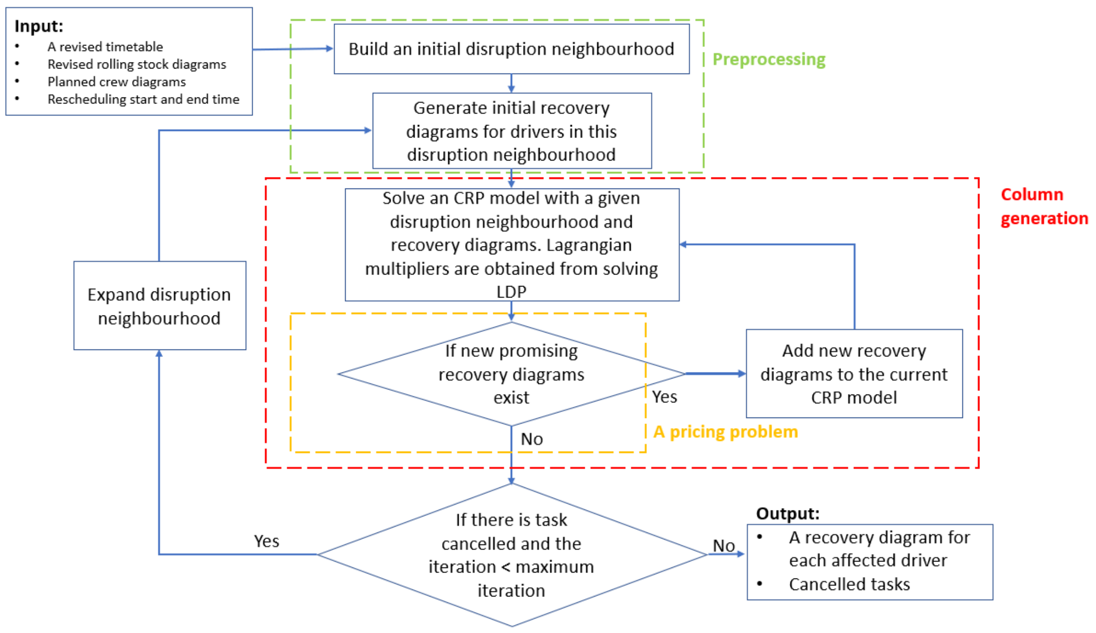

Input:
A revised timetable
Revised rolling stock diagrams
Planned crew diagrams
Rescheduling start and end time
Build an initial disruption neighbourhood
Preprocessing
Generate initial recovery diagrams for drivers in this disruption neighbourhood
Column generation
Solve an CRP model with a given disruption neighbourhood and recovery diagrams. Lagrangian multipliers are obtained from solving LDP
Expand disruption neighbourhood
If new promising recovery diagrams exist
Yes
Add new recovery diagrams to the current CRP model
A pricing problem
No
If there is task cancelled and the iteration < maximum iteration
Yes
No
Output:
A recovery diagram for each affected driver
Cancelled tasks

Figure 17 An overview of the LRCG method solving the CRP model

Initial and Expanded Disruption Neighbourhood

A disruption neighbourhood is introduced to limit the model size. A disruption neighbourhood is characterised by a recovery period, from the rescheduling start time to the end time. This recovery period is when a disruption occurs to when a regular timetable can be restored. A disruption neighbourhood contains drivers and the driving tasks of these drivers during the recovery period.

In LRCG, disruption neighbourhoods are automatically expanded. In the ith disruption neighbourhood Δ_i, if the solution shows that some driving tasks are not covered, the disruption neighbourhood is expanded to Δ_{i+1}.

Figure 18 illustrates building an initial disruption neighbourhood Δ_1 starting with directly affected trains. Assuming train $V1$ is a directly affected train, $D1$ is the driver who is assigned to this train's first trip or takes this trip as a passenger. Then $D1$ needs to be added to Δ_1 since $D1$ is a directly affected driver. $D1$ also covers one trip from a train $V2$ which runs during the recovery period. Thus, $V2$ is a potentially affected train. Since driver $D2$ covers one trip or takes a passenger ride of $V2$, driver D2 is also added to the disruption neighbourhood Δ_1. The above process repeats until no new drivers can be found.

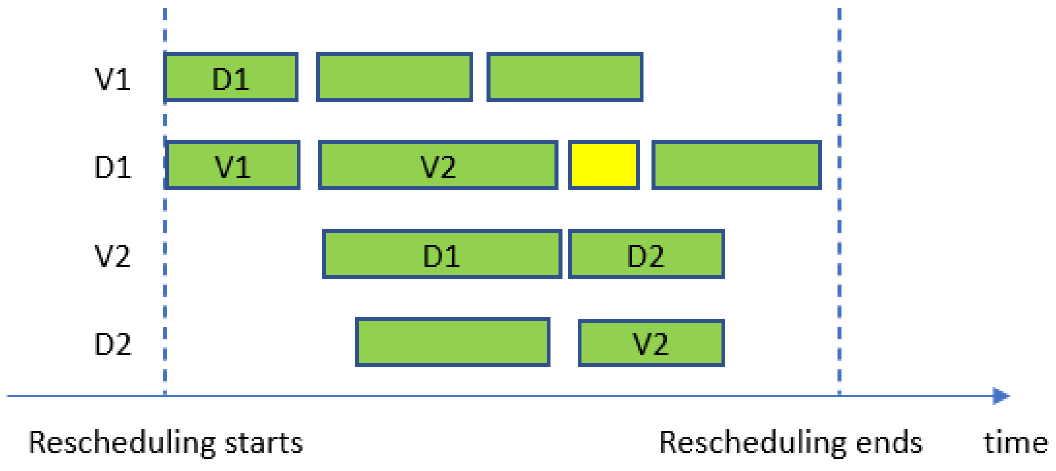


Figure 18 Process of building an initial disruption neighbourhood

In Figure 18, V1 and V2 denote two different train services. D1 and D2 represent two different drivers. Each rectangle in the rows of V1 (V2) represents a trip in the train service. Each rectangle in the rows of D1 (D2) represents an activity in the driver diagram. A green rectangle on a driver’s diagram means a driving task which is a trip assigned to the driver. A yellow rectangle represents a meal break for the driver.

The above process of building an initial disruption neighbourhood is shown in Algorithm 8. The input for this algorithm is the rescheduling start and end times, planned crew diagrams and a revised timetable.

```
Algorithm 8 Building an initial disruption neighbourhood $\Delta_1$
Input: $t_{schedule\_start}$, $t_{schedule\_end}$, planned crew diagrams, revised timetable
Initialise: $Drivers\_In\_DisNhood = \emptyset$, $Tasks\_In\_DisNhood = \emptyset$

1: find directly affected trains $Affect\_V$ due to blockage
2: while new affected trains found do
3:    find drivers with planned crew diagrams who are planned to cover affected
      trains or take passenger trips on affected trains during rescheduling period
      $t_{schedule\_start}$ to $t_{schedule\_end}$, add to $Drivers\_In\_DisNhood$
4:    find new affected trains from a revised timetable that are planned
      to be covered by drivers found in Step 3 during rescheduling period
      $t_{schedule\_start}$ to $t_{schedule\_end}$
5: end while
6: collect all tasks of drivers in $Drivers\_In\_DisNhood$ during the rescheduling
   period, form $Tasks\_In\_DisNhood$
Output: $\Delta_1$ = {Drivers_In_DisNhood,Tasks_In_DisNhood}
```

Algorithm 8 Building an initial disruption neighbourhood

Expanded Disruption Neighbourhood

To construct an expanded disruption neighbourhood, two kinds of drivers are added. One is diagrammed drivers (drivers are assigned a diagram in the planned data), and the other is spare drivers (drivers at work but with no activities originally planned on the day). Before a disruption neighbourhood needs to be expanded, a solution to the CRP model has already been obtained, as described in Figure 17. With the solution, a list of uncovered tasks is also obtained.

Two kinds of diagrammed drivers can be found as follows. The first kind is the drivers who are near the uncovered task. For each uncovered task, *around tasks* are first defined. A task is called an around task with respect to an uncovered task if one of the following two conditions is satisfied: (1) it departs from the origin of the uncovered task within a specific time (like two hours) of the uncovered task departure (2) it departs from the destination of the uncovered task within a specific time after the uncovered task arrival. A driver whose diagram contains this task is added to the expanded disruption neighbourhood for each around task.

The second type of diagrammed drivers are similar to those who plan to cover the uncovered task. The reason to find such drivers is that they have a high chance of swapping tasks with each other. A similarity score is calculated for each diagrammed driver, and drivers with high scores will be added to the new disruption neighbourhood. If two drivers are from the same depot, 1 is added to the similarity score. If they are at the same station when the rescheduling starts, 5 is added to the similarity score. For each task they cover, if they depart from the same station within 30 minutes, 3 is added to the score. If a driver does not require a meal during recovery, 5 is added to the score. The ratio of the free time without a driver's workload divided by the time between the rescheduling time and sign-off time is also added to the score.

Spare drivers can be found as follows. A score for a spare driver for an uncovered task is calculated as follows: if the spare driver is at the station where the task departs, 1000 is added to the score, and the time difference between the task departure time and the driver’s sign off time is added to the score. This rule prefers to choose spare drivers who can start the task on time and have enough time to sign off after performing the task. If the spare driver is not at the station where the task departs, the time difference between the task departure time and when the driver can arrive at the station is added to the score. It prefers to choose spare drivers who have enough time to get ready for the task and be able to perform it.

With the chosen diagrammed and spare drivers, a disruption neighbourhood Δ_i is expanded to Δ_{i+1} by adding these drivers and their planned to be covered tasks during the recovery period.

Solving CRP with an Iterative Expanding Disruption Neighbourhood Approach

Subroutine Solve_CRP(), shown in Algorithm 9, describes how a disruption neighbourhood is dynamically expanded to obtain the best solutions to the crew rescheduling problem. It takes disruption neighbourhood Δ_i and maximum expand iteration as inputs. Step 1 presents a graph G for generating recovery diagrams using the tasks considered in the disruption neighbourhood Δ_i. In step 2, initial recovery diagrams are generated for each driver in Δ_i using graph G. In step 3, Solve_DisNhood() is called to get the best result for the current disruption neighbourhood. CRP_Obj is updated in step 4 if the crew rescheduling cost obtained with Δ_i is lower. In step 5, if there are driving tasks left uncovered and the iteration number is less than the maximum expansion iteration number, the disruption neighbourhood is expanded in Step 6, and Solve_CRP() is called again in step 7 to solve the crew rescheduling problem with Δ_{i+1}. The algorithm's output is the best solution found among all disruption neighbourhoods. In a solution, each driver is assigned a recovery diagram in the disruption neighbourhood.

```
Algorithm 9 Solve_CRP($\Delta_i$, Iter)
Input: $\Delta_i$ (neighbourhood), Iter
Initialise: $CRP\_Obj$=1e10, $CRP\_Solution = \{\}$, $Max\_Iter$

 1: generate graph $G$
 2: for each driver $d$ in $\Delta_i$, generate initial recovery diagrams $R^d$, $\mathcal{R} = \{R^d : d \in \Delta_i\}$
 3: $(DisNhood\_Obj, DisNhood\_Solution) = Solve\_DisNhood(\Delta_i, \mathcal{R})$
 4: update $CRP\_Obj$ and $CRP\_Solution$
 5: if there is task cancelled and Iter < $Max\_Iter$ then
 6:     expand $\Delta_i$ to $\Delta_{i+1}$
 7:     $Solve\_CRP(\Delta_{i+1}$, Iter+1)
 8: else
 9:     return
10: end if
Output: $CRP\_Obj$, $CRP\_Solution$
```

Algorithm 9 Solving CRP with expanding disruption neighbourhoods

Solving CRP for a Given Disruption Neighbourhood

Subroutine Solve_DisNhood() is executed for a given disruption neighbourhood in Algorithm 10 to solve a CRP model for the neighbourhood Δ_i using a fixing and column generation scheme. $\mathcal{R}$ is the set of all generated recovery diagrams for drivers in the disruption neighbourhood. In Algorithm 10, two loops are used in steps 1 and 3, representing the fixing and column generation schemes. A dual Lagrangian problem of the CRP model is built and solved in step 4 to produce the best lower bound, Col_Lb, to the CRP model. A greedy algorithm is used in step 5 to find the upper bound, Col_Ub. The best crew rescheduling solution found so far is updated in step 6 if the cost produced by step 5 is lower. A pricing problem is solved in step 7 to generate potential recovery diagrams. The terminal condition for the column generation scheme is: no new recovery diagrams are generated.

A fixing scheme is used in steps 13 to 16. Suppose a diagram is repeatedly chosen for a driver in solving the dual Lagrangian problem in step 4. This diagram is fixed for this driver, and no pricing problem is solved in step 7 for the driver. The condition to fix a diagram for a driver is that the probability of choosing this diagram in solving the dual Lagrangian problem with Max_Iter (set as 100) is not smaller than 0.7. The terminal condition for the fixing scheme is that no diagram is fixed, or the upper bound of the fixing scheme is very close to the lower bound of the fixing scheme.

Algorithm 10 Solve_DisNhood(Δ_i, $\mathcal{R}$)

Input: Δ_i (neighbourhood), $\mathcal{R}$

Initialise: $DisNhood\_Obj = 1e10$, $DisNhood\_Solution = \{\}$, $Fix\_Ub = 1e10$, $Fix\_Lb = 0$, $Not\_Fix\_Drivers$ ={all drivers in Δ_i}

```
1: while not $End\_Fix$ do
2:   $End\_Col\_Gen$ =false, $Col\_Ub = 1e10$, $Col\_Lb = 0$
3:   while not $End\_Col\_Gen$ do
4:     $(Col\_Lb, \lambda\_List, Reduced\_Cost\_List, \lambda^*)=Solve\_LDP(\mathcal{R}, Fix\_Ub)$
5:     $(Col\_Ub, Greedy\_Solution)$=Greedy($\Delta_i$, $\lambda\_List$, $Reduced\_Cost\_List$)
6:     update $DisNhood\_Obj$, $DisNhood\_Solution$ if $Col\_Ub < DisNhood\_Obj$
7:     $\mathcal{R}=Solve\_Pricing\_Problem(Not\_Fix\_Drivers, \mathcal{R}, \lambda^*)$
8:     if no diagram is added then
9:       $End\_Col\_Gen$ = true
10:    end if
11:  end while
12:  $Fix\_Ub = Col\_Ub$, $Fix\_Lb = Col\_Lb$
13:  if no new diagram is fixed or $Fix\_Ub$ is close to $Fix\_Lb$ then
14:    $End\_Fix = true$
15:  else
       fix drivers and update $Not\_Fix\_Drivers$
16:  end if
17: end while
18: return
```

Output: $DisNhood\_Obj, DisNhood\_Solution$

Algorithm 10 Solving CRP with a given disruption neighbourhood

Solving Lagrangian Dual Problem

Variable λ is the vector of Lagrangian multipliers. λ_t is the corresponding multiplier for task t. The Lagrangian relaxation problem (LRP) for the CRP model is:

$$\min_{x,f} \sum_{d\in D} \sum_{r\in R^d} c_r^d x_r^d + \sum_{t\in T} p f_t + \sum_{t\in T} \lambda_t \left(1 - \sum_{d\in D} \sum_{r\in R^d} a_{tr}^d x_r^d - f_t\right)$$

$$= \min_{x,f} \sum_{t\in T} \lambda_t + \sum_{d\in D} \sum_{r\in R^d} (c_r^d - \sum_{t\in T} \lambda_t a_{tr}^d) x_r^d + \sum_{t\in T} (p - \lambda_t) f_t$$

$$s.t. \sum_{r \in R^d} x_r^d = 1, \forall d \in D$$

$$x_r^d \in \{0,1\} \; \forall d \in D, \forall r \in R^d \; and \; f_t \in \{0,1\} \; \forall t \in T$$

There exists an apparent optimal solution to the LRP model. For each driver d, diagram r where $c_r^d - \sum_{t \in T} \lambda_t a_{tr}^d$ achieves its minimum is chosen. For each task t, let f_t be 1 if $(p - \lambda_t) < 0$, otherwise, f_t is 0.

The solution value to the LRP model is a lower bound of the CRP model. Thus, a Lagrangian lower-bound problem should be solved. The dual Lagrangian problem (LDP) is defined as

$$\max_{\lambda} LRP$$

A subgradient algorithm based on (Stephen, Lin, & Almir, 2004) is used to solve the LDP model, shown in Algorithm 11. It takes recovery diagrams R and an upper bound as inputs. It iterates to get the best lower bound and λ. In the beginning, a Lagrangian multiplier λ is initialised with 0. In step 2, a Lagrangian relaxation problem is solved using the method as described above. z^* is the optimal value. x^* is the optimal solution consisting of variables x_r^d and f_t. The best lower bound is updated with z^* in step 3. To update λ in Step 7 in Algorithm 11, the subgradient is calculated for each driving task with the following formula:

$$\delta_t = 1 - \sum_{d \in D} \sum_{r \in R^d} a_{tr}^d x_r^d - f_t$$

In step 9, if the best lower bound $Best\_Lb$ is not updated for more than a specific time, the constant $const$ used to update λ is halved. In step 12, if the distance (dist) is close to 0, the algorithm terminates. The output of this algorithm is the best lower bound, the list of λ generated in Max_Iter iterations (λ_List), the list of reduced costs generated in Max_Iter iterations (Reduced_Cost_List), and λ^* (where the LDP attains maximum).

Algorithm 11 Solve_LDP($\mathcal{R}$,Ub)

```
Input: $\mathcal{R} = \{R^d : d \in \Delta_i\}$ (recovery diagrams for all drivers in $\Delta_i$), $Ub$
Initialise: $\lambda_t = 0\ \forall t \in T$, Iter=0, const=2, $\epsilon = 1e-5$, $Best\_LB = -1e10$,
$LDP\_Sol = \{\}$, $\lambda\_List = []$, $Reduced\_Cost\_List = []$, $Max\_Iter = 100$, $\lambda^* = []$

 1: while Iter < $Max\_Iter$ do
 2:    $(z^*, x^*) = Solve\_LRP(\lambda, \mathcal{R})$
 3:    if $z^* > Best\_Lb$ then
 4:       $Best\_Lb = z^*$
 5:       $\lambda^* = \lambda$
 6:       $LDP\_Sol = x^*$
 7:    end if
 8:    update $\lambda$:
       $dist = \sum_{t \in T} \delta_t^2$, where $\delta_t = 1 - \sum_{d \in D} \sum_{r \in R^d} a_{tr}^d x_r^d - f_t$
       $step\_size$ = const $\times$ $(Ub - z^*)/dist$
       $\lambda_t = \max(0, \lambda_t + \text{step\_size} \times \delta_t)\ \forall t \in T$
 9:    if no improvement for more than a certain time then
10:       const=const/2
11:    end if
12:    if dist $\leq \epsilon$ then
13:       break
14:    end if
15:    add vector $\lambda$ to $\lambda\_List$
16:    calculate reduced cost for each chosen diagram in $x^*$ and add them to
       $Reduced\_Cost\_List$
17:    iter=iter+1
18: end while
19: return
Output: $Best\_Lb$, $\lambda\_List$, $Reduced\_Cost\_List$, $\lambda^*$
```

Algorithm 11 Solving a dual Lagrangian problem using the subgradient method

Greedy Algorithm

To obtain an upper bound for the CRP problem, a greedy algorithm based on (Potthoff, 2010)'s work (Algorithm 12) is used. It takes recovery diagrams generated for all drivers $\mathcal{R}$, a list of Lagrangian multipliers, and a list of reduced costs for all drivers as input. It loops over all Lagrangian multipliers λ to find a best solution.

In step 2, drivers are ordered by the amount of cost reduction in increasing order. In step 3, for each λ, a vector z is initialised with 1. Recall that z_t being equal to 1 or 0 represents whether the task is cancelled. At first, each task is assumed cancelled. In step 4, for each driver d, a recovery diagram r with the least reduced cost calculated with λ is chosen. Vectors λ and z are updated based on the recovery diagram r. For all tasks t covered by r, we set λ_t and z_t equal to 0. During the experimental tests, it is noticed that some drivers could be assigned to a recovery diagram which does not cover any task. Thus, an improvement scheme is used from steps 6 to 16. Step 8, if task t is not covered, the corresponding λ_t is set as the cost for task cancellation. At step 9, drivers whose recovery diagrams do not cover any task are selected, denoted $\underline{D}$. At step 10, for each driver d in $\underline{D}$, the diagram r with the least reduced cost is chosen. If no task can be covered anymore, the improvement scheme stops at step 16.

In step 17, the recovery diagram cost is calculated as the sum of the recovery diagram cost for all drivers. In step 18, the task cancellation cost is calculated as the multiplication of the cancellation penalty and the number of cancelled tasks. In step 19, the total cost is computed as the sum of the recovery diagram and task cancellation costs. In step 20, Greedy_Obj and Greedy_Solution are updated if the newly generated total cost is lower than the current Greedy_Obj. The output of the algorithm is Greedy_Obj and Greedy_Solution.

```
Algorithm 12 Greedy($\mathcal{R}$, $\lambda$_List, Reduced_Cost_List)
Input: $\mathcal{R} = \{R^d : d \in \Delta_i\}$ (recovery diagrams for all drivers in $\Delta_i$), $\lambda$_List
(list of $\lambda$), Reduced_Cost_List (list of reduced costs)
Initialise: Greedy_Obj =1e7, Greedy_Solution = {}

 1: for $\lambda$ in $\lambda$_List do
 2:   Driver_List = order drivers by the increasing of reduced cost
 3:   initialise $z_t = 1$ for each task $t$ in $\Delta_i$
 4:   for each driver d in Driver_List do
        choose the diagram $r$ with the least reduced cost, update $\lambda_t = 0$ and
        $z_t = 0$ for all tasks $t$ that diagram $r$ covers
 5:   end for
 6:   IMPROVE = true
 7:   while IMPROVE do
 8:     set $\lambda_t = p$ if $z_t = 1$ for each t. p is the task cancellation penalty
 9:     choose drivers $\underline{D}$ ={$d$: if recovery diagram $r$ chosen for driver $d$ does
        not cover any driving task }
10:     for $d$ in $\underline{D}$ do
11:       choose the diagram $r$ with the least reduced cost
          update $\lambda_t = 0$ and $z_t = 0$ for all tasks $t$ that diagram $r$ covers
12:     end for
13:     if no $\lambda_t$ has been updated from 1 to 0 then
14:       IMPROVE = false
15:     end if
16:   end while
17:   recovery diagram cost= sum the costs of chosen recovery diagram r for
      each driver d.
18:   task cancellation cost= cancellation penalty $\times$ the number of $z_t$ where
      $z_t = 1$
19:   Total_Cost=recovery diagram cost+task cancellation cost
20:   update Greedy_Obj, Greedy_Solution if Total_Cost < Greedy_Obj
21: end for
22: return
Output: Greedy_Obj, Greedy_Solution
```

Algorithm 12 Greedy algorithm

Solving a Pricing Problem

A pricing problem is solved to generate new promising recovery diagrams in LRCG. A recovery diagram r for driver d is a list of driving tasks with a cost c_r^d. Finding a recovery diagram means finding the shortest path with resource constraints in a graph $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$. Resource constraints come from the necessary meal breaks included in recovery diagrams.

The arc cost in $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$ is further priced by the multipliers λ^* obtained in Step 4 of Algorithm 10 before any subgraph and recovery diagrams are produced. More precisely, λ^* is deducted from the arc cost if this arc connects to the node representing the driving task t, to obtain the reduced cost of a recovery diagram defined by

$$\overline{c}_r^d(\lambda^*) = c_r^d - \sum_{t \in r} \lambda_t^*$$

It is the cost of a diagram from which the sum of Lagrangian multipliers representing the driving tasks included in this diagram is deducted. A new promising recovery diagram p will be added to R^d if $\overline{c}_p^d(\lambda^*)$ is smaller than the minimum $\overline{c}_r^d(\lambda^*)$, where r is an existing diagram for driver d. This guarantees that no repeat recovery diagrams will be added to the current set of recovery diagrams for the driver. This process is shown in Algorithm 13. The output of this algorithm is the set of recovery diagrams.

```
Algorithm 13 Solve_Pricing_Problem($Not\_Fix\_Drivers$, $\mathcal{R}$, $\lambda^*$)
Input: $Not\_Fix\_Drivers$, $\mathcal{R} = \{R^d : d \in \Delta_i\}$ (recovery diagrams for all drivers in $\Delta_i$), $\lambda^*$

1: for each d in $Not\_Fix\_Drivers$ do
2:     p=SPPRC($d$,$\lambda^*$): find new recovery diagrams $p$ with lowest reduced cost $\overline{c}_p^d(\lambda^*)$
3:     add $p$ into $R^d$ if $\overline{c}_p^d(\lambda^*)$ is smaller than the minimum $\overline{c}_r^d(\lambda^*)$, where diagram $r$ is an existing recovery diagrams for driver $d$
4: end for
Output: $\mathcal{R} = \{R^d : d \in \Delta_i\}$ (recovery diagrams for all drivers in $\Delta_i$)
```

Algorithm 13 Solving a pricing problem

Since the SPPRC problem is an NP-hard problem, solving it optimally may take too much time for some scenarios. In this work, solving SPPRC terminates early when a certain number of recovery diagrams are found. Three search techniques (bi-directional, forwards and backwards) are used with dynamic programming to find recovery diagrams. Their performances are compared in Section 4.4.7 using the experimental tests in Section 4.4.6.

4.4.5. a Didactic Example

Consider the following simple scenario. The initial plan in the morning is for all three drivers to drive different trains from the station W to P. However, due to a broken rolling stock problem at 6 am, train service from W to B is cancelled for train $1F03$ and train service from W to C is cancelled for the train $1B01$. Drivers Tony and William are stuck at the station W and their following tasks start from stations C and B, respectively. (To simplify the problem, we assume that there are spare rolling stock that can be used for their next job). Another driver, Tim, is going to drive a train $1F07$ departing from the station W at 6:15 am. We assume that the rescheduling process should end at 10:15 am when every driver should arrive at the station P and prepare for their next task. The tasks that need to be covered in this scenario are tasks 1, …, and 5, shown in Table 13.

Table 13 Task information for a didactic example

TaskId	Dep	Arrive	Origin	Destination	Train
1	08:15	9:45	C	P	1B01
2	07:30	10:00	B	P	1F03
3	06:15	06:50	W	B	1F07
4	07:00	08:00	B	C	1F07
5	08:45	10:15	C	P	1F07

To solve this problem, a crew rescheduling model is built as in Section 4.4.1. Table 14 is the constraint matrix in the model. Each column generation iteration generates a new group of recovery diagrams. They are shown in Table 14 in the corresponding groups of columns. Each row represents a constraint in the model. A line in the middle separates the constraints for drivers and tasks. For each row, the sum of variables whose coefficients are 1 should be equal to 1.

Table 14 Constraint matrix

Conts	Ini_0			Col_1			Col_2			Col_3			Col_4			Col_5		
	Will	Tony	Tim	Will	Tony	Tim	Will	Tony	Tim	Will	Tony	Tim	Will	Tony	Tim	Will	Tony	Tim
Will	1			1			1			1			1			1		
Tony		1			1			1			1			1			1	
Tim			1			1			1			1			1			
task1				1	1								1	1		1	1	
task2							1	1		1	1							
task3	1	1	1							1	1	1	1	1	1			
task4				1	1	1						1						
task5						1			1			1			1			

Every row is a constraint in the model, and every column is a variable x_r^d representing one recovery diagram for one driver. Figure 19 is a simple example of the graph used to find recovery diagrams for driver Tim. It contains the source node, sink node and nodes representing all tasks in the disruption neighbourhood. It uses five basic arcs which do not consider the meal break. In Figure 19, “Train deadheading twice” means that driver Tim uses two other tasks as passenger trips to arrive at the origin of the next task to perform it from current location. For example, Tim uses task3 and task4 as passenger trips to arrive at task1 to perform it after his current location at station W. The pricing problem is to find a path in this graph for driver Tim with the most negative reduced cost. Train deadheading means a passenger trip.

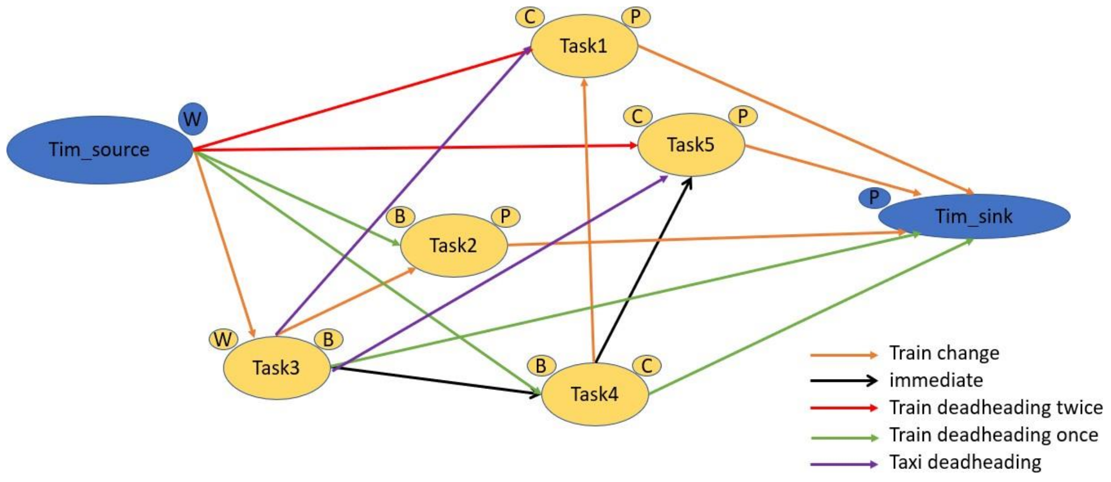


Figure 19 Subgraph for driver Tim

In the solution, Tim performs tasks 3, 4, and 5. Tony performs task 1, and William performs task 2. The interpretation of this result is that Tim drives the train from W to P performing tasks 3, 4, and 5. Tony and William take a passenger trip on the train that Tim drives and get off at the stations C and B respectively. Then they perform tasks 1 and 2. This solution corresponds to the red columns in Table 14. The solution path for Tim can also be found easily in Figure 19.

4.4.6. Experimental Tests

The model for CRP and LRCG method is applied to the 14 scenarios considered in Section 4.1.6. These 14 scenarios have been described in Section 4.1.6, and revised timetables have been obtained. A simple algorithm to reschedule rolling stock is used since rescheduling rolling stock itself is a complex problem. The turnaround time is 15 minutes at the stations where trains short turn before blockage and 5 minutes at other stations. The maximum delay is set as 10. The dataset used in the experimental tests has 222 diagrammed drivers and 61 spare drivers. These spare drivers are attached to 10 depots across the day. Table 15 shows the 14 scenarios, the number of affected drivers and drivers in the initial disruption neighbourhood as found with the method shown in Algorithm 8.

Table 15 Blockage scenarios and their effect on drivers

Id	Recovery period	Directly affect drivers	1st neighbourhood

AA_06	06:00-11:00	9	52
AA_07	07:00-12:00	9	70
AA_08	08:00-13:00	13	56
AA_09	09:00-14:00	11	11
AA_10	10:00-15:00	11	11
AA_11	11:00-16:00	10	10
AA_12	12:00-17:00	10	10
AA_13	13:00-18:00	10	10
AA_14	14:00-19:00	13	13
AA_15	15:00-20:00	12	16
AA_16	16:00-21:00	13	16
AA_17	17:00-22:00	16	59
AA_18	18:00-23:00	13	49
AA_19	19:00-24:00	10	12

The number of directly affected drivers varies from 9 (06:00 - 07:00) in the early morning to 16 in the late afternoon (17:00-22:00). During the day, the number of directly affected drivers keeps at 10 constantly from scenario AA_11 to AA_13. For the drivers included in the first disruption neighbourhood, it is noticeable that scenarios in the early morning (AA_06, AA_07, AA_08) and evening (AA_17, AA_18) have a large number of drivers. One reason is that many short trips are required to move rolling stock to prepare for the working day or back to the rolling stock depot during these times.

The maximum allowed number of disruption expansions is set as 3. Thus, no more than 4 disruption neighbourhoods can be searched. The connection time is set as 10 minutes. The maximum work length is set as 10 hours. The maximum continuous work time is set as 4 hours and 40 minutes. Maximum work overtime is 30 minutes. Communication time is set as 10 minutes. Meal break is set as 30 minutes (with connection time, 50 minutes can be used for drivers to walk to their depot, have a meal and walk back to the platform for the next task). Three methods are tested for solving SPPRC problems: bi-directional search, forwards search, and backwards search. The results for a bi-directional, forwards and backwards search are explained in what follows.

Bi-directional Search

Table 16 Crew rescheduling with solving SPPRC using bi-directional search

	ITER	ND	NT	UB	LB	ColGen	Gap %	NCT	Solution time (s)	Objective	Spare
AA_06	1	52	180	705	705	8	0.00	0	31	705	0

AA_07	1	70	219	365	365	6	0.00	0	50	365	0
AA_08	1	56	159	560	560	5	0.00	0	33	560	0
AA_09	2	47	125	560	500	5	12.00	0	32	560	0
AA_10	2	41	126	720	720	10	0.00	0	19	720	0
AA_11	2	55	115	590	569	5	3.69	0	25	590	0
AA_12	2	73	176	1410	1410	14	0.00	0	54	1410	1
AA_13	2	65	173	730	730	6	0.00	0	43	730	0
AA_14	2	69	173	1245	1245	12	0.00	0	122	1245	1
AA_15	2	95	227	895	850	6	5.29	0	228	895	0
AA_16	2	55	184	1205	1205	12	0.00	0	147	1205	1
AA_17	4	80	227	5830	5413	27	7.70	1	220	5830	0
AA_18	2	76	206	645	645	10	0.00	0	82	645	0
AA_19	2	28	78	1150	1150	7	0.00	0	13	1150	1

In Table 16, ITER is the disruption neighbourhood when the method terminates. ND and NT represent the number of drivers and tasks in the disruption neighbourhood. UB and LB are the upper and corresponding lower bound for the scenario. The UB is the lowest cost found in the method and LB is the corresponding lower bound. Mostly the lowest upper bound is obtained just before the method terminates. Gap is calculated as the percentage of by how much an UB is bigger than a LB. The best solution value is between the upper bound and lower bound. When the gap is significant, the method does not find a solution close to the optimum, otherwise it finds the best solution. ColGen stands for the number of times when new recovery diagrams are generated (pricing problems are solved and new recovery diagrams are added to the current RMP, steps 7 and 8 in Algorithm 10). NCT stands for the number of cancelled tasks in the solution. Spare stands for the number of used spare drivers in the solution.

The first three scenarios (AA_06 - AA_08) are solved in the initial disruption neighbourhood. Most of the remaining scenarios are solved in the second disruption neighbourhood except scenario AA_17, which ends in the fourth disruption neighbourhood. Scenario AA_19 is solved with the least drivers considered (28 drivers and 78 driving tasks). Scenario AA_15 is solved with most drivers considered (95 drivers and 227 tasks). Most scenarios are solved with a gap of 0 between the upper bound and lower bound. Scenario AA_17 requires one task to be cancelled. Other scenarios can find optimal solutions that cover all driving tasks as required in revised timetables. Most scenarios are solved within 1 minute. Scenarios (AA_14 - AA_17) require more time (from 122 seconds to 228 seconds). The reason is that more drivers and tasks are considered, and more iterations of column generation in these scenarios.

Forwards Search

Table 17 Crew rescheduling with solving SPPRC using forwards search

	ITER	ND	NT	UB	LB	ColGen	Gap %	NCT	Solution time (s)	Objective	Spare
AA_06	2	62	192	595	595	17	0.00	0	59	595	0
AA_07	1	70	219	780	780	3	0.00	0	54	780	0

AA_08	1	56	159	450	450	6	0.00	0	35	450	0
AA_09	2	53	123	495	460	8	7.61	0	59	495	0
AA_10	2	51	112	465	465	15	0.00	0	31	465	0
AA_11	2	61	117	735	729	17	0.82	0	32	735	0
AA_12	2	64	157	1995	1995	18	0.00	0	54	1995	2
AA_13	2	71	139	1870	1258	17	48.65	0	69	1870	2
AA_14	2	78	169	1400	1400	11	0.00	0	152	1400	1
AA_15	2	74	206	910	910	15	0.00	0	364	910	0
AA_16*	2	57	176	1475	1143	17	29.05	0	518	1475	1
AA_17	4	80	227	5700	5254	52	8.49	1	385	5700	0
AA_18	1	49	173	860	795	4	8.18	0	30	860	0
AA_19	2	42	111	1055	993	7	6.24	0	23	1055	1

Table 17 shows the crew rescheduling results with SPPRC solved using a forwards search. As in Table 17, most scenarios are solved in the first or second disruption neighbourhood except AA_17 in the fourth disruption neighbourhood. Forwards search fails to find solutions to cover all tasks for AA_17, with 1 task left uncovered. For most scenarios, the gaps are smaller, while the gap for scenario AA_13 is close to 50%. Scenario AA_16 is marked with * because the algorithm stops early, as solving this scenario requires more than 500 seconds. Scenarios AA_15 and AA_17 also need a long time to solve (364 and 385 seconds). Other scenarios can be solved in two minutes or even one minute. It is noticeable that the iteration of column generation for scenario AA_17 is as high as 52, which consumes a lot of running time.

Backwards Search

Table 18 Crew rescheduling with solving SPPRC using backwards search

	ITER	ND	NT	UB	LB	ColGen	Gap %	NTC	Solution time (s)	Objective	Spare
AA_06	2	62	204	1515	1159	5	30.72	0	65	1515	2
AA_07	2	82	242	2270	10	8	22600.00	0	94	2270	3
AA_08	2	69	191	950	941	8	0.96	0	64	950	1
AA_09	2	41	105	440	440	5	0.00	0	25	440	0
AA_10	2	35	109	465	453	2	2.65	0	18	465	0
AA_11	2	59	134	590	564	4	4.61	0	31	590	0
AA_12	2	77	180	2530	2277	3	11.11	0	53	2530	3
AA_13	2	35	107	650	650	6	0.00	0	19	650	0
AA_14	2	77	174	1370	1370	6	0.00	0	97	1370	1
AA_15	2	112	249	675	645	7	4.65	0	323	675	0
AA_16	2	71	209	2590	1270	6	103.94	0	242	2590	3
AA_17	4	86	232	5675	5578	13	1.74	1	232	5675	0
AA_18	1	49	173	610	610	3	0.00	0	27	610	0
AA_19	2	28	78	1500	1162	3	29.09	0	12	1500	2

Table 18 shows the crew rescheduling by solving SPPRC using a backwards search. Most scenarios are solved in the first and second disruption neighbourhoods except scenario AA_17 in the fourth disruption neighbourhood. All scenarios can find solutions that cover all driving tasks except AA_17 which requires one task to be cancelled. It is worth noting that the same task is cancelled for AA_17 in all three methods: bi-directional, forwards and backwards search. This task is a train trip, which is a part of the train service that departs at 16:54 from S1 in Figure 15. This is an extra train added during the peak time on the route from S1 to S4. This task is not cancelled in the timetable rescheduling step. However, in the crew rescheduling step, no spare drivers can arrive on time to take this task after rescheduling, thus, it needs to be cancelled.

The gaps for most scenarios are close to 0. The gap for scenario AA_07 is as high as 22600%. It is noticed that when the method terminates, the gap between the last upper bound and lower bound is 0 for this scenario. However, the last upper bound is not the lowest cost, which usually it is. The reason for this is that the method uses a fixing scheme. Some drivers' diagrams are fixed based on the rules during the solution process to speed up the method (Section 4.4.4). However, such fixing can lead to a worse upper bound to the crew rescheduling problem.

The solution time is quick, with most scenarios being solved within or just above 1 minute except 323 seconds for AA_15, 242 seconds for AA_16 and 232 seconds for AA_17. All scenarios in Table 18 have less than 20 times of column generation iterations.

4.4.7. Comparison: Bi-directional, Forwards and Backwards Search Methods

The performance of the three search methods is compared by cost, speed, problem size and gap.

Cost Comparison

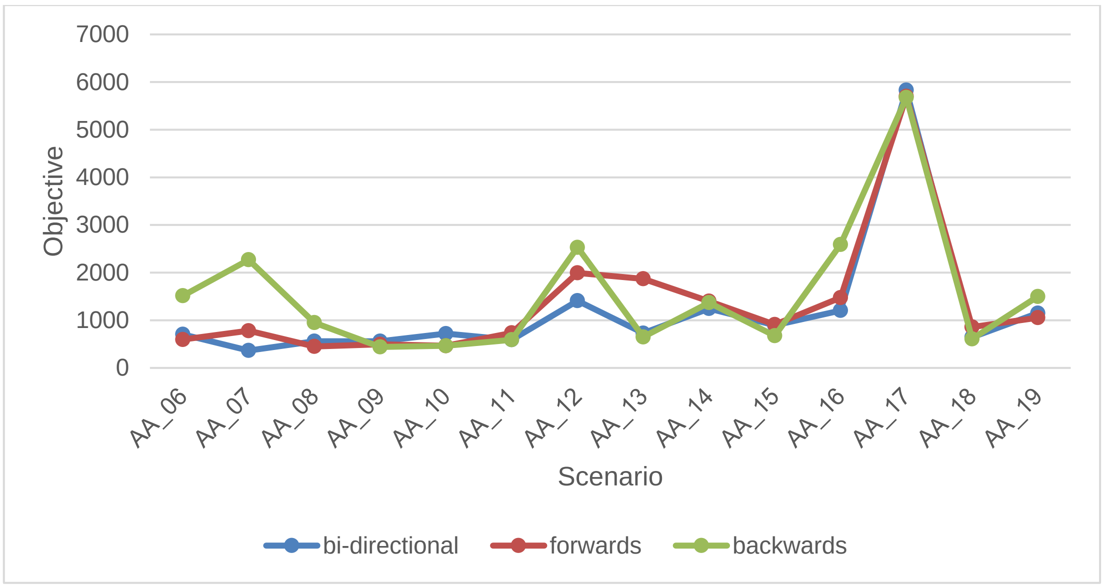


Figure 20 Costs compared for the three methods

The optimal costs obtained for 14 scenarios using bi-directional, forwards and backward searches are compared in Figure 20. Overall, the three methods have similar trends. The costs using three methods are low for scenarios AA_06 to AA_11, except that the backwards search finds a higher cost solution for AA_07. All three methods find solutions with higher costs for scenarios AA_12 and AA_17. In general, backwards search can find solutions with lower costs in most scenarios (7) followed by bi-directional search (5) and forwards search (4). One reason that backwards can find the least cost solutions for most scenarios is that it searches a bigger disruption neighbourhood with more drivers and tasks. The comparison of problem size for all three searches is shown later in this section.

Solution Time Comparison

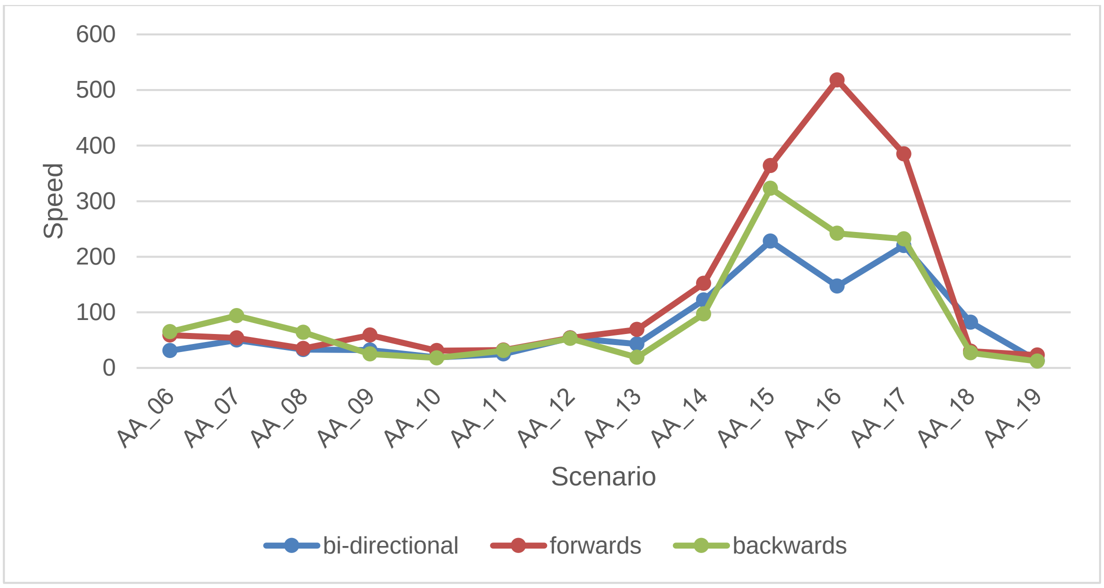


Figure 21 Solution time compared for the three methods

The running time of the three methods on 14 scenarios is shown in Figure 21. Three methods find solutions within 100 seconds for scenarios AA_06 to AA_13 and AA_18 to AA_19. Between scenarios AA_14 and AA_17, it is obvious that forwards search uses the most time to find optimal solutions. In general, backwards search uses the least time for 7 scenarios and bi-directional search uses the least time for the other 7 scenarios.

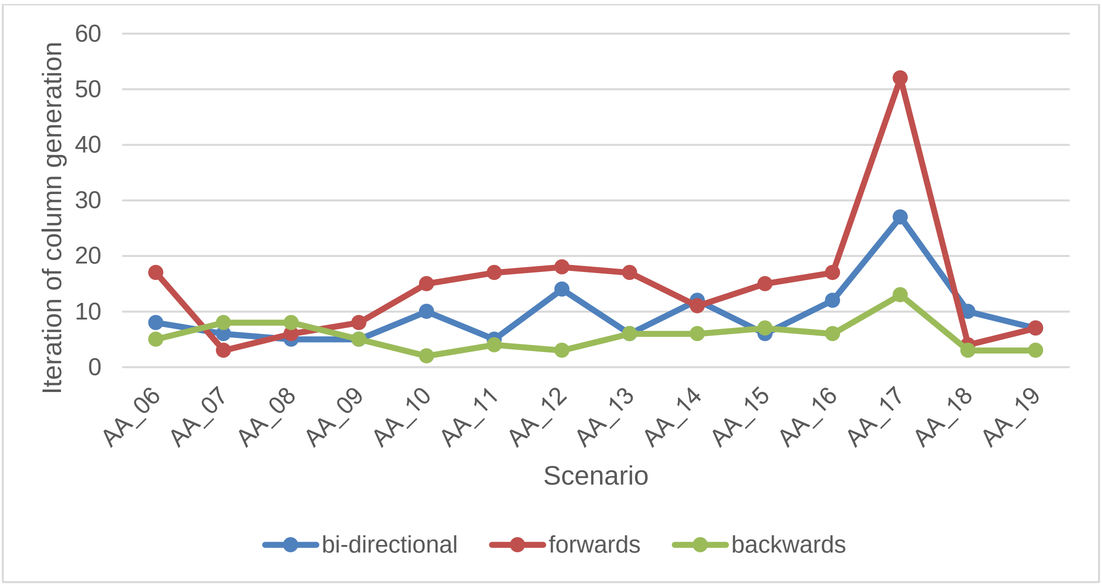
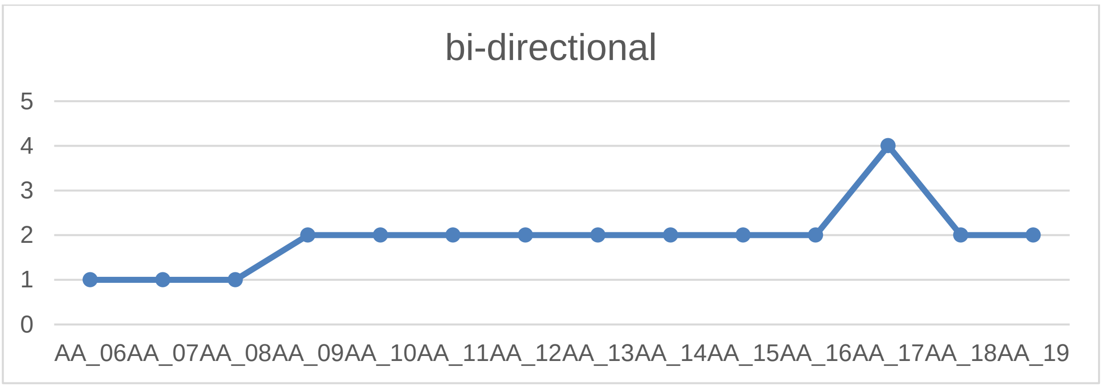


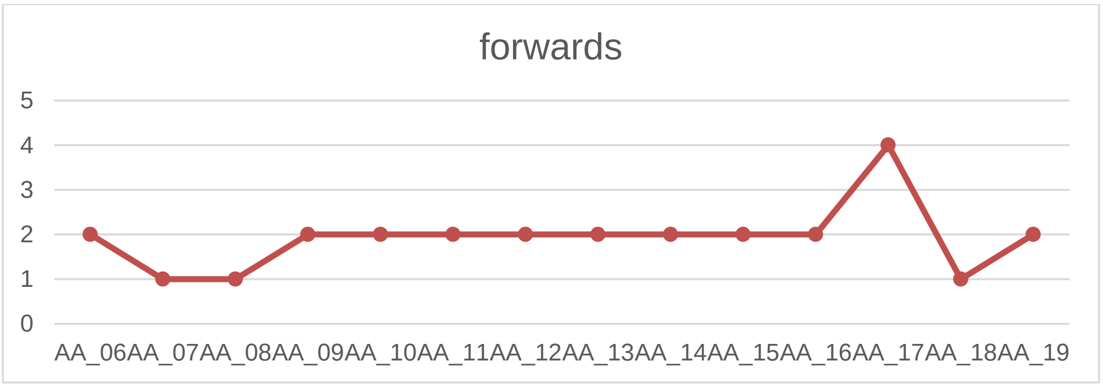
Figure 22 Iteration of column generation compared for the three methods

Many factors can affect the running time, like the number of drivers and tasks considered in the model and also the iteration of column generation. Figure 22 shows the iteration of column generation for three methods. It is clear that forwards search has the highest number of iterations for most of the scenarios (10 out of 14 scenarios). Recall that forwards search finds recovery diagrams from a source node representing a commencing activity to a sink node representing a terminating activity. Backwards search finds recovery diagrams in the opposite direction. During the experimental tests, it is noticeable that, from the source node, many more arcs can be generated compared to the arcs that connect to the sink node for a driver. It means that the number of tasks the commencing activity can connect to is higher than the number of tasks that can connect to the terminating activity for a driver. The reason for this is that a driver may have enough time to arrive at a task to perform it, but the driver may not be able to sign off considering overtime in time after performing this task.

Since only a few possible recovery diagrams can be developed starting from the sink node side, backwards search has a smaller iteration of column generation. In steps 8 and 9 of Algorithm 10, if no recovery diagrams are generated, the column generation loop terminates, and the fixing scheme starts to work. Thus, backwards search spends less time on the column generation loop. Note that if enumerating all recovery diagrams is required, both forwards and backwards searches can enumerate the same subset of recovery diagrams.

Problem Size Comparison





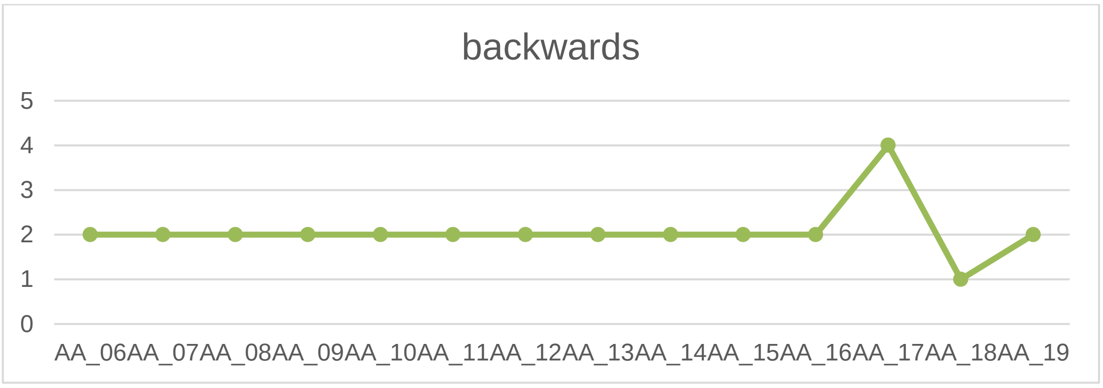


Figure 23 Disruption neighbourhood iterations in the three methods

Figure 23 shows the number of disruption neighbourhood that the method has explored when it terminates. It is clear that all three methods search two disruption neighbourhoods for most scenarios. Forwards search and bi-directional search only search one disruption neighbourhood for 3 scenarios compared to 1 scenario for backwards search.

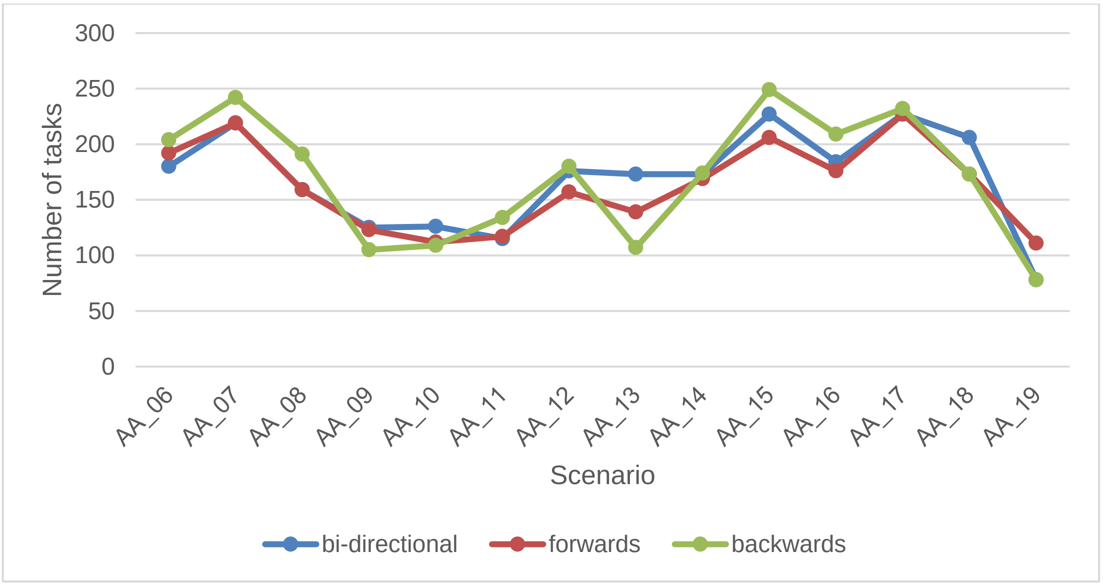


Figure 24 Number of tasks considered in the three methods

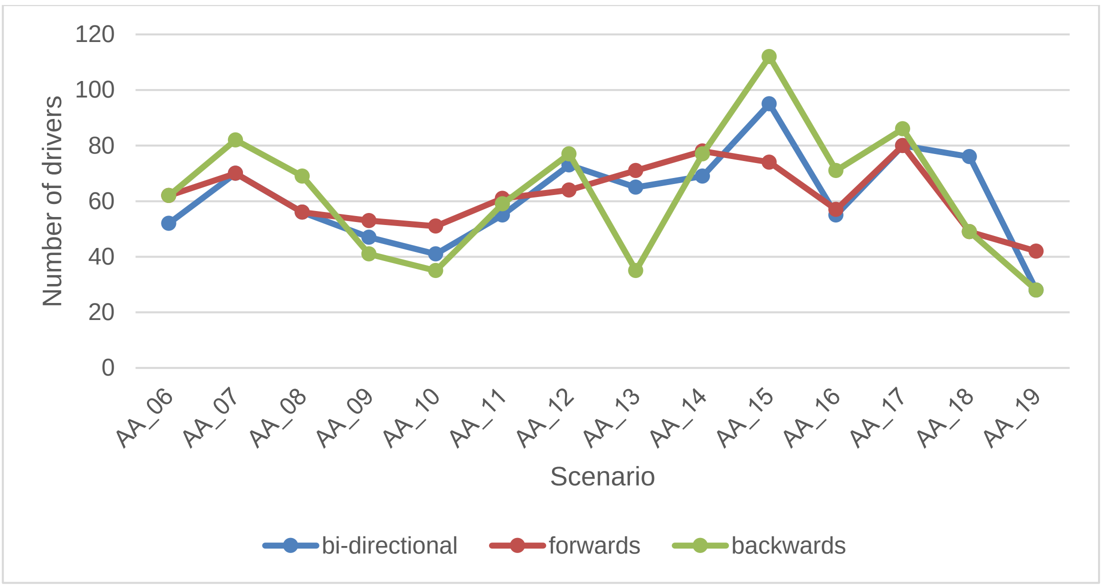


Figure 25 Number of drivers considered in the three methods

For the number of tasks considered in a model (Figure 24), backwards search has the biggest number of tasks for 9 scenarios, followed by bi-directional search for 4 scenarios and forwards search with 1 scenario. Overall, the three methods use roughly the same number of drivers (Figure 25) in scenarios AA_06 to AA_12 and AA_17. Forwards and backwards searches explore the greatest number of drivers for 7 scenarios, followed by bi-directional search for 1 scenario. It is clear that, in most scenarios, backwards searches a bigger disruption neighbourhood with more drivers and tasks. Recall the rules for expanding a disruption neighbourhood in Section 4.4.4, spare/diagrammed drivers are added for each uncovered task in the previous disruption neighbourhood. It is noticeable that backwards search finds a worse solution in the initial disruption neighbourhood, which leads to more spare/diagrammed drivers added to expand a disruption neighbourhood.

Gap Comparison

In all 14 scenarios, with bi-directional search, only 4 scenarios have gaps which are not 0 and the biggest gap is 12%. With forwards search, 7 scenarios have gaps which are not 0 and the biggest gap is almost 50%. With backwards search, 10 scenarios have gaps which are not 0 and the biggest gap is as high as 22600%. It is clear that bi-directional search has advantage in looking for solutions with smaller gaps. The reason for this is bi-directional searches the graph to construct recovery diagrams from both directions (a source node to a sink node, and a sink node to a source node). It searches a more averagely spread solution space. However, for forwards and backwards searches, they tend to search one side of the solution space with less attention to the other side. Thus, these two methods find solutions with bigger gaps, which means these solutions can be far from the optimal solutions.

To summarise, backwards search has advantages in speed even if it searches a bigger solution space. The reason is that in the column generation loop, backwards terminates quicker since less promising recovery diagrams can be generated. Thus, the backwards search uses less time to finish the algorithm. Also, since backwards search a bigger solution space, it has advantages in finding the minimal objective. Forwards search is the worst in speed and solution costs. Bi-directional search has advantages in finding solutions with small gaps since it searches a more averagely spread solution space.

4.5.Conclusion

This chapter addressed the crew rescheduling problem during significant disruptions. Since crew rescheduling requires a revised timetable as input. A model was first posed to adjust the timetable in case of a complete blockage. The model can provide a revised timetable for the three phases of disruption management while considering rolling stock, infrastructure availability, and blockage period. It uses rescheduling strategies: cancellation, and retiming and aims to revise the timetable to minimise train cancellations and delays, and the changes to rolling stock circulation patterns. The model was tested on the busiest line on a TOC from GB. The comparison between how a smaller and bigger maximum allowed delay affects the solutions was conducted. It was discovered that there is a trade-off between train cancellations and delays. Fewer trains must be cancelled when a longer maximum delay time is allowed. It is worth noting that the maximum delay time should not be longer than the period between frequently scheduled trains, otherwise cancellation may be more practical. More cancellations and delays can happen when trains require more time to turn around at a station before approaching the blockage site.

A model formulated as an integer linear programming problem was proposed to solve the crew rescheduling problem for significant disruptions. It was explained why a recovery diagram can be generated by solving SPPRC. Solving SPPRC to generate recovery diagrams has two steps: 1) build a graph using tasks and drivers considered in the model, 2) use dynamic programming and label setting algorithm to find recovery diagrams in the graph built in step 1. Further, three search methods which can combine with dynamic programming and change the search directions were explained.

A heuristic method, LRCG using Lagrangian relaxation and column generation, was proposed to solve the model for the crew rescheduling problem. The process uses an iterative approach to search for better solutions. A disruption neighbourhood is used to constrain the problem size. An algorithm was presented to show how to build the initial disruption neighbourhood. Rules were given on how to expand a disruption neighbourhood.

The model for the crew rescheduling problem and LRCG were applied to 14 scenarios using three different search methods (forwards search, bi-directional search and backwards search) to solve the SPPRC problem in LRCG. Crew rescheduling results obtained with LRCG using forwards, bi-directional and backwards searches were shown and compared. It was explained why backwards search can be the quickest and obtain the best results, forwards search can be the slowest regarding searching speed, and bi-directional search is the best in finding solutions with smaller gaps for most scenarios.

CHAPTER FIVE: SOLVING INTEGRATED ROLLING STOCK AND CREW RESCHEDULING PROBLEMS

Usually, the rescheduling of rolling stock and crew is carried out in two phases: rolling stock rescheduling followed by crew rescheduling. Theoretically, the adjustments between rolling stock and crew rescheduling can go back and forth for several rounds until a mutually compatible solution appears.

Similar to crew, each rolling stock unit also has a diagram to regulate their daily work. A diagram for a rolling stock unit contains a series of driving tasks and possible deadheading tasks (empty rolling stock moves on the network without passengers). When a disruption occurs, each rolling stock unit and crew member should be assigned a recovery diagram that is feasible to operate. Thus, to solve an integrated rolling stock and crew problem is to assign a recovery diagram to each rolling stock unit and driver that is considered in the problem.

One reason rolling stock and crew rescheduling is usually carried out sequentially is that the exact rolling stock type assigned to a task should be known. Then controllers can assign a driver with the required rolling stock knowledge to perform the driving task. This may be straightforward in some simple networks. For example, if only one rolling stock type can be used for a driving task, then the exact rolling stock type is known for the purpose of assigning a driver to cover the task. Another reason is that the integrated problem is usually of high complexity due to practical safety constraints both for rolling stock and crew. It is difficult to solve such complex problems in real time.

The potential benefits of rescheduling rolling stock and crew together are significant. It can provide a feasible solution for both rolling stock and crew rescheduling and is optimal for the integrated problem. It saves communication time between rolling stock controllers and crew controllers and can help an IM to publish reliable and timely railway traffic information to passengers during disruption.

“What should be the objective of resource rescheduling?” is a difficult question for the integrated rolling stock and crew rescheduling problem due to the various factors that need to be considered. To decrease the impact on passengers, minimising cancelled and delayed trips should be considered. To improve flexibility, spare drivers should be reserved for further potential disruption. To decrease implementation risk, fewer resources should be rescheduled. To prepare for the following operations, the rolling stock balance at each station should be as close as possible to the original plan. Further, finding the relative importance of these factors is another tricky subject. Different operators may value some factors more than others. Delaying a trip looks like a better solution than cancelling a trip. However, if considerable further delay is brought by an initial delay, cancelling the trip may be a better solution.

In this chapter, a model for the integrated rolling stock and crew rescheduling problem with retiming possibilities (IRSCRR) and a two-stage approach which uses multicriteria decision making (2SMO) to solve the IRSCRR are developed. The model is set as a multicriteria optimisation problem and two multicriteria optimisation techniques are used in 2SMO to solve the model. A feedback mechanism is used in 2SMO. The feedback mechanism can indicate which driving task not covered in the first step should be retimed and its exact delay time, which will be used in the second stage to generate solutions considering retiming possibilities.

The structure of the remainder of the section is as follows. In Section 5.1, research work related to the rolling stock rescheduling, or the integrated rolling stock and crew problems has been reviewed. Section 5.2 presents the model formulation for the integrated rolling stock and crew rescheduling problem with retiming possibilities. Section 5.3 describes the two-stage approach 2SMO used to solve the IRSCRR. Section 5.4 shows the performance of applying IRSCRR and 2SMO on various single delay scenarios. Section 5.5 shows the performance of the model and method on various multiple delay scenarios. Section 5.6 is the conclusion.

5.1. Literature Review

In the literature, the rescheduling of rolling stock and crew are usually studied separately and sequentially. For rolling stock rescheduling, (Budai, Maróti, Dekker, Huisman, & Kroon, 2010) studied the rolling stock rebalancing problem which is relevant both in the short-term planning and real-time operations. Two heuristics were developed to solve the rolling stock rebalancing problem and compared with each other and the performance of an exact method. (Nielsen, Kroon, & Maróti, 2012) solved the rolling stock rescheduling for disruption management. A single rolling stock problem is solved using a two-step approach: circulation generation phase and duty generation phase. To deal with the uncertainty of the impact and duration of a disruption, a methodology based on rescheduling with a rolling horizon was proposed. Furthermore, (Kroon, Maróti, & Nielsen, 2014) studied rolling stock rescheduling with dynamic passenger flow. An iterative approach involving rolling stock rescheduling and passenger flow simulation was proposed. A simulation model is used to generate expected passenger flows. The interpreting of flows can give optimisation directions to rolling stock rescheduling in the next iteration. The current literature in crew rescheduling has been revised in Section 2.5.

To the best of our knowledge there is only one paper which deals with the integrated rolling stock and crew rescheduling problem (Zeng, Meng, & Hong, 2018). An integrated rolling and crew rescheduling model based on a multi-commodity flow model is proposed, and a customised ant colony algorithm is developed to solve the integrated problem efficiently. However, in their model, the cross-check constraints for rolling stock and crew are not included. That is, if a task fails to be covered by rolling stock, then it cannot be covered by the crew, and vice versa. Also, the meal break needs of the crew during their work are not considered, which may cause the rescheduling result to violate fatigue rules.

Multicriteria optimisation problems are a special case of vector optimisation problems. In a single criterion optimisation problem, the definition of an optimal solution is straightforward. However, for multicriteria optimisation, it is rare to have a solution which attains minimum values simultaneously in all criteria. A more general definition of optimality is the Pareto optimum (see (Ehrgott, 2005), (T'kindt & Billaut, 2006)). Multicriteria optimisation methods have been used in timetable planning and rescheduling, (see (Sama, Meloni, D'Ariano, & Corman, 2015), (Stoilova, 2020)). However, multicriteria optimisation has not been used in rolling stock or crew rescheduling.

5.2. Integrated Rolling Stock and Crew Rescheduling Model with Retiming

A model is proposed in this section to provide rescheduling solutions for rolling stock and crew together. Compared to rescheduling rolling stock and crew sequentially, solving an integrated rolling stock and crew rescheduling problem requires some extra constraints. For example, if a task is not covered by any rolling stock unit, then it should not be covered by any crew member, and vice versa. As in Section 4.4.4, the disruption neighbourhood idea is used as a concept to help formulate and solve the rescheduling problem. In this section, it is characterised by the rolling stock and drivers that need to be rescheduled and a recovery period that should be set by the controllers. How to construct a disruption neighbourhood is explained in detail in Section 5.3.1.

This section uses the data from a TOC in Great Britain. In this TOC, most trains are operated by one self-powered rolling stock unit during the day. The situation of two or more rolling stock units are used by one train mostly appear in the early morning or late evening for moving rolling stock from or to depot. Sometimes rolling stock units are coupled or uncoupled to a train, called composition change. Rolling stock units that have been uncoupled from a train needs to be stored at the shunting yard of the station and can be used later for other trains. During shunting operations, some non-timetabled movements of rolling stock inside railway nodes are created due to composition changes. In this study, shunting is taken into account in an implicit manner: by ensuring a minimum connection time between tasks that require shunting in between.

A recovery diagram is assigned to each rolling stock unit as part of a solution. If the same task appears on two recovery diagrams assigned to different rolling stock units, it means that both rolling stock units cover this task. It implied that these two rolling stock units should be coupled to perform the same task. If after the same task, two rolling stock units are assigned to different tasks, these two are uncoupled to perform different tasks. As for the order of two or more units appearing in a composition and how they can be coupled and uncoupled are not considered in the model in this section.

5.2.1. Model Constraints

First, some notation is introduced before formulating the model IRSCRR. Let T be the set of tasks that need to be assigned to rolling stock units and drivers inside a disruption neighbourhood. A task should be covered by a driver and rolling stock units simultaneously (or cancelled otherwise). If a task is assigned to a rolling stock unit, it needs a driver to perform it, and vice versa. Let S and D denote the rolling stock units and drivers, respectively, that are considered in a disruption neighbourhood. The set $R^s(R^d)$ is the set of recovery diagrams generated for a rolling stock unit (driver), which corresponds to one disruption. As in the CRP model built in Chapter 4, the model for the integrated rolling stock and crew rescheduling problem is a path-based model. The construction of recovery diagrams is not explicitly considered in the model formula. They are constructed by solving SPPRC described in Section 4.4.3 and used in the model directly.

Four types of variables are considered. Let x_r^s be 1 if the recovery diagram r is chosen for the rolling stock unit s, 0 otherwise. Similarly, let y_r^d be 1 if the recovery diagram r is assigned to the driver d, 0 otherwise. Variable g_t is equal to 1 if the task t is cancelled due to unavailable rolling stock units, 0 otherwise. Variable h_t is equal to 1 if the task t is cancelled due to unavailable drivers. Two parameters are used: a_{tr}^s (b_{tr}^d) is 1 if the recovery diagram r for rolling stock unit s (driver d) contains task t, 0 otherwise. The following constraints should be considered to formulate a basic IRSCRR.

$$\left(\sum_{s\in S}\sum_{r\in R^s} a_{tr}^s\, x_r^s\right) + g_t \geq 1, \forall t \in T \tag{5.1}$$

$$\left(\sum_{d\in D}\sum_{r\in R^d} b_{tr}^d\, y_r^d\right) + h_t \geq 1, \forall t \in T \tag{5.2}$$

$$\sum_{r\in R^s} x_r^s = 1, \forall s \in S \tag{5.3}$$

$$\sum_{r\in R^d} y_r^d = 1, \forall d \in D \tag{5.4}$$

$$x_r^s + h_t \leq 1, (\forall t \in T), (\forall s \in S), (\forall r \in R^s\colon t \in r) \tag{5.5}$$

$$y_r^d + g_t \leq 1, (\forall t \in T), (\forall d \in D), (\forall r \in R^d: t \in r) \qquad (5.6)$$

Equations (5.1) and (5.2) mean that each task t should be covered by a rolling stock unit and driver (and cancelled otherwise). Equations (5.3) and (5.4) guarantee that precisely one recovery diagram should be assigned to each rolling stock unit and driver, respectively. Equation (5.5) means that if a task is cancelled due to unavailable drivers, it should not be used in any recovery diagrams that are selected for rolling stock units. Also, it implies that if the task t is used in a recovery diagram for a rolling stock unit, the task t should also be covered by drivers. Vice versa, in equation (5.6), if a task is cancelled due to unavailable rolling stock units, it cannot be used in any recovery diagrams selected for drivers. A task used in selected recovery diagrams for drivers should also be covered by rolling stock units.

5.2.2. Extra Constraints Concerning Retiming

Sometimes tasks cannot be covered due to the turnaround time required by rolling stock units at terminal stations or connection times required by a driver to change trains. In this situation, allowing tasks to be retimed may lead to fewer task cancellations. In IRSCRR, the possibility of retiming tasks is also considered. The retiming constraints from (Veelenturf, Potthoff, Huisman, & Kroon, 2012) are used. Retiming can give a rescheduling solution when there is no solution without retiming. However, retiming a task means that the timetable needs to be modified to reflect this. Here, these constraints are briefly introduced. For a task, several copies are created. One copy is the same as the task. The other copies are the retimed versions of the task, which means their departure and arrival times are shifted by a given time length. For a task t, it has a set of copies $C_t = \{c(t): c(t)$ is a copy of the task t and $c(t)$ has a non-negative delay time$\}$. The set C_t contains at least one copy representing the planned task itself. For each copy $c(t)$, the binary variable $v_{c(t)}$ is 1 if there is a recovery diagram that contains copy $c(t)$ which is selected in the solution, 0 otherwise. $t(c)$ is the task from which copy $c(t)$is generated. Let C be the union of all C_t. Moreover, $|S|$ and $|D|$ are the number of rolling stock units and drivers, respectively.

$$|S|v_{c(t)} - \sum_{s\in S}\sum_{r\in R^s} a^s_{c(t)r}\, x^s_r \geq 0, \forall c(t) \in C \tag{5.7}$$

$$|D|v_{c(t)} - \sum_{d\in D}\sum_{r\in R^d} b^d_{c(t)r}\, y^d_r \geq 0, \forall c(t) \in C \tag{5.8}$$

$$\sum_{c(t)\in C_t} v_{c(t)} + g_t = 1, \forall t \in T \tag{5.9}$$

$$\sum_{c(t)\in C_t} v_{c(t)} + h_t = 1, \forall t \in T \tag{5.10}$$

Constraints (5.7) and (5.8) guarantee that if a copy $c(t)$ is used in any selected recovery diagram, $v_{c(t)}$ will be set as 1. Constraints (5.9) and (5.10) make sure that either a task is cancelled or a copy of it is chosen.

Before writing out the constraints for delay propagation, some notation is first introduced. $C^{<}_{c(t),t} \subseteq C_t$ is defined as the set of all copies of task t which have a shorter delay time than that of copy $c(t)$. $C^{>}_{c(t),t} \subseteq C_t$ is the set of all copies which have a longer delay time than that of a copy $c(t)$. $af(t)$ is used to denote the successive task of task

t if the next task exists. $conn(t)$ is the minimum time for a rolling stock unit to connect t and $af(t)$ (meaning that the rolling stock unit performs the task t and $af(t)$ successively). A suitable copy $c(t) \in C_t$ should be chosen if a copy $e(af(t)) \in C_{af(t)}$ is chosen, where $e(af(t))$ is a copy of the task $af(t)$. For a copy $c(t)$, $L_{c(t)}$ is defined as

$$L_{c(t)} = \{e(af(t)) \in C_{af(t)} \backslash \cup_{\underline{c(t)} \in C^{<}_{c(t),t}} L_{\underline{c(t)}} \tag{5.11}$$

$$|e(af(t))^{dep} - c(t)^{arr} \geq conn(t) \text{ and } \forall \overline{c(t)} \in C^{>}_{c(t),t}\ e(af(t))^{dep} - \overline{c(t)}^{arr} < conn(t)\}\}$$

It denotes the set of e that copy c(t) can connect to but $\overline{c(t)}$cannot connect to, where $\overline{c(t)}$has a larger delay time than a copy $c(t)$. Also, $L_{c(t)}$ should exclude e that are already included in $L_{\underline{c(t)}}$ for copy $\underline{c(t)}$, where copy $\underline{c(t)}$ has a shorter delay time than that of copy $c(t)$ (thus, the definition of $L_{c(t)}$ is recursive).

For each copy $c(t)$ of a task t which has more than one copy and this task has a successive task $af(t)$, if a copy e is chosen for the task $af(t)$, then the task t should be cancelled, or an appropriate c should be chosen. Below, $T^{\geq 2}$ denotes the set of tasks that have at least two copies including the task itself. Constraint (5.12) means that for task t in $T^{\geq 2}$ and the next task $af(t)$ exists, if a copy for the next task is chosen, then the task needs to be cancelled (h_t=1) in terms of rolling stock or a suitable copy for t is chosen. Constraint (5.13) sets the same constraints for the crew.

$$g_t + v_{c(t)} + \sum_{c'(t) \in C^{<}_{c(t),t}} v_{c'(t)} - \sum_{c'(t) \in L_c(t)} v_{c'(t)} \geq 0, \forall t \in T^{\geq 2}: af(t) exists, \forall c(t) \in C_t \tag{5.12}$$

$$h_t + v_{c(t)} + \sum_{c'(t) \in C^{<}_{c(t),t}} v_{c'(t)} - \sum_{c'(t) \in L_c(t)} v_{c'(t)} \geq 0, \forall t \in T^{\geq 2}: af(t) exists, \forall c(t) \in C_t \quad (5.13)$$

5.2.3. Model Objective

An objective function $f(x, y, g, h, v)$ should consider the following aspects:

(1) **RDC**: cost of chosen recovery diagrams for drivers and rolling stock units considered in a model, which represents the deviation from the planned diagrams assigned to each driver and rolling stock unit. The cost of a recovery diagram is the sum of costs of all connection types in a recovery diagram. Please see Section 4.4.2 for the costs of connection types for driver recovery diagrams. The connection types for rolling stock recovery diagrams are shown in Section 5.3.3

(2) **NRC**: cost of the number of in drivers and rolling stock units that are not spare resource and take at least one task that is not their planned diagrams. The more such resources that are rescheduled during a disruption, the heavier the penalty should be. It is measured by the number of resources used and its weight

(3) **RSBC**: cost of the difference in number of rolling stock units at each station after rescheduling compared to the planned data. At the rescheduling end time, a certain number of rolling stock units is required at each station to guarantee that operations can return to the normal level as planned in the timetable. However, rolling stock units may end at a different station due to rescheduling, which further may lead to the number of rolling stock units being different at each

station compared to the planned data. Thus, the difference between the real number of rolling stock units and their planned number at each station is penalised. It is measured by the absolute difference of planned and real number of rolling stock units at a station and its weight

(4) **SOC**: shunting operation cost that represents the cost of uncoupling and coupling rolling stock units from and to rolling stock units. It is measured by the shunting operation times and its weight

(5) **TCC**: penalty for task cancellations due to unavailable drivers or rolling stock units. The cancellation cost is set to be equal to the product of task duration and its weight

(6) **TRC**: penalty for task retiming impact due to drivers or rolling stock not being able to cover a task unless it is retimed. It is measured by the retiming duration and its weight

(7) **SRC**: cost of the number of utilised spare drivers and rolling stock units. It is measured by the time of a spare resource work and its weight

Table 19 shows how each factor is measured. In the experimental tests, the weights w1, w2, w3, w4, w5 and w6 can vary. However, the costs of the number of utilised resources and shunting operations should be high, thus w1 and w3 should be big. The reason for this is that unnecessary resources or shunting operations should be avoided in a solution whenever possible. Also, these indicators are measured by numerical values unlike the costs for task cancellation, retiming and using a spare driver, which are measured in minutes and hence should be low (50, 30 and 10, respectively). In the further discussed approach to solving IRSCRR, two multicriteria techniques are used to obtain multiple solutions. The results show that setting different weights for the factors does not always affect the solutions. The numbers in the brackets are the parameter values used in the study.

Table 19 Cost indicators and their measurements in the model

No	Indicator	Measurement
1	recovery diagram cost (RDC)	the sum of connection costs
2	number of used drivers and rolling stock that are not directly affected by disruption (NRC)	number * w1 (5000)
3	rolling stock unit balance (RSBC)	balance difference * w2 (50)
4	shunting operations (SOC)	number * w3 (5000)
5	task cancellation cost (TCC)	task duration * w4 (50)
6	task retiming cost (TRC)	delay time * w5 (30)
7	spare driver cost (SRC)	working time * w6 (10)

5.2.4. Model

$$\min f(x, y, g, h, v) \qquad \text{IRSCRR}$$

$$\text{s.t. } (5.1) - (5.13)$$

Overall, the model IRSCRR aims to find an integrated rolling stock and crew rescheduling solution subject to constraints (5.1) to (5.13) which achieves a minimum for $f(x, y, g, h, v)$. The definition of $f(x, y, g, h, v)$ will be given in Sections 5.4 and 5.5.

5.3.Method: 2SMO

A 2-stage approach using multicriteria decision making (2SMO) is used to solve IRSCRR. The input information includes a revised timetable (beyond the scope of this work), the planned initial rolling stock unit and crew diagrams, and rescheduling start and end time. The revised timetable and rescheduling start and end times are treated as given. The rescheduling result is that each affected driver and rolling stock unit considered in the model is assigned a recovery diagram. 2SMO contains a feedback mechanism which allows the solution process to generate retiming possibilities of tasks to find solutions. Before a detailed explanation of 2SMO is given, the following terms are first explained.

5.3.1. Disruption Neighbourhood

With the given input, a disruption neighbourhood $\mathcal{A}$ can be created for the integrated rolling stock and crew rescheduling problem. Disruption neighbourhood $\mathcal{A}$ consists of a set of drivers D, a set of rolling stock units S and the tasks T that are in their planned diagrams between the rescheduling start time and end time, $t_{schedule\_start}$ to $t_{schedule\_end}$.

Building an Initial Disruption Neighbourhood

When a disruption happens, it is known which trains are delayed or cancelled directly due to the disruption. The set of such trains is denoted as $\varUpsilon$. For each train $\gamma \in \varUpsilon$, the initially affected rolling stock units can be found. The set of all such rolling stock units for all trains is denoted as S_0. For rolling stock unit $s \in S_0$, drivers who drive rolling stock unit s between the rescheduling start time and end time can be found. Such drivers are the initially affected drivers; their set is denoted as D_0.

Drivers in D_0 may also drive trains using rolling stock not in S_0 during a disruption period. The set of such rolling stock units is denoted as S_1. Similarly, rolling stock units in S_1 could also have other drivers who work on them during the disruption period but are not included in D_0. This iterative approach is used to find all drivers and rolling stock units to build the initial disruption neighbourhood $\mathcal{A}$. It includes drivers, rolling stock units and tasks that are assigned to these resources during rescheduling, $\mathcal{A} = \{S, D, T\}$.

Extending a Disruption Neighbourhood

A disruption neighbourhood can be extended by adding retiming copies of tasks and spare resources. The reason that spare resources are not added in the initial disruption neighbourhood is that a list of tasks is expected to be obtained, which cannot be covered without using spare resources. Retiming copies can be produced just for these uncovered tasks in the second stage of the approach. If spare resources are used in the first stage, the need to use the retiming option is more minor.

5.3.2. Task Coverage Feedback Mechanism

One type of feedback information generated from the model is how much a task should be retimed so that a resource can cover it. Between two tasks, a minimum connection time is required for a resource to change from one task to another. For example, this connection time can be used to move to another platform for a driver and perform turnaround activities at a station for a rolling stock unit. If a resource finishes a previous task too late and cannot take the following task with the minimum connection time requirement, it can lead to the next task being uncovered. Such failure to cover the next task is recorded in a log file.

When a task is not covered in the first stage, in the feedback analysis step, the log file is analysed. A retiming possibility which requires the least delay time that is also within the parameter value of the maximum delay time is chosen and a retiming copy is created for this uncovered task and added to the disruption neighbourhood. A variable representing this retiming copy is added to the model. A retiming copy is only considered for tasks that are not commencing or terminating activities in a disruption neighbourhood. To maximally allow feasible solutions to appear, terminating activities are shifted back to the extent that will not affect operations beyond the recovery period in the data processing phase.

5.3.3. Graph for Solving Rolling Stock SPPRC

Algorithm 14 Determine the type of arc between task1 and task2 for a rolling stock unit

```
Input: task1,task2
1:  if task1 and task2 are carried out by the same rolling stock unit in planned
    data then
2:    if task1 and task2 can still be connected as in this rolling stock unit's
      planned diagram then
3:      return oldArc
4:    end if
5:  end if
6:  if task1 arrives at the station where task2 starts then
7:    if task1 and task2 are on different trains and time difference $\geq$ change
      train time then
8:      return changeArc
9:    else
10:     return null
11:   end if
12:   if task1 and task2 are on the same trains and time difference $\geq$ same
      train time then
13:     return immArc
14:   else
15:     return null
16:   end if
17: end if
Output: arc type
```

Algorithm 14 Determining the arc type between task1 and task2 for a rolling stock unit

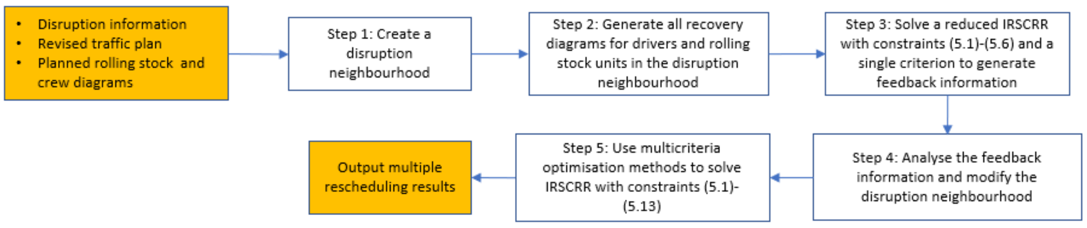
The graph created for solving a SPPRC problem for rolling stock is shown in Algorithm 14. It is simpler than building a graph for solving a SPPRC for a driver (see Algorithm 4 - Algorithm 6). Only three arcs are considered: “oldArc”, “changeArc” and “immArc”. If two tasks can be connected as in the planned data, “oldArc” is used in step 3. A possible arc can only be considered if the previous task ends at the same station as next task departs. If two tasks belong to the same train and the time difference is satisfied, an “immArc” is used. If two tasks belong to different trains and the time difference is enough for rolling stock to change trains, a “changeArc” is used. The costs of these connection types are the same as listed in Table 12 in Section 4.4.2.

Compared to the SPPRC graph built for crew, meal break, overtime and passenger trip are not considered. Rolling stock requires maintenance, which in planning diagrams can be treated similarly to including meal breaks. In practice, there is a specific plan for rolling stock maintenance, which is not considered in the model. Overtime is implied in Algorithm 14 since an “immArc” or “changeArc” connection could require rolling stock to work longer time than planned in their diagrams. Passenger trips are not considered for rolling stock because it does not make sense to relocate rolling stock by taking passenger trips.

One difference between generating a recovery diagram for drivers and rolling stock units is that drivers need to sign off at their planned depots. However, rolling stock is, to a large extent, interchangeable. It is quite common for two rolling stock units to swap the remaining tasks of a day following disruption. For example, when a blockage happens, trains on one side of the blockage short turn before approaching the blockage to run the train services that are planned to run by trains on the other side of the blockage. For rolling stock units of these short turn trains, they swap the remaining tasks with each other. So, a recovery diagram for a driver should end with the task which the driver was initially scheduled to perform at the rescheduling end time. However, a recovery diagram for a rolling stock unit can end at any task that any rolling stock unit from a disruption neighbourhood should perform at the rescheduling end time. A penalty is added to a recovery diagram for a rolling stock unit if it ends at a task which is not the task initially planned at the rescheduling end time. A SPPRC is solved for each rolling stock unit as explained in Section 4.4.3.

5.3.4. Overview of 2SMO

The framework of 2SMO is shown in Figure 26. In step 1, a disruption neighbourhood $\mathcal{A}$ can be built as explained in Section 5.3.1. In step 2, recovery diagrams for each rolling stock unit and driver are enumerated by solving SPPRC as explained in Section 4.4.3. In step 3, a reduced IRSCRR model that does not consider retiming possibilities is built and solved with a commercial solver. The single objective of this reduced model is to minimise $f(x, y, g, h) = RDC + NRC + RSBC + SOC + TCC$. The constraints are (5.1) – (5.6). During the solution process, a log file records the retiming requirements from a resource for a task to be covered. In step 4, the log file is automatically analysed. For each cancelled task obtained in step 3, if a retiming requirement is obtained from the log file, the retiming copies of that task will be generated and added to the disruption neighbourhood $\mathcal{A}$. Spare drivers and rolling stock units are also added to $\mathcal{A}$ in this step. In step 5, a complete IRSCRR model with constraints (5.1) - (5.13) and criteria specified below in Section 5.4 and 5.5 is solved using multicriteria optimisation methods and a commercial solver. Steps 1 to 3 are grouped as stage one, and steps 4 and 5 are grouped as stage two.



Disruption information
Revised traffic plan
Planned rolling stock and crew diagrams
Step 1: Create a disruption neighbourhood
Step 2: Generate all recovery diagrams for drivers and rolling stock units in the disruption neighbourhood
Step 3: Solve a reduced IRSCRR with constraints (5.1)-(5.6) and a single criterion to generate feedback information
Step 4: Analyse the feedback information and modify the disruption neighbourhood
Step 5: Use multicriteria optimisation methods to solve IRSCRR with constraints (5.1)-(5.13)
Output multiple rescheduling results

Figure 26 2-stage multicriteria decision-making approach (2SMO)

IRSCRR and 2SMO are applied to single delay and multiple delay scenarios in the following two sections. Different objectives and multicriteria optimisation methods are used for single and multiple delay scenarios.

5.4. Experiments and Results: Single Delay Scenarios

In this section, a typical single primary delay rescheduling problem is first explained. Then the common rescheduling strategies used in practice are presented. Next, model IRSCRR and 2SMO are applied using a convex combination of criteria to solve a series of single delay problems. The goal is to see if the types of solutions used in practice can be obtained with model IRSCRR and method 2SMO. The dataset used in this Chapter to test the model and method is the same as in Chapter 4.

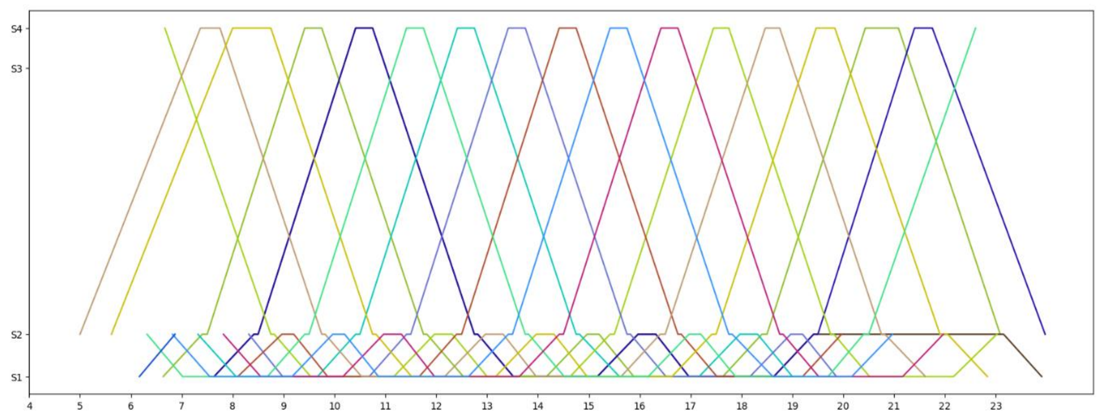


Figure 27 Rolling stock running on a single line

Figure 27 shows several rolling stock units running on a railway line between stations S1 and S4. The different colours represent different rolling stock units. Segments between S1 and S4 can be seen as individual train journeys. When a rolling stock unit arrives at station S4, it waits for a short time and runs the following service from S4 to S1. Since a rolling stock unit only stays at S4 for a short period, then, if there is a delay in running train service from S1 to S4, such a delay is easily propagated into the train services running from S4 to S1.

Figure 28 demonstrates a single delay scenario and three of the most frequently used rescheduling strategies by controllers from TOCs on the line between S1 and S4. The late train $Train\_A$ uses rolling stock $RS1$. $Train\_B$ is the immediate train after $Train\_A$ that uses the same rolling stock $RS1$. Assume $Train\_A$ is running late towards S2, there are three possible rescheduling strategies.

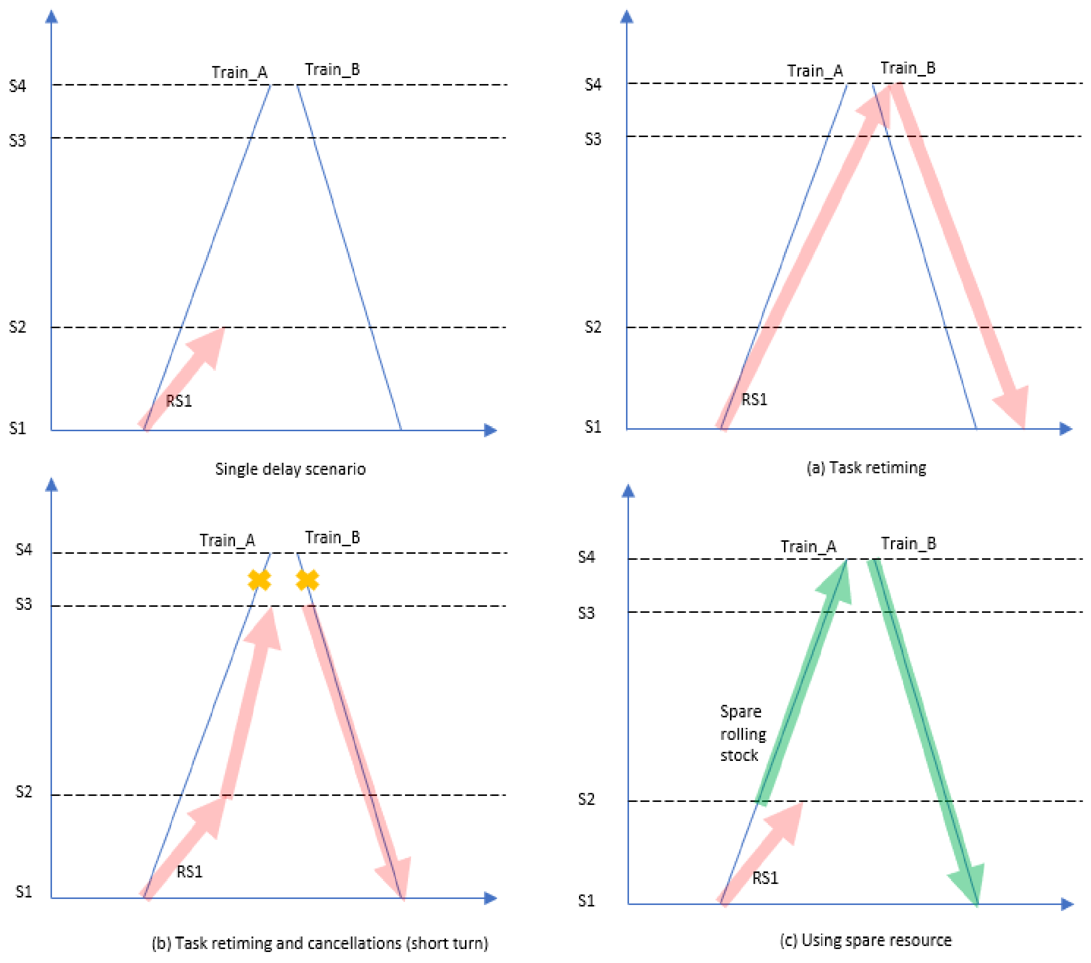


Figure 28 Three frequently used solutions with different characteristics

It is worth noting that Figure 28 is only for illustration. The gradients of each colored line in Figure 28 can vary. Solution (a) suggests running both $Train_A$ and $Train_B$ late. It does not require controllers to reschedule at all. The solution does not cancel any task or use spare rolling stock units. Solution (b) involves task retiming and cancellations. It suggests $RS1$ running late from S2 to S3 but short turning it at S3 so it can arrive at S2 to S1 on time. The train services between S3 and S4 are cancelled. Solution (b) does not use any spare rolling stock units. In solution (c), a spare rolling stock unit (marked in green) is used to run the service $Train\_A$ from S2 to S4 and $Train\_B$ from S4 to S1. $RS1$

arrives late at S2 and will become the new spare rolling stock unit. Comparing the three solutions (a), (b), and (c), it is hard to evaluate which one is better among them purely because of the solution characteristics. One solution may outperform another in different scenarios. For example, if bringing delay into a major station S2 will cause many trains to be delayed in S2, then solution (a) is not a good choice. When cancellation is not acceptable, then solution (b) is not a good choice. If there is no spare rolling stock to use, then solution (c) does not exist.

5.4.1. Using Convex Combination of Criteria

Suppose there is a multicriteria optimisation problem where the criteria are $Z_i(x)$ for $i = 1, \dots, K$ and S is the search space, and consider the problem (P_λ) defined by

$$\min \sum_{i=1}^{K} \lambda_i Z_i(x)$$
$$x \in S$$

where $\lambda_i \in (0,1)$ for $i = 1, \dots, k$ such that $\sum_{i=1}^{K} \lambda_i = 1$. We have the following theorem.

Theorem 4 (Vincent T'kindt, 2006) If $x^0 \in S$ is an optimal solution for (P_λ) then x^0 is a proper Pareto optimum.

Based on the three frequently used rescheduling strategies, three costs are especially important: task cancellation (TCC), task retiming (TRC) and using spare resources (SRC) which are explained in Section 5.2.3. The remaining insignificant cost factors in a single delay disruption scenario are combined as one objective $f_1(z)$. In total, there are four objectives:

$$f_1(z) = RDC + NRC + RSBC + SOC$$
$$f_2(z) = TCC$$
$$f_3(z) = TRC$$
$$f_4(z) = SRC$$

According to Theorem 4, one proper Pareto optimum for an IRSCRR can be obtained wi th multicriteria $f_i(z), i = 1, \dots 4$ by solving the problem (P_λ) with a convex combinatio n $f_i(z), i = 1, \dots 4$. A parametric analysis using λ is conducted. We denote $\Lambda = \{\lambda = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}\, \forall i, \lambda_i \in (0,1)$ and $\sum_i^4 \lambda_i = 1, \lambda_i \in \{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}\}$. A simple loop is used to enumerate all combinations of weights λ. The number of total combinations of weights is 84.

5.4.2. Experimental Tests

The purpose of experiments in this section is to test if the solutions of the model and method using a parametric analysis of λ can mimic the three frequently used solutions in practice shown in Figure 28 (a), (b) and (c). The basic idea consists of dividing Λ into N disjoint parts Λ_i such that $\Lambda = \bigcup_{i=1}^{N} \Lambda_i$, where in each Λ_i the optimal solution found by the algorithm is of a particular form seen in actual practice. It is expected that there are three regions Λ_1, Λ_2 and Λ_3 where the optimal solutions found by the algorithm are as shown in Figure 28 (a), (b) and (c), respectively, and region Λ_4 where the optimal solutions found by the algorithm are not of any of the forms described in Figure 28.

The parameters that are relative to crew are set with the same values as in Section 4.4.6. The connection time for rolling stock units to perform two tasks from two different trains is 5 minutes. If a rolling stock unit does not end at the same terminating task as planned in their diagram, 500 is added. The maximum time a task can be delayed is set as 30 minutes.

Example

Here is one scenario tested with the methodology shown in Figure 26. It is tested with 84 combinations of weights. The results are shown in Table 20. Column “objType” shows the weight combination for the four objectives. Column “objective” shows solution costs. diagramCost, cancelCost and retimeCost stand for recovery diagram costs, task cancellation costs and task retiming costs. spareStockCost stands for the cost for using spare rolling stock. The other costs mentioned in section 5.2.3, in particular the shunting operation costs, are 0 in this scenario.

Table 20 One scenario test with 84 combinations of weights

objType	objective	diagramCost	cancelCost	retimeCost	spareStockCost	category
0.1_0.1_0.1_0.7	375.5	35	2100	1620	0	(b)
0.1_0.1_0.2_0.6	537.5	35	2100	1620	0	(b)
0.1_0.1_0.3_0.5	699.5	35	2100	1620	0	(b)

0.1_0.1_0.4_0.4	861.5	35	2100	1620	0	(b)
0.1_0.1_0.5_0.3	678.5	35	0	0	2250	(c)
0.1_0.1_0.6_0.2	453.5	35	0	0	2250	(c)
0.1_0.1_0.7_0.1	228.5	35	0	0	2250	(c)
0.1_0.2_0.1_0.6	529.5	75	0	5220	0	(a)
0.1_0.2_0.2_0.5	747.5	35	2100	1620	0	(b)
0.1_0.2_0.3_0.4	903.5	35	0	0	2250	(c)
0.1_0.2_0.4_0.3	678.5	35	0	0	2250	(c)
0.1_0.2_0.5_0.2	453.5	35	0	0	2250	(c)
0.1_0.2_0.6_0.1	228.5	35	0	0	2250	(c)
0.1_0.3_0.1_0.5	529.5	75	0	5220	0	(a)
0.1_0.3_0.2_0.4	903.5	35	0	0	2250	(c)
0.1_0.3_0.3_0.3	678.5	35	0	0	2250	(c)
0.1_0.3_0.4_0.2	453.5	35	0	0	2250	(c)
0.1_0.3_0.5_0.1	228.5	35	0	0	2250	(c)
0.1_0.4_0.1_0.4	529.5	75	0	5220	0	(a)

0.1_0.4_0.2_0.3	678.5	35	0	0	2250	(c)
0.1_0.4_0.3_0.2	453.5	35	0	0	2250	(c)
0.1_0.4_0.4_0.1	228.5	35	0	0	2250	(c)
0.1_0.5_0.1_0.3	529.5	75	0	5220	0	(a)
0.1_0.5_0.2_0.2	453.5	35	0	0	2250	(c)
0.1_0.5_0.3_0.1	228.5	35	0	0	2250	(c)
0.1_0.6_0.1_0.2	453.5	35	0	0	2250	(c)
0.1_0.6_0.2_0.1	228.5	35	0	0	2250	(c)
0.1_0.7_0.1_0.1	228.5	35	0	0	2250	(c)
0.2_0.1_0.1_0.6	379	35	2100	1620	0	(b)
0.2_0.1_0.2_0.5	541	35	2100	1620	0	(b)
0.2_0.1_0.3_0.4	703	35	2100	1620	0	(b)
0.2_0.1_0.4_0.3	682	35	0	0	2250	(c)
0.2_0.1_0.5_0.2	457	35	0	0	2250	(c)
0.2_0.1_0.6_0.1	232	35	0	0	2250	(c)
0.2_0.2_0.1_0.5	537	75	0	5220	0	(a)

0.2_0.2_0.2_0.4	751	35	2100	1620	0	(b)
0.2_0.2_0.3_0.3	682	35	0	0	2250	(c)
0.2_0.2_0.4_0.2	457	35	0	0	2250	(c)
0.2_0.2_0.5_0.1	232	35	0	0	2250	(c)
0.2_0.3_0.1_0.4	537	75	0	5220	0	(a)
0.2_0.3_0.2_0.3	682	35	0	0	2250	(c)
0.2_0.3_0.3_0.2	457	35	0	0	2250	(c)
0.2_0.3_0.4_0.1	232	35	0	0	2250	(c)
0.2_0.4_0.1_0.3	537	75	0	5220	0	(a)
0.2_0.4_0.2_0.2	457	35	0	0	2250	(c)
0.2_0.4_0.3_0.1	232	35	0	0	2250	(c)
0.2_0.5_0.1_0.2	457	35	0	0	2250	(c)
0.2_0.5_0.2_0.1	232	35	0	0	2250	(c)
0.2_0.6_0.1_0.1	232	35	0	0	2250	(c)
0.3_0.1_0.1_0.5	382.5	35	2100	1620	0	(b)
0.3_0.1_0.2_0.4	544.5	35	2100	1620	0	(b)

0.3_0.1_0.3_0.3	685.5	35	0	0	2250	(c)
0.3_0.1_0.4_0.2	460.5	35	0	0	2250	(c)
0.3_0.1_0.5_0.1	235.5	35	0	0	2250	(c)
0.3_0.2_0.1_0.4	544.5	75	0	5220	0	(a)
0.3_0.2_0.2_0.3	685.5	35	0	0	2250	(c)
0.3_0.2_0.3_0.2	460.5	35	0	0	2250	(c)
0.3_0.2_0.4_0.1	235.5	35	0	0	2250	(c)
0.3_0.3_0.1_0.3	544.5	75	0	5220	0	(a)
0.3_0.3_0.2_0.2	460.5	35	0	0	2250	(c)
0.3_0.3_0.3_0.1	235.5	35	0	0	2250	(c)
0.3_0.4_0.1_0.2	460.5	35	0	0	2250	(c)
0.3_0.4_0.2_0.1	235.5	35	0	0	2250	(c)
0.3_0.5_0.1_0.1	235.5	35	0	0	2250	(c)
0.4_0.1_0.1_0.4	386	35	2100	1620	0	(b)
0.4_0.1_0.2_0.3	548	35	2100	1620	0	(b)
0.4_0.1_0.3_0.2	464	35	0	0	2250	(c)

0.4_0.1_0.4_0.1	239	35	0	0	2250	(c)
0.4_0.2_0.1_0.3	552	75	0	5220	0	(a)
0.4_0.2_0.2_0.2	464	35	0	0	2250	(c)
0.4_0.2_0.3_0.1	239	35	0	0	2250	(c)
0.4_0.3_0.1_0.2	464	35	0	0	2250	(c)
0.4_0.3_0.2_0.1	239	35	0	0	2250	(c)
0.4_0.4_0.1_0.1	239	35	0	0	2250	(c)
0.5_0.1_0.1_0.3	389.5	35	2100	1620	0	(b)
0.5_0.1_0.2_0.2	467.5	35	0	0	2250	(c)
0.5_0.1_0.3_0.1	242.5	35	0	0	2250	(c)
0.5_0.2_0.1_0.2	467.5	35	0	0	2250	(c)
0.5_0.2_0.2_0.1	242.5	35	0	0	2250	(c)
0.5_0.3_0.1_0.1	242.5	35	0	0	2250	(c)
0.6_0.1_0.1_0.2	393	35	2100	1620	0	(b)
0.6_0.1_0.2_0.1	246	35	0	0	2250	(c)
0.6_0.2_0.1_0.1	246	35	0	0	2250	(c)

0.7_0.1_0.1_0.1	249.5	35	0	0	2250	(c)

The solutions in Table 20 are divided into three groups: (a) using retiming, (b) using cancellations and retiming and (c) using spare rolling stock, which corresponds to the three types of solutions shown in Figure 28. The column "category" shows the category of each solution. All solutions found in each category are actually the same. The total objective is different only because λ_is are different, as the costs of all factors remain the same. The number of λ_is used to obtain solutions which belong to each category is summarised in Table 21.

Table 21 Number of weights used to obtain solutions belonging to each category

Type	Number of λ_is
(1) using retiming (a)	15
(2) using cancellations and retiming (b)	59
(3) using spare rolling stock (c)	10

Multiple Tests

The model is further tested on one line where a single delay happens frequently. The task which is frequently delayed is scheduled once an hour from 6 am to 6 pm on any operational day. Together there are 13 scenarios. All 13 scenarios involve one train being delayed by 10 minutes. Each scenario is listed as a row in Table 22. Each scenario is assigned an ID as "SD_t", where t stands for the disruption occurrence time. For each scenario, the numbers of drivers, rolling stock units, and tasks in the disruption neighbourhood are given. Feasible recovery diagrams are enumerated for all rolling stock units, and drivers and their numbers are given. For each scenario, the total solution time for all 84 combinations of weights is given, and the average time per one combination of weights is also given.

Table 22 Single delay scenarios for testing and results

Scenario	Number of drivers	Number of RS	Number of tasks	Diagrams for drivers	Diagrams for RS units	Solution time (s)	Average solution time (s)
SD_6	127	74	291	63,082	1,786	1303	15.33
SD_7	138	75	293	230,823	6,635	56865	669.00
SD_8	154	72	288	55,009	2,262	1313	15.45
SD_9	136	56	216	21,811	1,137	262	3.08

SD_10	145	61	244	32,712	1,731	309	3.64
SD_11	127	54	226	58,613	2,622	661	7.78
SD_12	129	55	223	90,866	2,624	1153	13.56
SD_13	149	66	265	105,711	1,692	3855	45.35
SD_14	150	72	301	163,208	2,277	2993	35.21
SD_15	134	71	299	161,844	1,976	2453	28.86
SD_16	123	68	282	166,175	1,393	22113	260.15
SD_17	149	66	265	105,711	1,692	3855	45.35
SD_18	91	60	232	85,594	989	1419	16.69

In Table 22, the number of tasks considered in the model has its peak level for scenarios happening at 6-8 am and 2-5 pm. The number of driver recovery diagrams in the model fluctuates mainly from around 21 thousand to 230 thousand of diagrams. Compared to the number of driver diagrams, the number of rolling stock diagrams in the model is much smaller, from a few hundred to around seven thousand diagrams. The solution time is significant for some scenarios. The total solution time with 84 combinations of weights for scenario that happens at 7 am (SD_07) is almost 16 hours. For some scenarios, the total solution time only takes less than 5 minutes (SD_09). The solution time strongly depends on the total number of diagrams considered in the model.

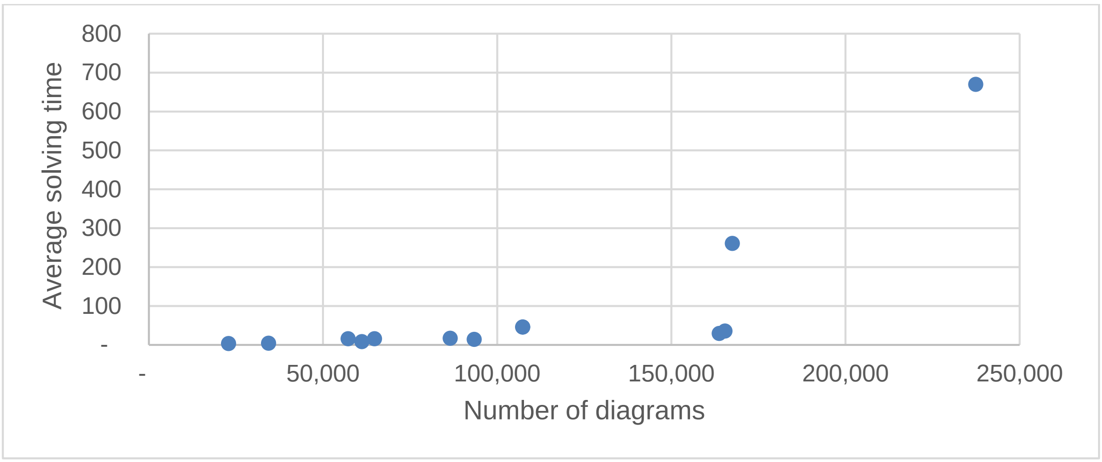


Figure 29 Solution time with respect to the number of diagrams

The relationship between the average solution time and the total number of diagrams is shown in Figure 29. The average solution time is low (below 100 seconds) when the number of diagrams is below 150 thousand. Then it sharply rises to around 260 seconds for 167 thousand diagrams and 669 seconds for 237 thousand diagrams. Not all points in Figure *29* show a rising relationship between the average solution time and the total number of diagrams. The reason is that the average solution time is also affected by the number of drivers, rolling stock and tasks. Overall, the solution time exhibits exponential growth.

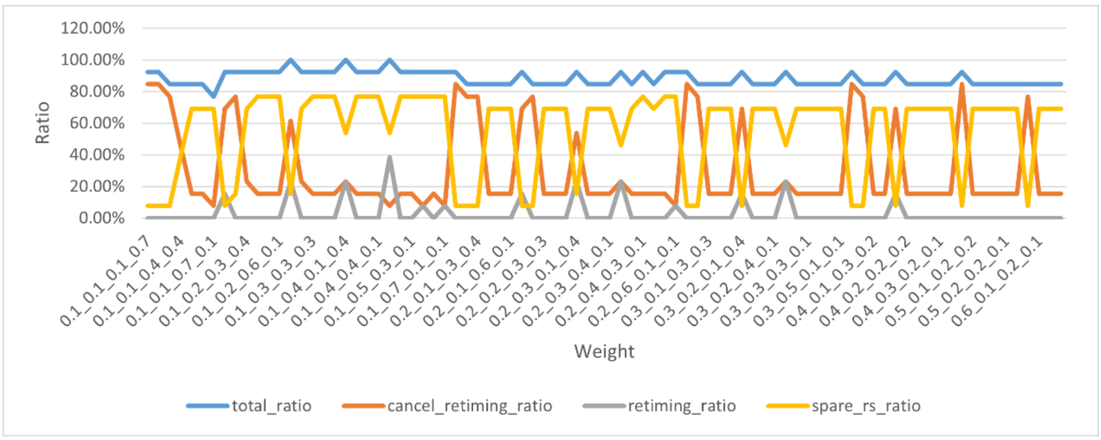


Figure 30 The performance of 84 weights on 13 tests for a single delay scenario

From Figure 30, for each weight, most of the optimal solutions from all 13 scenarios fall into the following three categories: task cancellation and retiming, task retiming or using spare stock resources. To be specific, 88.10% of optimal solutions belong to one of these three categories. Thus, Λ can be divided into four parts Λ_i such that $\Lambda = \bigcup_{i=1}^{4} \Lambda_i$. Among them, the first three parts produce optimal solutions as displayed in Figure 28 a), b) and c).

It is noticeable that solutions that only use retiming (grey line) have low ratio compared to the other two categories (orange and yellow lines). The reason for this is that retiming tasks cannot always be a feasible solution no matter how weights are set. It may be infeasible because the drivers who are on the task need to have a meal break after the task. So in this case the drivers can have their meal breaks on time without violating fatigue rules only if the task is not delayed. The yellow line (using spare rolling stock) and orange line (using cancellation and retiming) are clearly distinguished for different combinations of weights. When the ratio for solutions using spare rolling stock and retiming is high, the ratio for using cancellation is low. To be specific, the combination of weights 0.1_0.5_0.1_0.3 has the highest ratio to produce optimal solutions only using retiming. A number of weight combinations (such as 0.1_0.3_0.5_0.1) can be used to produce optimal solutions using spare rolling stock in most of the scenarios. Some weights (such as 0.1_0.2_0.2_0.5) can be used to produce optimal solutions using cancellation and retiming in most of the scenarios.

5.5. Experiments and Results: Multiple Delay Scenarios

For various reasons multiple delays could happen and many drivers and rolling stock could be affected. The same convex combination of criteria and parametric analysis of λ has been used to solve a significant disruption scenario. However, the solution cannot be categorised clearly as in single delay scenarios. The reason for this is that, for significant disruption scenarios, delays happen on more than one route. The solutions shown in Figure 28 only consider one train delay on one route. Thus, another multicriteria optimisation method, ε -constraint approach, is used to solve scenarios with multiple delays.

5.5.1. Using ε -constraint Approach

The idea of the ε-constraint approach is to minimise one objective and add other objectives into constraints so other constraints are upper bounded with a parameter ε, which is known or can be decided by a user based on experience, or it can be modified by a user to analyse the impact of modifications on the final solutions.

In multiple delay scenarios, using spare resources is a default rescheduling strategy. Thus, it is not seen as a separate criterion. Here two costs are considered: one is the cost $f_1(z)$ that is the cost of train operators implementing any rescheduling strategies. Another is the cost $f_2(z)$ that represents the inconvenience caused to passengers, which is the sum of task cancellations and retiming.

$$f_1(z) = RDC + NRC + SRC + SOC + RSBC$$

$$f_2(z) = TCC + TRC$$

In the case study, $f_1(z)$ is minimised and $f_2(z)$ is added as a constraint which is upper bounded by ε. Thus, the following problem using an ε -constraint approach is defined. z is the variable vector, and S is the search space.

$$\begin{aligned} &\min f_1(z) \\ &\text{s.t. } z \in S \\ &f_2(z) \leq \varepsilon \end{aligned}$$

5.5.2. Experimental Tests

The purpose of experiments in this section is to test if the model and methods can give solutions that form a Pareto frontier, that is, cancellation and retiming cost cannot be decreased unless there is an increase in rescheduling cost. The parameter settings are the same as in Section 5.4.2. The upper bound of the objective $f_2(z)$ is the worst cancellation and retiming cost. The highest cancellation and retiming costs occur when controllers make no rescheduling decisions and allow delay or cancellation to propagate. Because, in this case, no active remedy is implemented. Thus, this cancellation and retiming cost can be used as the upper bound. To get this upper bound, the methodology from Section 5.3.4 is slightly modified; the new methodology is shown in Figure 31.

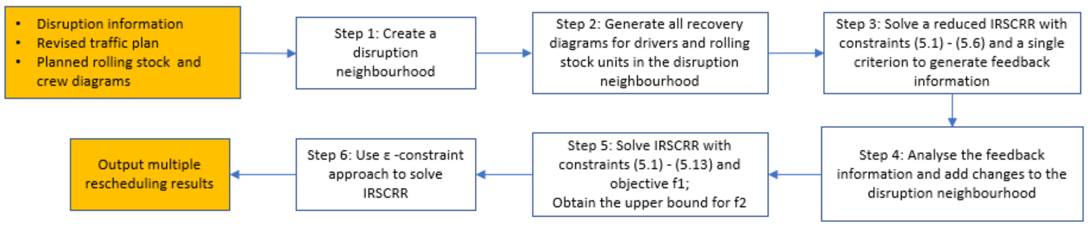

Disruption information
Revised traffic plan
Planned rolling stock and crew diagrams
Step 1: Create a disruption neighbourhood
Step 2: Generate all recovery diagrams for drivers and rolling stock units in the disruption neighbourhood
Step 3: Solve a reduced IRSCRR with constraints (5.1) - (5.6) and a single criterion to generate feedback information
Step 4: Analyse the feedback information and add changes to the disruption neighbourhood
Step 5: Solve IRSCRR with constraints (5.1) - (5.13) and objective f1; Obtain the upper bound for f2
Step 6: Use ε -constraint approach to solve IRSCRR
Output multiple rescheduling results

Figure 31 2-stage multicriteria decision-making approach (2SMO) for multiple delay scenarios

In step 5, IRSCRR is solved with a single criterion, f1. It means minimising the rescheduling operations without considering the cost of task cancellations or retiming. In this case, the tasks that are cancelled and retimed with no rescheduling are obtained. Then the worst task cancellation and retiming cost can be calculated, which is the upper bound ε. To obtain the Pareto front for each scenario, in step 6, ε is gradually decreased by 10% to obtain a set of optimal solutions.

IRSCRR and 2SMO are applied to 12 instances which differ from each other by the disruption occurrence time. The results are shown in Table 23. In each scenario, it is assumed that trains that should depart from a major station between t and $t + 20$ minutes are delayed by 20 minutes due to failed infrastructure, where t is the disruption occurrence time set as 7 am, 8am, …, or 6 pm. The rescheduling starts at t and the recovery period is set as 4 hours. The numbers of drivers and rolling stock units are found as described in Section 5.3.1.

Each row in Table 23 represents a scenario. Each scenario is assigned an ID as "MD_t", where t is the disruption occurrence time. Table 23 shows the number of directly affected trains, drivers, rolling stock units, and solution time. 4 trains are directly affected for all scenarios except scenario MD_16 and MD_17. The number of drivers varies from 113 to 161 and the number of rolling stock units varies from 66 to 77. The number of tasks is at its highest for MD_15 and MD_16.

In the method, ε is gradually decreased by 10% to obtain a set of optimal solutions. Thus, the model is solved 10 times for different upper bounds ε . The solution time fluctuates from 177 seconds for scenario MD_11 to 839 seconds for scenario MD_17. The average solution time for one ε is 1 minute. The number of optimal solutions for one scenario varies from 3 to 7. For scenarios MD_7 - MD_09 and MD_12, 3 distinguished solutions are found. For scenarios MD_13 and MD_17, 7 solutions are found.

Table 23 Multiple delay scenarios for testing and results

Scenario					Result	
Instance ID	Directly affected trains	Number of drivers	Number of rolling stock units	Number of tasks	Number of solutions	Solution time (s)

MD_7	4	137	74	284	3	450
MD_8	4	148	76	286	3	250
MD_9	4	161	74	277	3	344
MD_10	4	156	68	271	6	246
MD_11	4	150	66	254	4	177
MD_12	4	157	75	265	3	189
MD_13	4	151	68	264	7	581
MD_14	4	158	75	293	4	330
MD_15	4	156	77	309	5	270
MD_16	5	137	77	308	4	502
MD_17	5	126	77	296	7	839
MD_18	4	113	74	281	5	418

Figure 32 shows the objectives changes for 12 scenarios. The vertical axis shows f_2 (task cancellation and retiming) and the upper bounds for f_2. The horizontal axis shows the objective value f_1 (the total cost of using spare resources, using resources that take new tasks, shunting operations, rolling stock units differences and recovery diagrams). Figure 32 shows a trade-off between objectives f_1 and f_2 in each instance, which matches the expectations. At first f_2 (blue line) has a large value, which means that many tasks are either cancelled or delayed and no rescheduling moves are implemented. With a descending upper bound (red line) on f_2, it starts to drop and f_1 starts to rise, which means with more rescheduling moves, the cancellation and delay costs drop.

The upper bounds drop by 10% at each step, so in each subgraph in Figure 32, only 10 upper bounds and their corresponding solutions are shown. However, it is clear that when upper bound ε decreases, sometimes the same solution is obtained. For some scenarios (MD_13 - MD_18), when upper bounds drop at a certain level, there will be no feasible solutions under a small upper bound. For some scenarios, MD_07 - MD_12, the cost of task cancellations and retiming can be reduced to a low point or to even 0 in some cases, However, for other scenarios, the cost of task cancellations and retiming remains high no matter the change of rescheduling effort. The reason for this is that some rolling stock units have a heavily loaded recovery diagram. The shift back time of retiming tasks will cause the rolling stock's terminating task to be delayed more than allowed. There are steep falls appearing in f_2 for scenarios MD_07, MD_13 and MD_15 - MD_18. It means that with a small cost rising in f_1 , cancellations and retiming costs can be largely reduced. The reason is that using resources on different tasks can lead to a big reduction in f_2 but with a slight rise in recovery diagram costs or using spare resource cost.

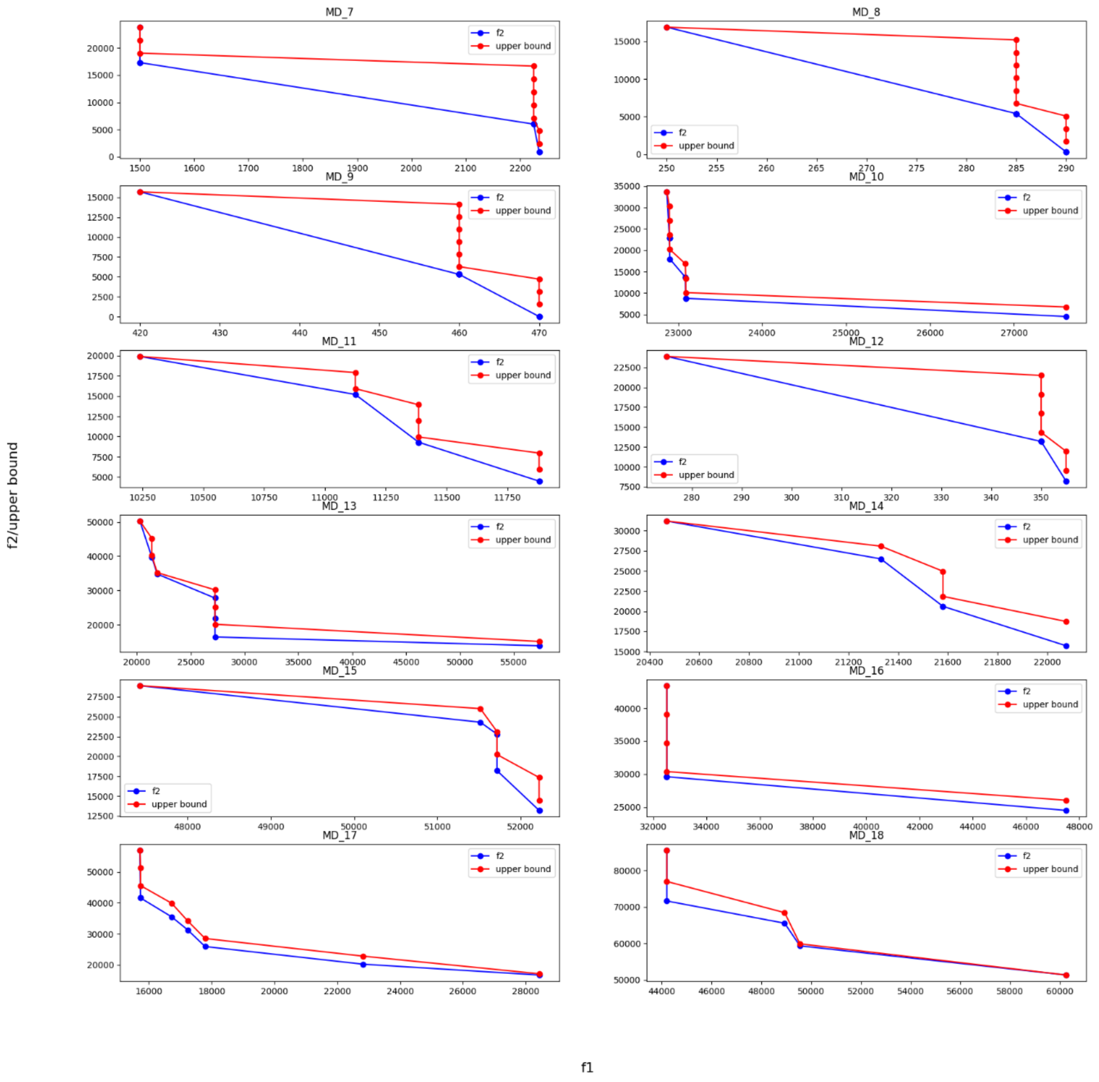


Figure 32 A set of optimal solutions for 12 multiple delay scenarios

5.6. Conclusion

In this chapter, the integrated rolling stock and crew rescheduling problem with retiming possibilities (IRSCRR) was considered and the problem is modelled using integer linear programming. The detailed constraints about crew rescheduling, rolling stock rescheduling and their cross-reference constraints were presented. Constraints about using task retiming possibilities and retiming propagation are described. 7 different factors are considered in the objective: cost of chosen recovery diagrams, number of utilised resources, difference in rolling stock balance, the using of shunting operations, task cancellations, task retiming and the using of spare resources. A customized 2 stage framework using multicriteria optimisation methods (2SMO) for solving the integrated problem was developed. 2SMO has a feedback mechanism which can suggest retiming possibilities of task that are used in the solution process automatically. The feedback mechanism can give direction to which tasks should be retimed and by how much, which can largely decrease problem size. Also, 7 cost factors were analysed and grouped to construct multiple objectives. The integrated rolling stock and crew rescheduling problem was solved as a multicriteria optimisation problem. Two typical multicriteria optimisation methods were used: Convex Combination of Criteria and ε-constraint approach.

IRSCRR and 2SMO have been tested on several single delay scenarios and multiple delay scenarios successfully. For single delay scenarios, the method successfully tested all scenarios with 84 combinations of weights. The analysis on 84 combinations of weights was conducted. The relationship between method speed and the number of recovery diagrams used in a model was described using a figure. 84 combinations of weights can be categorised into three groups to produce the three most used rescheduling strategies in practice.

For multiple delay scenarios, two objectives were considered. One is rescheduling effort (cost of chosen recovery diagrams, the number of utilised resources, the difference in rolling stock balance, the using of shunting operations and the using of spare resources) and another is task cancellation and retiming. The model and method are applied to a number of multiple delay scenarios. For each scenario, the model and method can give an efficient frontier, which is a set of optimal solutions. Also, the efficient frontier shows the trade-off relationship between rescheduling efforts and cancellation and delay costs. The solution quality has been analysed. It shows that sometimes with a slight rise in rescheduling efforts, task cancellation and retiming can drop a lot. Thus, there is significant value in reducing penalties for train cancellation and delay using the integrated method to reschedule rolling stock and crew together which might be too complex for manual rescheduling.

There are some limitations of the model and method. First, the model does not guarantee that the operations can return to its normal level. The reason is that after rescheduling, the number of rolling stock units at each station may not match the required number. Thus, the model and method can provide a rescheduling solution for crew and rolling stock for the recovery period, say 4 hours. It does not take care of railway operations after beyond the recovery period. Second, the model does not consider the maintenance requirements of some rolling stock. During the rescheduling process, rolling stock are treated as interchangeable and assigned different tasks compared to their planned diagrams. Some tasks may require the rolling stock to work overtime. However, in reality, the same rolling stock may need to head to a maintenance depot after some time. Third, the rolling stock and crew are rescheduled together. Thus, the model assumes that the type of rolling stock that can be used for a task is known in advance. Then only a driver with knowledge of this type of rolling stock can be assigned to the task. Fourth, the method 2SMO requires enumerating all recovery diagrams for rolling stock and crew considered in a model, which could take a long time. In Table 22, the running time could be as high as 669 seconds for one test. Thus, the method may not be suitable for an extensive network with many resources (rolling stock and crew) and a long recovery time in real time disruption management. However, the integrated model and method can be used in preparing rolling stock and crew contingency plans beforehand. A set of typical scenarios and corresponding timetable should be prepared. For each scenario, multiple solutions for rescheduling rolling stock and crew are expected to be produced. A limited set of solutions can be chosen and further documented into contingency plans which can be used as a starting point for rescheduling in real time.

CHAPTER SIX: CONCLUSION

This thesis has tackled the crew rescheduling problem for railway disruption management. It analyses the difficulties of crew rescheduling due to a range of factors: problem size, various constraints, retiming, robust solutions and evaluating solutions. The main purpose of this PhD project is to develop optimisation models and methods for crew rescheduling with the intention that these models and methods can be used to build a reliable decision support system to assist crew controllers in real practice.

To understand the background of passenger railway operation, one needs to recall (as in Chapter 2 of this thesis) that the two most important railway parties are the Infrastructure Manager and Train Operating Companies. These two parties perform the three important railway planning activities: timetable planning, rolling stock planning and crew planning. Railway disruption management is covered in Chapter 2, where it is also explained how crew rescheduling fits into the overall railway disruption management process. Disruptions are categorised. Two categories of disruptions are of particular interest to this thesis: minor disruptions and significant disruptions.

The new model and method called DFSCR for solving the crew rescheduling problem for minor disruptions has been presented in Chapter 3. DFSCR takes one uncovered task as input and typically outputs multiple conflict-free rescheduling solutions. It has been shown that DFSCR can be applied to scenarios that can happen in everyday practice. Multiple solutions are found for these scenarios which can be directly used in practice to ease the rescheduling pressure on controllers by providing significant decision support. The impact of five important parameters in DFSCR: rescheduling start time, taxi time variation, maximum work overtime, connection time and maximum work length on solution qualities and quantities has been analysed and summarised. It has been found that DFSCR solves 150 out of 382 considered scenarios with an average of 4.48 solutions in 0.75 seconds. Other scenarios could not be solved mainly because the input uncovered tasks of these tests happen very early or late at a station away from major stations as well as because of the parameter values set in the model. In reality, these problems may be solved based on different combinations of parameter values or if controllers could urgently find drivers who are not on duty at the time. A feedback mechanism is also used to improve the solvability by 6% and the average number of solutions by 1.13. The experimental results show that using the feedback mechanism can be a better approach than relaxing the maximum number of affected drivers used in DFSCR. Thus, using a feedback mechanism can assist controllers to use the optimisation tools more efficiently.

There is some possible further work relating to the model and method proposed in Chapter 3. First is to recognise the scale of disruption impact that can be efficiently solved by DFSCR and extract a list of uncovered tasks from the disruption. Controllers can review the common disruption scenarios, categorise them and try to extract uncovered tasks from typical minor disruption scenarios. Second, retiming is used in the method and further work is needed to address the follow-up conflicts. An automatic conflict detection method may need to be developed to examine a solution using retiming. Third, the impact of solving a list of uncovered tasks in different orders on overall rescheduling cost will require further testing on different scenarios and datasets. Fourth, the parameters used in the feedback mechanism may need to be further tuned for different TOCs.

A new timetable rescheduling model has been proposed (Chapter 4) to produce a revised timetable and predict the recovery period which are important inputs to solve the crew rescheduling problem for significant disruptions. The model can produce revised timetables which can be used to cover the three stages of the train recovery process and be used directly as input to reschedule rolling stock and crew. The model has been successfully tested on 14 scenario variations of a complete blockage on a double-track line, and the trade-off between train cancellation and delay is presented. Also, the impact of turnaround time, maximum allowed delay time and blockage occurring time on rescheduling solutions is presented and analysed. The same parameter settings may affect timetable rescheduling solutions differently since the timetable is non-cyclic.

A new model CRP and method LRCG for the crew rescheduling problem for significant disruptions has been also proposed (Chapter 4). It is explained how to construct and expand a disruption neighbourhood to limit problem size while producing good solutions. CRP is a path-based model, where a path represents a recovery diagram. Solving SPPRC to construct recovery diagrams to optimality is an NP-hard problem. To tackle this problem, three search methods: bi-directional, forwards and backwards methods are proposed to construct recovery diagrams in LRCG. LRCG can solve the crew rescheduling problem in a short time, from seconds to a few minutes, which is much faster than manual rescheduling by controllers. An important discovery is that there is a direct connection between solution time and iterations of column generation. Backwards search is the quickest search method and bi-directional search has advantages in finding solutions with smaller gaps between upper bounds and lower bounds, for which the reasons are given in Section 4.4.7. The backwards search is recommended to be used in LRCG to solve the crew rescheduling problem due to its speed advantage. One suggestion to the further work of Chapter 4 is that different kinds of significant disruptions should be tested, which requires a new timetable rescheduling model to produce revised timetables. The same crew rescheduling model CRP and method LRCG can be used to test different disruption scenarios. Further work is also required to reduce the possible large gaps between the lower bound and upper bound of solutions produced by the backwards search in LRCG.

The integrated rolling stock and crew rescheduling problem is explored in Chapter 5. Detailed formulations are given to model the integrated problem with retiming possibilities. Two kinds of delay scenarios are studied: single delay and multiple delay scenarios. The method for solving the integrated problem for single delay scenarios can replicate the three commonly used rescheduling strategies and 88.1% of the solutions in the test scenarios belong to these three rescheduling strategies. The method for solving the integrated problem for multiple delay scenarios can show the trade-off between operational cost on one hand and task cancellation and retiming on the other hand. The experimental results show that sometimes with a slight increase in operational cost, task cancellation and retiming can drop a lot, which provides evidence that using an optimisation tool based on the present work can be of great value to controllers. One potential improvement is the solution time of the integrated model, which is much higher than solving a single crew rescheduling model. Thus, this model and method may be suitable for creating rolling stock and crew contingency plans. Also, the model does not guarantee that the operations can return to their normal level. Thus, solving the integrated problem in a rolling time horizon may be required.

Careful consideration needs to be given to selecting the most appropriate tool once a disruption has been identified. The tool developed in Chapter 3 focuses on minor disruptions, and the tool developed in Chapter 4 focuses on significant disruptions. Minor disruption includes delays without specific incident occurring, often including disruption on certain lines of service groups that are self-contained to some extent. The service groups usually have limited interaction with other service groups. Significant disruption sees restricted access to parts of the network, affecting the ability for the timetable to be delivered. The optimisation tool developed in Chapter 5 is for the integrated rolling stock and crew rescheduling problem. The method developed in Chapter 3 may not be a good choice for significant disruptions if a list of a limited number of uncovered tasks cannot be extracted easily by controllers after a disruption. The methods developed in Chapters 4 and 5 can be used to solve a minor disruption scenario. Further tests are required to verify this. The set of affected drivers (rolling stock) due to a minor disruption and a recovery period should be defined before using the methods and the output can be interpreted as assigning a feasible recovery diagram to each affected resource.

For real time operations, this thesis provides optimisation tools and examples on how to reschedule crew for different levels of disruption. Currently, crew rescheduling is conducted manually in Great Britain and controllers are under huge pressure to find one feasible solution as soon as possible after disruption happens. Using the optimisation tools created in this thesis to develop a mature operational product, crew controllers could potentially find multiple optimised solutions in seconds. Better crew management during disruptions should lead to better railway performance and passenger experience. To successfully implement a crew rescheduling solution, immediate communication with train drivers is essential. In the current GB operational environment controllers should use various messaging tools like mobile phones or on-the-ground staff to meet train drivers at a station to communicate working changes. During the day, a crew management system should be kept up to date once changes to the current diagrams are decided so the system is always ready for the next disruption.

At the basic level, lessons learned are the tangible result of project review, breaking down the project experience of what went right, what went wrong and what could be done better. But these lessons will not amount to much if they are not integrated into the organisation's knowledge and used for continuous improvement. Continuous review and improvement is a key stage in enabling better crew rescheduling for disruptions.

TOCs should review the changes made to crew diagrams and the impact of changes on service recovery to evaluate the operational benefits and cost effectiveness of using the optimisation tools. The lessons learned should be briefed to all related parties, including controllers, drivers, and on-the ground support staff. The lessons learned in the continuous improvement should be used to adapt the crew contingency plans. A set of typical scenarios and good corresponding solutions should be documented. New rescheduling constraints found from lessons learned can be used to improve the optimisation tools. A trial session could be made to rerun the disruption scenarios and test the improved contingency plans and optimisation tools. To set up a trial session, the original scenario should be simulated; the corresponding database used for rescheduling is also required. Thus, the organisation should have a proper traffic management simulation system and back up the database used for rescheduling. All related controllers, drivers and on-the ground support staff should attend the trial session. With such trial sessions, the improvements in contingency plans or optimisation tools can be tested, discussed and validated by related parties. With knowledge and experience accumulating, the performance of crew rescheduling can be steadily improved, which should lead to better railway performance.

The optimisation tools developed in this thesis can be used not only in real time operations but also in preparing crew contingency plans beforehand. For different levels and types of disruptions, an appropriate crew rescheduling plan could be created and used to standardise the decision-making process concerning changes to crew diagrams. For a set of typical disruption scenarios, with the contingency plans for rescheduling timetable and rolling stock, the methods developed in this thesis can be used to create crew contingency plans using crew diagrams for a typical operational day. The number of crew contingency plans should be kept limited, so it does not become overwhelming for controllers to choose the most appropriate plan. Modifications to a plan are required if a scenario affects the network differently. It is also highly recommended that any contingency plans align with the organisations' other processes to ensure that there is no potential conflict. Also, it is recommended that the implementation process for crew contingency plans is learned and rehearsed by drivers during training so they can understand and accept the changes more effectively during disruptions.

Controller competency training and management is vital to use optimisation tools for crew rescheduling in disruption management. Some core competencies include: conducting a starting assessment of crew positions, using optimisation tools, communicating diagram alterations to relevant parties, requesting on-the-ground support to communicate with crew, updating the crew management system, recording disruption scenarios and rescheduling results, reviewing and continuously improving the rescheduling process. To use the optimisation tools developed from the methods in the present work in practice, there are some technical and non-technical skills required. Technical skills include using and understanding optimisation tools, interpretating crew rescheduling results and determining the changes to the crew diagrams, and using messaging in the control system. Non-technical skills include situational awareness that is, when to initiate the rescheduling process and use the optimisation tools, communicating with related parties about disruption and rescheduling decisions, working with others, decision making, etc. These skills should be looked for in industry recruitment and thoroughly trained during daily work.

When disruption occurs, passengers may become frustrated. How the industry deals with the disruption and how quickly the service can be recovered back to the normal timetable are the common concerns of passengers. However, how industry supports passengers during disruption can also be vital to their experience. It is clear that when passengers are given accurate and up-to-date information during disruption, they feel they have more power in understanding the situation and making the best decisions for their trips. Thus, besides minimising the disruption impact, a more active approach should be taken to give passengers with the right information using various channels, like social media or the front line interacting with passengers.

The models and methods proposed in the present work are ready to move to implementation in an optimisation tool and practical testing phase. In practical implementation, we expect that a processing step is required to process the raw data into the required data format that can be accepted by the methods, as well as that the outputs from the methods need to be processed and written into the operations system. A user interface needs to be designed to enable controllers to use the optimisation tool. An important consideration is that exchanging information between the methods and other systems like a database may slow down the solution process if the information flow is not reliable. Thus, minimising the information flow with other systems may be preferred. To verify and validate the optimisation tool, a detailed code inspection should be carried out to check for good programming style and functional quality. Frequently occurring scenarios and critical extreme scenarios should be tested and the results should be compared with the manual rescheduling results from controllers. A testing report should be generated by controllers using the optimisation tool in a simulated environment and reviewed by the development team to evaluate the optimisation tool. The development team needs to verify with the controllers if the aim of using the optimisation tool has been achieved, which is to ease rescheduling pressure on controllers and assist them to make better decisions.

References

Abbink, E. (2014). Crew Management in Passenger Rail Transport. PhD thesis, Erasmus University Rotterdam, the Netherlands.

American Public Transportation Association, A. (2019, 6 20). Compendium of Definition s and Acronyms for Rail Systems. Retrieved 1 24, 2023, from https://www.apta. com/wp-content/uploads/APTA-Compendium-of-Definitions-Acronyms-for-Rail -Systems.pdf

Arabeyre, J. P., Fearnley, J., Steiger, F. C., & Teather, W. (1969). The Airline Crew Scheduling Problem: A Survey. Transportation Science, 3(2), 140-163.

Banihashemi, M., & Haghani, A. (2001). A New Model for the Mass Transit Crew Scheduling Problem. In S. Voss, & J. R. Daduna (Eds.), Computer-Aided Scheduling of Public Transport (pp. 1-15). Berlin: Springer.

BCRRE, NetworkRail, RFI, & TV. (2014). Optimal Networks for Train Integration Management across Europe. Retrieved from Final Report Summary - ON-TIME (Optimal Networks for Train Integration Management across Europe): https://cordis.europa.eu/project/id/285243/reporting

Bessiere, C. (2006). Chapter 3: Constraint Propagation. In F. Rossi, P. Van Beek, & T. Walsh (Eds.), Handbook of Constraint Programming (Vol. 2, pp. 29-83). Amsterdam: Elsevier.

Budai, G., Maróti, G., Dekker, R., Huisman, D., & Kroon, L. (2010). Rescheduling in passenger railways: the rolling stock rebalancing problem. Journal of Scheduling, 13, 281-297.

Cacchiani, V. (2008). Models and Algorithms for Combinatorial Optimization Problems arising in Railway Applications. PhD thesis, Universita Degli Studi Di Bologna, Italy.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., & Wagenaar, J. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. Transportation Research Part B: Methodological, 63, 15-37.

Caprara, A., Fischetti, M., & Toth, P. (1999). A heuristic method for the set covering problem. Operations Research, 47(5), 730-743.

Caprara, A., Fischetti, M., Guida, P. L., Toth, P., & Vigo, D. (1999). Solution of large-scale railway crew planning problems: The Italian experience. In N. Wilson (Ed.), Computer-Aided Transit Scheduling. 471, pp. 1-18. Berlin: Springer.

Caprara, A., Fischetti, M., Toth, P., Vigo, D., & Guida, P. L. (1997). Algorithms for railway crew management. Mathematical Programming, 79, 125-141.

Chiang, T., Hau, H., Ming Chiang, H., Yun Kob, S., & Ho Hsieh, C. (1998). Knowledge-based system for railway scheduling. Data & Knowledge Engineering, 27, 289-312.

Christian, L., & Hanno, S. (2019). A collection of aspects why optimization projects for railway companies could risk not to succeed – A multi-perspective approach. Journal of Rail Transport Planning & Management, 11, 100149.

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

Dantzig, G. B., & Thapa., M. N. (2003). Linear programming: Theory and extensions (Vol. 2). New York: Springer.

D'Ariano, A. (2008). Improving real-time train dispatching: models, algorithms and applications. PhD thesis, Roma Tre University, Italy.

Department for Transport. (2021). Great British Railways: Williams-Shapps plan for rail. Retrieved 9 20, 2022, from https://www.gov.uk/government/publications/great-british-railways-williams-shapps-plan-for-rail

Derigs, U., Malcherek, D., & Schäfer, S. (2010). Supporting strategic crew management at passenger railways—model, method and system. Public Transport, 2, 307-334.

Desrochers, M. (1986). La Fabrication d'horaires de travail pour les conducteurs d'autobus par une methode de generation de colonnes. PhD thesis, Universite de Montreal, Canada.

Ehrgott, M. (2005). Multicriteria Optimization. Berlin: Springer.

Fang, S.-C., & Puthenpura., S. (1993). Linear optimization and extensions: theory and algorithms. Prentice-Hall, Inc.

Ford Jr, L. R., & Fulkerson, D. R. (1958). A suggested computation for maximal multi-commodity network flows. Management Science, 5(1), 97-101.

Geoffrion, A. M. (1974). Lagrangean relaxation for integer programming. In M. Balinski (Ed.), Approaches to integer programming (Vol. 2, pp. 82-114). Berlin, Heidelberg: Springer.

Grossman, T., & Wool, A. (1997). Computational experience with approximation algorithms for the set covering problem. European Journal of Operational Research, 101(1), 81-92.

Heil, J., Hoffmann, K., & Buschera, U. (2020). Railway crew scheduling: Models, methods and applications. European Journal of Operational Research, 283(2), 405-425.

Huisman, D. (2007). A column generation approach for the rail crew re-scheduling problem. European Journal of Operational Research, 180(1), 163-173.

Huisman, D., Kroon, L. G., Lentink, R. M., & Vromans, M. J. (2005). Operations Research in passenger railway transportation. Statistica Neerlandica, 59(4), 467-497.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In J. D. Guy Desaulniers (Ed.), Column generation (pp. 33-65). Boston, MA: Springer.

Jacobs, J. (2004). Reducing delays by means of computer-aided 'on-the-spot' rescheduling. Computers in Railways IX, 10.

Kantorovich, L. (1939). The Mathematical Method of Production Planning and Organization. Management Science, 6(4), 363-422.

Kroon, L., & Fischetti, M. (2000). Scheduling train drivers and guards: the Dutch "Noord-Oost" Case. Proceedings of the 33rd Hawaii International Conference on System Sciences.

Kroon, L., & Fischetti, M. (2001). Crew scheduling for Netherlands Railways "Destination: Customer". In S. Voss, & J. R. Daduna (Eds.), Computer-Aided Scheduling of Public Transport (Vol. volume 505 of Lecture Notes in Economic and Mathematical Systems, pp. 181-201). Berlin: Springer-Verlag.

Kroon, L., Maróti, G., & Nielsen, L. (2014). Rescheduling of Railway Rolling Stock with Dynamic Passenger Flows. Transportation Science, 49(2), 165-184.

Kwan, A. S., Kwan, R. S., Parker, M. E., & Wren, A. (1999). Producing train driver schedules under different operating strategies. In N. H. Wilson (Ed.), Computer-Aided Transit Scheduling (pp. 129-154). volume 471 of Lecture Notes in Economic and Mathematical Systems: Springer-Verlag.

Kwan, A., Kwan, R., Parker, M., & Wren, A. (1996). Producing train driver shifts by computer. In S. K. Kwa, R. S. Kwan, M. E. Parker, A. Wren, J. Allan, C. Brebbia, R. Hill, G. Sciutto, & S. Sone (Eds.), Computers in Railways V- Vol.1 Railway Systems and Management (pp. 421-435). Computational Mechanics Publications.

Louwerse, I., & Huisman, D. (2014). Adjusting a railway timetable in case of partial or complete blockades. European Journal of Operational Research, 235(3), 583-593.

Mann, D., & Panter, G. (2013). Contingency Planning for Train Service Recovery -Servic e Recovery. Retrieved 2 1, 2022, from https://www.raildeliverygroup.com/abou t-us/publications/acop/208-acop-sr2013-contingencyplanningfortrainservicere covery-issue1/file.html

Narayanaswami, S., & Rangaraj, N. (2012). Scheduling and Rescheduling of Railway Operations: A Review and Expository Analysis. Society of Operations Management, 2, 102-122.

NetworkRail. (2022). Retrieved 5 21, 2022, from https://www.networkrail.co.uk/wp-content/uploads/2017/11/How-rail-timetabling-works-factsheet.pdf

Nielsen, L. K. (2011). Rolling Stock Rescheduling in Passenger Railways. PhD thesis, Erasmus University Rotterdam, the Netherlands.

Nielsen, L. K., Kroon, L., & Maróti, G. (2012). A rolling horizon approach for disruption management of railway rolling stock. European Journal of Operational Research, 220(2), 496-509.

Oetting, A., Rittner, M., & Fey, S. (2013). Synchronal algorithms for determination and evaluation of conflict resolution scenarios for real-time rescheduling. Proceedings of RailCopenhagen 2013, 5th International Conference on Railway Operations Modelling and Analysis (ICROMA), May 13th – 15th, 2013. Copenhagen, Denmark.

Office of Rail and Road. (2013). Managing rail staff fatigue. Retrieved 10 5, 2022, from https://www.orr.gov.uk/media/10934

Office of Rail and Road. (2015). Understanding the rolling stock costs of TOCs in the UK. Retrieved 9 28, 2022, from https://www.orr.gov.uk/sites/default/files/om/und erstanding-the-rolling-stock-costs-of-uk-tocs.pdf

Potthoff, D. (2010). Railway Crew Rescheduling: Novel Approaches and Extensions. PhD thesis, Erasmus University Rotterdam, the Netherlands.

Rail Safety and Standard Board. (2012). Fatigue Management - A Good Practice Guide. Retrieved 8 1, 2021, from https://www.rssb.co.uk/standards-catalogue/CatalogueItem/RS504-Iss-1

Rezanova, N. J. (2009). The Train Driver Recovery Problem - Solution Method and Decision Support System. PhD thesis, Technical University of Denmark, Denmark.

Righini, G., & Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optimization, 3(3), 255-273.

RSSB. (2020). PERFORM: Enabling better planning and resource management during di sruption (T1154). Retrieved 10 3, 2022, from https://www.rssb.co.uk/research-catalogue/CatalogueItem/T1154

Ryan, D. M., & Foster, B. A. (1981). An integer programming approach to scheduling. In A.Wren (Ed.), Computer Scheduling of Public Transport (pp. 269-280). North-Holland Publishing Company.

Sahin, G., & Yuceoglu, B. (2011). Tactical crew planning in railways. Transportation Research Part E: Logistics and Transportation Review, 47(6), 1221-1243.

Sama, M., Meloni, C., D'Ariano, A., & Corman, F. (2015). A multi-criteria decision support methodology for real-time train scheduling. Journal of Rail Transport Planning & Management, 5(3), 146-162.

Sato, K., & Fukumura, N. (2010). An Algorithm for Freight Train Driver Rescheduling in Disruption Situations. Quarterly Report of RTRI, 51(2), 72-76.

Schiewe, P. (2020). Integrated Optimization in Public Transport Planning. Springer.

Serafini, P., & Ukovich, W. (1989). A Mathematical Model for Periodic Scheduling Problems. SIAM Journal of Discrete Mathematics, 2(4), 550-581.

Shor, N. Z. (1985). Minimization Methods for Non-differentiable Functions. Springer-Verlag.

Sodhi, M. S., & Norris, S. (2004). A flexible, fast, and optimal modeling approach applied to crew rostering at London Underground. Annals of Operations Research, 127, 259-281.

Stelzer, A. (2016). Automatisierte Konfliktbewertung und -losung fur die Anschlussdisposition im Schienen-Personenverkehr. PhD thesis, Technische Universitat Darmstadt, Germany.

Stephen, B., Lin, X., & Almir, M. (2004). Subgradient methods. Lecture notes of EE392o, Stanford University, Autumn Quarter, 2003.

Stoilova, S. (2020). An Integrated Multi-Criteria and Multi-Objective Optimisation Approach for Establishing the Transport Plan of Intercity Trains. Sustainability, 12(2), 687.

Suyabatmaz, A. Ç., & Şahin, G. (2015). Railway crew capacity planning problem with connectivity of schedules. Transportation Research Part E: Logistics and Transportation Review, 84, 88-100.

T'kindt, V., & Billaut, J.-C. (2006). Multicriteria Scheduling: Theory, Models and Algorithms. Berlin: Springer.

TransportFocus. (2020). Main Report Spring 2020. Retrieved 9 28, 2022, from National Rail Passenger Survey: https://d3cez36w5wymxj.cloudfront.net/wp-content/uploads/2020/07/16180916/Main-Report-Spring-2020.pdf

Vaidyanathan, B., Jha, K. C., & Ahuja, R. K. (2007). Multicommodity network flow approach to the railroad crew-scheduling problem. IBM Journal of Research and Development, 51, 325-344.

Vanderbeck, F. (1994). Decomposition and column generation for integer programs. PhD thesis, UCL-Université Catholique de Louvain, Belgium.

Veelenturf, L. P., Kidd, M. P., Cacchiani, V., Kroon, L. G., & Toth, P. (2016). A railway timetable rescheduling approach for handling large-scale disruptions. Transportation Science, 50(3), 841-862.

Veelenturf, L. P., Potthoff, D., Huisman, D., & Kroon, L. G. (2012). Railway crew rescheduling with retiming. Transportation Research Part C: Emerging Technologies, 20(1), 95-110.

Verhaegh, T., Huisman, D., Fioole, P.-J., & Vera, J. C. (2017). A heuristic for real-time crew rescheduling during small disruption. Public Transport, 9, 325-342.

Walker, C. G., Snowdon, J. N., & Ryan, D. M. (2005). Simultaneous disruption recovery of a train timetable and crew rost in real time. Computer & Operations Reseach, 32(8), 2077-2094.

Wegele, S., & Schnieder, E. (2004). Automated dispatching of train operations using genetic algorithms. Computers in Railways IX, 74, 10.

Williams Rail Review. (2019). Current Railway Models: Great Britain and Overseas. Retrieved 3 3, 2022, from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/786969/current-railway-models-gb-and-overseas-evidence-paper.pdf

Wren, A., & Kwan, R. S. (1999). Installing an urban transport scheduling system. Journal of Scheduling, 2(1), 3-17.

Wren, A., Fores, S., Kwan, A., Kwan, R., Parker, M., & Proll, L. (2003). A flexible system for scheduling drivers. Journal of Scheduling, 6(5), 437-455.

Zeng, Z., Meng, L., & Hong, X. (2018). Integrated optimization of Rolling Stock and Crew Rescheduling for High Speed Railway. Proceedings of 2018 International Conference on Intelligent Rail Transportation (ICIRT), (pp. 1-5). Singapore.

Zhan, S., Wong, S. C., Shang, P., & Lo, S. M. (2021). Train rescheduling in a major disruption on a highspeed railway network with seat reservation. Transportmetrica A: Transport Science, 18(3), 532-567.

Zhu, Y., & Goverde, R. (2020). Integrated timetable rescheduling and passenger reassignment during railway disruptions. Transportation Research Part B, 140, 282-314.