SCHOOL OF ENGINEERING

THE UNIVERSITY
OF BIRMINGHAM

INTELLIGENT REAL-TIME TRAIN RESCHEDULING MANAGEMENT FOR RAILWAY SYSTEM

By

LINSHA DAI

A thesis submitted to the University of Birmingham

for the degree of

DOCTOR OF PHILOSOPHY

14th June 2016

ABSTRACT

Demand for railway transport has grown within the last few years and continuous growth is projected in future years. Consequently, rail networks are increasingly being operated at the limits of their capacity and stability. The issue of managing a large and complex railway system with continuous traffic flows and mixed train services in a safe and punctual manner is very important, especially after disruptive events. A large number of recovery algorithms have been developed in recent years to support dispatchers in making decisions during railway operation. However, these algorithms do not have the ability to provide solutions to all situations which arise as delays develop.

In the first part of this thesis an analysis method is introduced which allows the visualisation and measurement of the propagation of delays in the railway network. By categorising the resilience of a railway operation into three levels according to the system response and operational strategies required for absorbing delays, train re-ordering and re-timing strategies are applied at junctions to solve real-time train rescheduling problems after minor disruptions to maintain a robust system. A simple case study has been conducted using the HERMES railway simulator in which different junction control strategies are applied in response to a delay situation. The case study shows how the visualisation and categorisation methods may be used to compare the effectiveness of different strategies in reducing the influence of knock-on delays.

The BRaVE simulator and the University of Birmingham Single Train Simulator (STS) are also introduced and a train running estimation using STS is described. The rescheduling process of the simulation is illustrated. A practical single junction rescheduling problem is then defined and a number of representative rescheduling approaches are applied to solve the problem. The algorithms considered are: Timetable-Order-Enforce (TOE), First-Come-First-Served (FCFS), First-Leave-First-Served (FLFS), Brute Force (BF), Dynamic Programming (DP), Decision Tree Based Elimination (DTBE), Tabu Search (TS), Local Search (LS), Simulated Annealing (SA), Genetic Algorithms (GA) and Ant Colony Optimisation Algorithm (ACO). These approaches are investigated and tested on a series of delay scenarios in microscopic simulation, and rescheduling solutions are compared and analysed. A case study investigates how different levels of delays and numbers of constraints may affect the performance of algorithms for network-wide rescheduling in terms of quality of solution and computation time. A recommendation for using these approaches is given based on their performance on different delay scenarios, and this can be used as a reference for further local rescheduling in decision centres.

In order to deal with operational dynamics, a methodology using performance-based supervisory control is proposed to provide rescheduling decisions over a wider area through the application of different rescheduling strategies in appropriate sequences. A single junction case study is designed to demonstrate how this process is realised, and it shows the improvement in reducing the propagation of delays by applying alternating approaches.

Finally, an architecture for a real-time train rescheduling framework, based on the distributed artificial intelligence system, is designed in order to handle railway traffic in a large-scale network intelligently, with different decision centres (DCs) processing together. A case study based on part of the East Coast Main Line using real-world data is considered, which demonstrates the effectiveness of adopting supervisory control to provide the rescheduling options for individual DCs.

Keywords: Real-time rescheduling, delay propagation, algorithms, supervisory control, DAI system.

ACKNOWLEDGEMENTS

I am very grateful to my supervisors, Professor Clive Roberts and Professor Felix Schmid, for their patient, continuous and attentive support and guidance during my study in the UK. I have greatly benefited from their inspiration, knowledge and experience in work and life. Without their generous support and encouragement, my study in the UK would not have been possible.

I would also like to extend my gratitude to the following people at the Birmingham Centre of Railway Research and Education: Dr Dave Kirkwood for sharing his expertise on JAVA programming and train simulations, Dr Gemma Nicolson for her helpful suggestions and inspiration for this thesis and Miss Katherine Slater for her great help for proof reading on my academic writing.

Many thanks also to all the members of the Birmingham Centre for Railway Research and Education for their kind help and supporting.

As well, I would like to thank my husband, my father and my parents-in-law, for their great love and support, which is my most precious thing in life.

Finally, I would like to say thank you to my mother, who was a great and selfless woman, for always loving me, supporting me, encouraging me and inspiring me, not only in the past, but also in the future.

TABLE OF CONTENTS

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

Appendix A

Appendix B

Appendix C

Appendix D

TABLE OF FIGURES

TABLE OF TABLES

ABBREVIATIONS

ACO	Ant Colony Optimisation
AG	Alternative Graph
AMCC	Avoid Most Critical Completion
ANN	Artificial Neural Network
ATO	Automatic Train Operation
ATP	Automatic Train Protection
B&B	Branch-and-Bound
BF	Brute Force
BRaVE	Birmingham Railway Virtual Environment
CDR/ CDS	Conflict Detection and Resolution/ Solution
CDRFR	Conflict Detection and Resolution with Fixed Routes
CT	Computing Time
DAI	Distributed Artificial Intelligence
DC	Decision Centre
DP	Dynamic Programming

DTBE	Decision Tree Based Elimination
ECML	East Coast Main Line
FCFS	First-Come-First-Served
FLFS	First-Leave-First-Served
GA	Genetic Algorithms
G-DL	Global Decision Logic
G-PM	Global Performance Monitor
HERMES	Holistic Environment for Rail Modelling and Experimental Simulation
JCM	Junction Control Method
KPI	Key Performance Indicator
KS	Knowledge Sources
L-DL	Local Decision Logic
L-PM	Local Performance Monitor
LS	Local Search
MAS	Maximum Allowable Speed
MIP	Mixed Integer Programming

OR	Optimisation Rate
PDD	Primary Delay Detection
PPOD	Possession Plans On Demand
PPS	Problem Space Search
QoS	Quality of Service
RB	Rule Based
RT	Recovery Time
SA	Simulated Annealing
SST	Stage-to stage transform
STS	Single Train Simulator
TD	Total Delay
TDP	Total Delay Penalty
TEG	Time Event Graph
TM	Traffic Manager
TOC	Train Operating Company
TOE	Timetable Order Enforce

TS	Tabu Search
TTP	Train Timetabling Problem
WTA	Waiting Time Agreement

CHAPTER 1

INTRODCTION

1.1 Research background

Demand for railway transport has grown in recent years and continuous growth is projected in coming years. Transport in the Great Britain is facilitated by road, rail, air and water networks. According to the transport statistics report [1], in 2003 91% of the total distance travelled was by road, of which 85% was by car or taxi and 6% by bus. Rail transport accounted for 6% and all other modes accounted for roughly 3% in total. By 2013, the percentage of distance travelled by rail had increased to 9%, whereas the percentage by road had decreased to 88%.

The growing demand for railway transport has led to an increased requirement for higher quality railway services, including improved speed, capacity, punctuality and comfort. To fulfil this purpose, new lines were built and existing infrastructure was enhanced, with more services offered. In the UK, the national railway network of 10,072 route miles (16,116 km) in Great Britain and 189 route miles (303 route km) in Northern Ireland with 18,000 passenger trains and 1,000 freight trains operating daily. Urban rail networks exist in the main cities of the UK, such as London, Manchester and Birmingham [2]. As the construction and upgrade of infrastructure is expensive, time consuming and sensitive to environmental and social concerns, greater efficiency and higher capacity on the existing rail networks is required to optimise investment within the limits of the infrastructure [3].

As the number of trains in operation increases, it is also more difficult to cope with delays. Network Rail's average annual punctuality stood at 91.5% in January 2013 [4, 5], based on the moving annual average Public Performance Measure. This is defined as the percentage of planned trains that are not cancelled and do not arrive at their destination late. Late is defined to be 5 minutes late for London, the South East and other suburban train operations, and 10 minutes late for other areas. This statistic shows that around 1 in 10 trains do not arrive on time. Due to the limited resources available (tracks, platforms, etc.), a small problem can lead to a chain of knock-on delays that affect other services. Generally, with robust timetables, the knock-on delays can be absorbed gradually by the margins in the timetables. However, there still are some delays which are too big to be absorbed by the system automatically, and dispatchers must make decisions to help to reduce these delays.

Railway traffic control is a way of maintaining the quality of service with a high standard of punctuality while ensuring the safety of train operations. According to basic safety regulations, no more than one train is allowed in a block section at any time in the fixed blocking system: the signalling system with Automatic Train Protection (ATP) functions is usually used to ensure a safe headway between successive trains by restricting the availability of track segments. Nowadays, with the development of railway technology, train traffic control has become more advanced and more comprehensive. Because of the complexity of railway operations, railway traffic control approaches mainly refer to two aspects: off-line and on-line [6]. Off-line railway traffic control means that the schedule is built to abide by public constraints and satisfy necessary requirements. The approaches for off-line optimisation include timetabling, train routing and train platforming, etc. In contrast, in on-line railway traffic control, dispatchers make decisions to maintain the schedule when unpredictable incidents occur. Furthermore, solutions must be found quickly in a valid time limit, thus problems must be solved in real-time, or near real-time. Thus, on-line optimisation is also called real-time management, including real-time train rescheduling and real-time operational management.

Nevertheless, it is still a challenge to carry out a rescheduling plan in a large-scale railway network with uncertain delays and mixed traffic. In addition, the fact that different train operating companies (TOCs) and open access operators share the limited infrastructure has resulted in a complicated situation for railway traffic control. Various algorithms are used as the centralised methods for train rescheduling in a local area but they lack the ability to deal with the complicated traffic in a large-scale network. Local control is defined as junctions or stations controlled by individual signal boxes, which can contact each other. With the introduction of a control panel, a dispatcher can set routes for the trains through an interlocking area [7].

1.2 Motivation and objective

The issue of managing a large and complex railway system with continuous traffic flows and mixed train services in a safe and punctual manner is very important, especially after disruptive events. The purpose of real-time train rescheduling is to find an optimised schedule for the services in a given area and a given time horizon during operations, by providing a solution that is compatible with the actual traffic conditions and infrastructure constraints. The objectives for rescheduling are varied, for example, to reduce the sum of delays of all considered trains [8]; to minimise energy consumption [9]; to minimise the number of delayed passengers [10]; and to decrease the delay cost of the most delayed train [11]. After communicating with TOCs, the general goal of the rescheduling operation for them is to avoid secondary delays or at least to reduce the effect of delays.

Train rescheduling requires changes to the timetable, and if necessary, the rolling stock and crews as well. Currently, the most commonly used method is to send a dispatcher to make decisions manually, which are usually based on experience and knowledge, without any support from an intelligent decision tool [12]. Recently, a large number of existing recovery algorithms have been developed to support the dispatchers in making their decisions, including centralised approaches and distributed approaches. However, they lack the ability to provide answers to all questions about the development of delays.

Because it is necessary to reduce delays in railway operation, it is essential to study the propagation of delays. Railway operation simulation provides a way of analysing delay propagation in the whole railway network. Knowledge of delay propagation helps the construction of a better rescheduling plan for the railway network due to a deeper understanding of the interaction between trains in complicated situations [13].

The motivation of this research is to provide the dispatchers with better decision support by considering the delay propagation of the whole railway network with continuous traffic flows and mixed services after disruptive events.

Consequently, this thesis consists of the following research tasks:

1. The propagation of delays in the railway network is visualised and evaluated.
2. Train running estimation using the University of Birmingham Single Train Simulator (STS) and the microscopic simulator (BRaVE) is introduced to simulate train operations in the railway network.
3. A rescheduling process is specified based on a case study with different advanced rescheduling approaches applied.

4. The methodology of supervisory control for railway rescheduling is modelled and verified to manage the railway traffic dynamically.
5. An architecture for a real-time rescheduling framework based on a distributed artificial intelligence (DAI) system is designed in order to handle railway traffic in a large-scale network intelligently, with different decision centres (DCs) processing together.

The thesis will offer responses to the following hypothesis: A railway system with a high degree of resilience could absorb a certain level of train delays, and recover from perturbed railway traffic rapidly and stably by applying train rescheduling approaches at junctions.

The aim of this research is thus to define the concept of resilience of railway systems and to improve the degree of resilience by using advanced railway traffic recovery approaches at junctions in the event of disturbances. Various train rescheduling approaches will be investigated at one junction and a performance-based supervisory control approach will be developed to deal with train rescheduling and dynamic traffic flows in distributed areas. Train rescheduling approaches can be altered to respond to the real-time situation according to this supervisory control. The resilience of the railway network is expected to be enhanced with these approaches applied.

An architecture of train rescheduling based on the distributed artificial intelligence (DAI) system will be constructed to deal with train rescheduling problems in a large-scale network. The resilience of the railway network is expected to be enhanced by alternating the approaches applied at different junctions.

1.3 Outline of the thesis

The thesis is organised as follows:

- Chapter 1 introduces the background, research motivation, research objective and the structure of the thesis.
- Chapter 2 classifies and reviews related literature of real-time railway rescheduling and introduces the existing recovery models used in real-time train rescheduling.
- Chapter 3 presents a clear definition of system resilience by dividing it into three categories. A visualisation method is developed to evaluate the system resilience by monitoring the delay propagation in the railway network.
- Chapter 4 introduces the simulation platform used in this thesis. The train running estimation is described in detail and the rescheduling process of simulation is illustrated.
- Chapter 5 investigates various train rescheduling approaches at one junction area to provide a comparison of different algorithms in different delay scenarios.
- Chapter 6 develops the supervisory control to alter algorithms at the junction area in order to deal with dynamic traffic flows and respond to different delay situations. The mathematical formulation of this methodology is presented and a verification case study is provided.
- Chapter 7 introduces an architecture for intelligent real-time train rescheduling in a large-scale railway network and presents a description of all architecture modules. A case study on the East Coast Main Line (ECML) with mixed traffic and uncertain delays is described. With the application of the intelligent train rescheduling system developed in this thesis, the delay propagation indicates that the resilience of the system is enhanced.
- Chapter 8 presents conclusions and comments on further work.

CHAPTER 2

AN OVERVIEW OF RECOVERY MODELS FOR REAL-TIME TRAIN RESCHEDULING

2.1 Introduction of the train rescheduling problem

The occurrence of unexpected disruptions to railway operations, such as the unavailability of tracks, bad weather, rolling stock breakdown and suicide, usually leads to a chain of delays to passenger trains with a consequent decrease in the quality of service to passengers. Therefore, decisions are required in order to minimise the effect of such delays, both at the timetabling stage and at the rescheduling stage.

Railway operations are usually based on a timetable, which specifies the journeys that are carried out by the trains in the system, as well as the detailed routes and stations at which the trains should stop. At the timetabling stage, the objectives include minimising the journey time for passengers, maintaining the robustness of the timetable and maximising the capacity and service frequency [14, 15]. The robustness of the timetable, i.e. the ability to resist minor disturbance, is crucial to ensure the system can cope with delays. Running time supplements and buffer times are introduced in the timetabling process to enable trains to compensate delays without influencing sequential trains to some extent. However, running time supplements and buffer times are still limited by the capacity constraints set by TOCs and they fail to ensure recovery from major disruptions without making any dispatching decisions.

At the rescheduling stage, it is necessary to maintain passenger satisfaction and restore the service of the railway system by providing decisions to dispatchers in a disruption situation. Due to the complexity of the railway system, the delay recovery problem is usually divided into phases, which are: (i) timetable rescheduling, (ii) rolling stock rescheduling, and (iii) crew rescheduling. Timetable rescheduling calls for a new conflict free timetable by applying reordering, retiming, rerouting or even cancellation decisions, and rolling stock and crew rescheduling are demanded to operate the derived timetable if necessary, due to the changes made in the timetable rescheduling phase [16]. In this thesis, the rescheduling problem is restricted to the timetable rescheduling phase for disturbances. The classification of the train rescheduling problem and different types of delays are discussed in the next section.

2.1.1 Classification of train rescheduling problem

A definition of the train rescheduling problem is given by Pacciarelli [17]: Given static information (e.g. timetable, railway infrastructure, train configurations) and dynamic information (e.g. disruptions, train positions at time t), find a new conflict-free timetable for the time window $[\mathrm{t}+\mathrm{a}, \mathrm{t}+\mathrm{b}]$, such that some performance index is minimised. Different train rescheduling problems are categorised due to the range of a and b.

- When $\mathrm{a} \leq 2\ \mathrm{min}, \mathrm{b} \leq 45\ \mathrm{min}$,

  This train rescheduling problem is defined as real-time rescheduling or CDR/CDS (Conflict Detection and Resolution/Solution). Real-time rescheduling aims to provide a new conflict-free disposition schedule without rerouting, replatforming, speed optimisation, etc. [18, 19]. The objectives for the real-time rescheduling process are varied and difficult to quantify; the existing research objectives include train delay minimisation, capacity utilization, railway cost, etc. [20].

- When $2\sim3\text{ min} \leq \text{a} \leq 10\sim15\text{min}, \text{b} \leq 2\sim3\text{ hours}$,

  This train rescheduling problem is defined as delay management. Delay management focuses on minimising the inconvenience for passengers at stations and the main task is deciding which connections to 'drop' and which to maintain. The current practise in delay management is Waiting Time Agreement (WTA)-for each connection, whereby a maximal waiting time for the connecting train is determined [21]. Classic delay management models the delay of passengers who miss a connection as exactly one cycle time [22]. Recent research on delay management adds more flexibility, such as rerouting the passengers, replatforming the trains, etc. [23].

- When $\text{b} \geq 2\sim3\text{ hours}$,

  This train rescheduling problem is defined as disruption management. Disruption management is focused on designing a new timetable with the available resources, including rolling stock rescheduling and crew rescheduling [24, 25]. Emergency timetables are usually defined in advance and adapted in real-time according to the delay situation [26].

2.1.2 Different types of train delays

During daily rail operations, it is inevitable for delays to occur due to reasons such as bad weather, train connections, technical problems, blocked doors, etc. According to the location and source of the generation of these delays, they are generally classified as initial, original or knock-on delays [27, 28]. Initial delays are those recorded at the boundary when a train is entering a network, or at the beginning of a journey. Original delays, usually known as primary delays, are those which occur due to external reasons, like engine failure, reduction in speed, boarding time of passengers, weather conditions and accidents. These delays are hard to predict and reduce. Knock-on delays are the consequences of other delays. Some knock-on delays happen because a delayed train is occupying the route and preventing other trains from passing (e.g. at a crossing). Some happen as trains that wait for connections from other delayed trains. The number and severity of knock-on delays can reflect on the resilience of the timetable and the reliability of train operations.

According to the duration of time, delays can be categorized into disturbances and disruptions [12]. A disturbance is “relatively small” and it can be managed by timetable rescheduling only, without any rolling stock and crew rescheduling. A disruption is “relatively large” and strongly influences the timetable. Rolling stock and crew rescheduling are required as well as timetable rescheduling. A disruption may result in large delays or the cancellation of trains. Usually, the duration of a disruption is unknown when it starts, and rescheduling management has to be applied several times, until the normal timetable is restored.

2.2 Problem description

Generally, the railway network consists of tracks, signals and stations. Signals are located along the tracks and inside the station area. A track segment between two main signals is called a block section (in the fixed block signalling system) that may host only one train at a time, which means that a train is only allowed to enter a block section after the preceding train has left this section completely.

For safety reasons, signals are used to control the movements of trains by imposing the minimum distance headway between consecutive trains. An interlocking system can control the train traffic by setting up conflict-free routes for the trains. The Automatic Train Protection (ATP) system is a control system used by the operators to help avoid collisions by automatically restricting the Maximum Allowable Speed (MAS) at which a train can travel at any given time according to its current status. ATP can operate automatic braking when a train ignores a valid speed restriction.

Timetables are designed to satisfy all traffic requirements. However, during practical operations, unexpected events occur that can make the original timetable infeasible. Unscheduled braking and stopping of trains increases running times and causes knock-on delays. Real-time train dispatchers must adjust the timetable of each train, in terms of retiming, reordering and routing the trains at the entrance of each crossing point. In order to choose the most effective timetable modifications, the goals of rescheduling should be defined while satisfying traffic regulation constraints. For example: to minimise train delay (sum, max, average, etc.) [8, 11, 29], to minimise passenger delay [10], to minimise energy consumption [9], or a combination of these goals.

In summary, the real-time rescheduling problem can be defined as follows: given a railway network, the position and speed of each train is known by the dispatchers, and decisions covering a set of train routes and passing times at each relevant crossing point in the network must be made when a delay is detected that may result in a conflict with the optimisation objectives. The classic rescheduling process can be described in the following steps [30]:

- Begin with a nominal timetable
- Transmission of data to the operation
- Detection of conflicts
- Resolution of conflicts
- Generation of new schedule plan
- Application of traffic control management

2.3 Recovery models for real-time train rescheduling

Different approaches have been developed to solve the real-time rescheduling problem. In this section, several recovery models are described at microscopic level and macroscopic level. The distinction between them concerns the level of detail of the infrastructure in the railway system. In former one, the infrastructure is considered in details. However, the latter one consider the railway network in a higher level, in which stations and tracks are represented by nodes and arcs respectively, the details of block sections and signalling are not considered. A classification of recent literature on different recovery models is presented in Table 2-1 and a detailed introduction to each model and the approaches to rescheduling are given in the following section.

	Microscopic models	**Macroscopic models**
Real-time train rescheduling	• Alternative Graph (AG) model [31], [32, 33], [34], [35], [36], [37]. • Stage-to stage transform (SST) model [38], [39], [40], [41]. • Decision tree model [42], [43], [44]. • Mixed Integer Programming (MIP) model [45], [46], [47], [48, 49], [19]. • Simulation model [50], [51], [52], [53], [54], [7].	• Discrete event model [55], [56], [57], [8], [58]. • Time event graph (TEG) model [59, 60], [61, 62]. • MIP model [63], [64], [65], [66]. • Other models[67], [68], [69], [70], [71].

Table 2-1: Classification of recent literatures

2.3.1 Microscopic models

Alternative Graph (AG) model

Mascis and Pacciarelli introduce an Alternative Graph (AG) model for no-store job shop scheduling, which is similar to the train rescheduling problem [31]. AG is based on the track layout of the railway network; the train routes can be divided into many block sections and the running times to pass different sections are fixed. The passing of a train through a particular block section is defined as a node and the processing time is defined as an arc. Since a block section cannot hold more than one train at the same time, to avoid conflict, a processing order must be defined and the order of two operations can be modelled as a pair of alternative arcs. Let V be the set of all the operations (nodes), F be the set of all the fixed arcs, and A be the set of all the alternative arcs in an instance. The alternative graph $G = (V, F \cup A)$ is defined for the train rescheduling problem. Figure 2-1 illustrates the AG for a simple train rescheduling problem. On the left side, the figure shows a small railway network with 5 block sections (denoted as 1, 2, 3, 4 and 6) and one junction section (denoted as 5). Assume at the time t, there are two trains (denoted as A and B) in the network and they are approaching the same junction. The nodes on the right side indicate a passing of a particular train through a block, here, $t(ij)$ indicates the cover time for train i in the section j. According to the block constraints in the railway network, only one train is allowed to run in a block section at any time. Collisions may happen if two or more trains approach to the same block section at the same time. In particular, trains A and B share resource section 5 and 6, and therefore the dispatchers have selected alternative arcs (A6, B5) and (B6, A5). If train A precedes B to pass the junction area, arc (A6, B5) is selected; on the contrary, if train B enters junction block before A, arc (B6, A5) is selected.

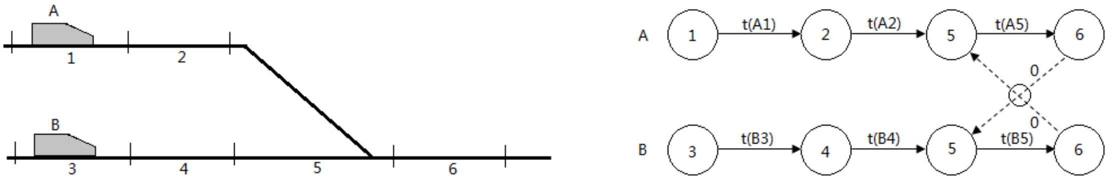


Figure 2-1: Example of an AG model

Mazzarello and Ottaviani proposed a heuristic method (Avoid Most Critical Completion time (AMCC)) for reordering, rerouting and speed adjustment to minimise the exit delays based on an AG model. The cases include 27 trains at the Schiphol Station bottleneck area in the Netherlands and 4 trains at the junction at Lage Zwaluwe, also in the Netherlands [32].

D'Ariano proposed a Branch-and-Bound (B&B) algorithm to reschedule a bottleneck area of the Dutch rail network based on an AG formulation [33]. The results present optimal or near optimal solutions which are calculated within a short time. Later, in 2008, D'Ariano described the implementation of the AG model in the ROMA (Railway Traffic Optimisation by Means of Alternative graphs) [34]. In 2009, D'Ariano developed a Tabu search for rerouting and speed adjustment to minimise the total delay, including the primary delay and maximum consecutive delay [35].

Corman extended the research in 2012 with the objective of minimising delays and missed connections by adopting two heuristic methods for selecting connections and a B&B method for reordering based on an AG model [36]. In the following year, Dollevoet and Corman proposed an iterative approach incorporating the computed wait-depart decisions in Utrecht Central Station with the objective of minimising total passenger delay [37].

Stage-to stage transform (SST) model

The railway rescheduling problem at junctions can be structured as a series of ‘stages’. As shown in Figure 2-2, four trains are approaching a junction, starting at Stage 0 which indicates the original situation (no train has passed), each sub-stage shows all possible next states (where one state indicates one possible sequence of trains) and links between each state give the cost of moving (for instance, time consumed passing the next sequential train). The optimal solution can be found by seeking the shortest path moving from the first stage to the last stage [38]. This kind of model is usually known as a stage-to stage transform (SST) model.

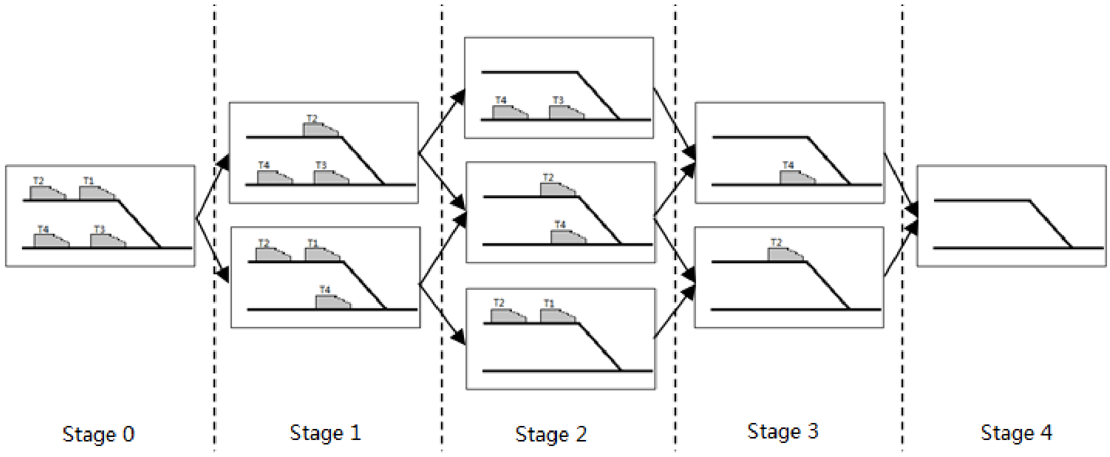


Figure 2-2: Example of a SST model [72]

Some literature on train rescheduling is based on a stage-to-stage model, for example, dynamic programming (DP) is one of the most frequently used algorithms based on this model [39]. Ho and Haugland introduce a control strategy based on DP by adopting a cost function based on total weighted train delay to determine an optimal train sequence [40]. Albrecht and Oettich describe the dynamic schedule synchronisation for service connection by using DP. The cost function is deemed as the passenger waiting time between services and the stations are structured as a multi-stage process [41].

Decision tree model

The decision tree model is another frequently used model to identify feasible solutions for a train rescheduling problem. For example, in Figure 2-3, on the left side, there are four trains which are approaching a junction. On the right side, the root of the tree indicates the initial state where no trains have passed the junction area. The first level of branches represents all possible first trains through the junction and the subsequent level of branches represents possible successive trains. The leaves of the tree show all valid sequences for trains passing the junction.

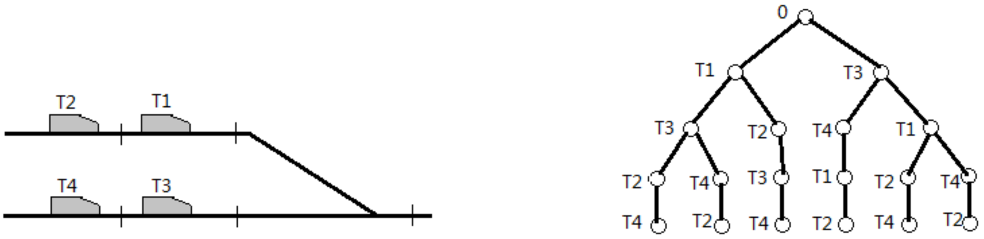


Figure 2-3: Example of a decision tree

Depending on the optimisation algorithms, a decision tree can be used to find the optimal solution for train reordering at the junction. Shah and Sastry proposed a pruning technique to evaluate the nodes by an objective function, such that if it does not meet a predefined threshold, the branch is cut [42]. This approach increases the speed of finding a solution. However, it cannot guarantee that the optimal solution will be found every time. In the train rescheduling study, a decision tree model is used to minimise the train delays on a mixed traffic network by Weston [43]. In this study, the objective function is the measure of passenger delay minutes. A decision tree can also be used to solve the train rerouting problem; Ismail models the destination of routing queries as a decision tree model and the solution for the train rerouting problem can be found within a practical response time [44].

Mixed Integer Programming (MIP) model

A Mixed Integer Programming (MIP) model is developed for solving the train rescheduling problem on a single line railway. The rescheduling process can be represented by binary decision variables, such as maintenance connections, the sequence of trains, the assignment of resources, etc. The continuous variables are defined to represent the arrival time and departure time of each train at the station. Delays can also be defined as integer variables. The constraints in the railway network are generally expressed as the constraint equations in a MIP model. The decision variables and sets of constraints are generally different from one researcher's work to the next, which therefore results in different levels of detail in the MIP models.

A mathematical model to describe the timetable rescheduling problem in case of delays based on the MIP model was presented by Gély (2006). This model presents binary variables indicating re-ordering choices and continuous variables representing the arrival or departure time of trains from each visited node in the network [45].

Van de Boom et al. described the train reordering problem using a MIP model by determining the order of trains as the binary variables and the time instants at which a train departs from the station as the continuous variables. All constraints are presented by a matrix. Genetic algorithms are used to minimise the sum of all predicted delays and the penalty for all broken connections and switched train orders is based on the model of the Dutch railway system [46].

Chen (2012) formulated the junction rescheduling problem with a MIP model, with the objective function defined as the Weighted Average Delay of all passing trains and the binary decision variables are defined for the route setting (whether or not the route is set for the train to pass) and headway maintenance (whether or not the headway in consecutive events must be kept). The algorithm DE_JRM (Differential Evolution for Junction Rescheduling Model) is derived and improved from a general Differential Evolution (DE) algorithm for solving proposed MIP problems [47].

From 2012 to 2013, Boccia et al. developed a new MIP model using binary variables to indicate whether a route is occupied by a train or not, and continuous variables to represent the time at which a train reaches a block section and the delay of a train on a route. Heuristic approaches are proposed in a simplified network with the idea of pre-selecting a “most promising route” for each train [48, 49].

To minimise the total costs incurred by the train delay, in 2012 Yan and Yang formulated the Movement Planner Problem as a MIP model and developed several heuristic rules for variable fixing. In addition, they proposed a rolling-horizon-based decomposition algorithm to “divide and conquer” the problem. The experiments on the three data sets show that the solution approaches perform better than the existing pure branch-and-cut algorithm in providing high quality solutions [19].

Simulation model

Simulation is the development of models to analyse and study the dynamic behaviour of actual or hypothesized real-life systems. The simulation approach can be divided into two types: discrete or continuous. In discrete simulation, the data are only collected when there are certain changes to the discrete event, whereas continuous simulation collects data continuously [50]. An event-driven based approach and network-based approach, as two widely used discrete models, are used for railway rescheduling by Cheng (1996), with different algorithms applied [50].

In the railway system, computer-based continuous simulation models (e.g. HERMES and BRaVE) are usually used to model the train operation at a microscopic level based on the detailed information of the infrastructure (e.g. track layout, signalling, station platform allocation, etc.) and train configuration (e.g. maximum speed, acceleration, braking, etc.). The simulation model can not only present the real time status, but can also forecast the future status. Therefore, the simulation model can be used to support real-time rescheduling. Simulation approaches have the capability of showing the performance of train rescheduling strategies. Based on a well-defined simulation model, the rescheduling strategies can be validated and evaluated.

The components of a railway network from Downtown Los Angeles to the Eastern Inland Empire area and movements of trains and passengers are modelled by Lu et al. A Velocity-Augmenting algorithm and deadlock-free algorithm are proposed and validated in minimising the travel time of trains based on this model [51]. Jacobs (2004) presented a computer procedure to generate conflict-free rescheduled plans for disturbed services with an asynchronous simulation approach based on blocking times [52].

An investigation of microscopic simulation models for rescheduling in a large-scale network can be found in a paper written by Jacobs [53]. Asynchronous simulation is coupled with heuristic approaches for conflicts by ranking trains into categories and always giving priority to trains in a higher category. It is a feasible solution for train rescheduling in a large-scale network but the resulting schedule can be far from the global optimum.

Recent research based on microscopic models decompose a large-scale rescheduling problem into several sub-problems with computable scale and these sub-problems are solved locally with collaborative activities to achieve a global optimum [54]. Chou introduced a collaborative method to optimise the train passing sequences in junctions of a railway network and validated this method through simulation. He decomposed the entire railway network into individual junction areas and used a greedy local search to find the optimal solutions for each junction until none of these solutions conflicted with each other [7].

2.3.2 Macroscopic models

Discrete event model

Continuous simulation models require the information of the position and speed of trains at every time increment and the large amount of computation prevents them from being practical. In 1991, Van Breusegem et al. presented an event-based traffic model for metro lines. This model accounts only for discrete events occurring on the line (e.g. arrival or departure of trains at platforms). The corresponding variables are related to both trains and platforms [55].

In the studies of discrete models, the railway system is described by discrete events and the state of each train at the time, *t*. A discrete-event dynamic system consists of nodes and arcs. Nodes indicate the events, and arcs refer to the processes. For a railway network, the individual train operation can be regarded as a series of events and processes to connect them. The node can be defined by the train identity, location, type (departure, arrival or passing) and scheduled event time. The arcs are defined by the train identity, type (run or stop), linked two adjacent events and the minimum processing time. The interactions between trains are modelled with headway and connection processes. Headway processes separate the passing events of different trains which share the same block section and connection processes separate the departure event of a connecting train and the arrival event of a feeder train [56]. An event can only occur after all processes represented by incoming arcs have been completed and it follows a fixed order determined at the beginning.

As a simple example, consider the railway transportation between three consecutive junctions based on the discrete-event model in Figure 2-4. This network consists of three stations (Station A, Station B and Station C) and two lines (Line 1 between station A and B, and Line 2 between station B and C). Nodes 1-4 indicate all departure and arrival events for train 1 and nodes 5-8 indicate all departure and arrival events for train 2. The process times are weighted in minutes by different arcs: the running times and dwell times are weighted by black arcs, the connection times and the departure headway times are weighted by light grey arcs.

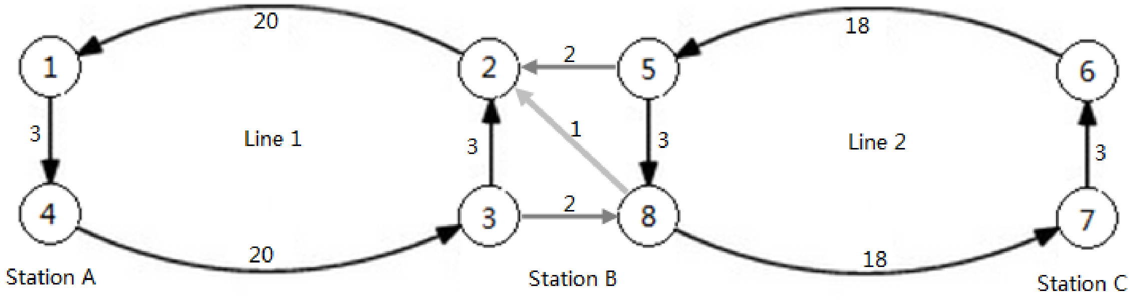


Figure 2-4: Example of discrete-event railway system

Compared with the discrete time based simulation models, the discrete event based model is more efficient [57]. Ho et al. (2001) used three heuristic algorithms (GA, SA and TS) in a discrete event model to solve the conflicts in a reasonable computation time with the objective of minimising total weighted delay in [8]. Dorfman (2004) developed a Greedy Travel Advance Strategy (TAS) to schedule trains on a railway network using a discrete event model to minimise the energy cost [58].

Time event graph (TEG) model

Timed Event Graphs (TEG), also known as timed Petri nets, are used to model the railway operation at a macroscopic level by Goverde [59, 60]. A TEG is a representation of a discrete-event dynamic system. The nodes can be portioned into two disjoint subsets: places (usually depicted as circles) and transitions (usually depicted as bars). Arcs from places to transitions do exist, as well as from transitions to places. However, arcs from transitions to transitions or from places to places do not exist. Moreover, the marking and holding times are assigned to the places. The marking (usually depicted as tokens in the places) can be interpreted as one or more active processes at the beginning of each observed period, while the holding time of a place is the minimum process time that a token is occupies the place. The dynamic processes in a discrete-event system are defined by the movements of tokens from place to place in the TEG model.

Figure 2-5 shows a TEG of the example network in Figure 2-4. The location of tokens indicates that train 1 is dwelling at Station A and train 2 is running on Line 2 towards Station B at the beginning of this observation period. When event 4 (departure event of train 1 at Station A) has occurred (fired), Figure 2-6 shows that the token of the input place of event 4 has moved to the output place of event 4.

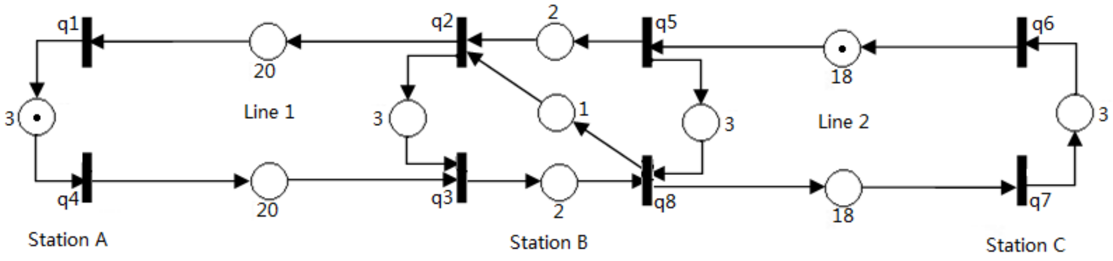


Figure 2-5: TEG model with initial marking

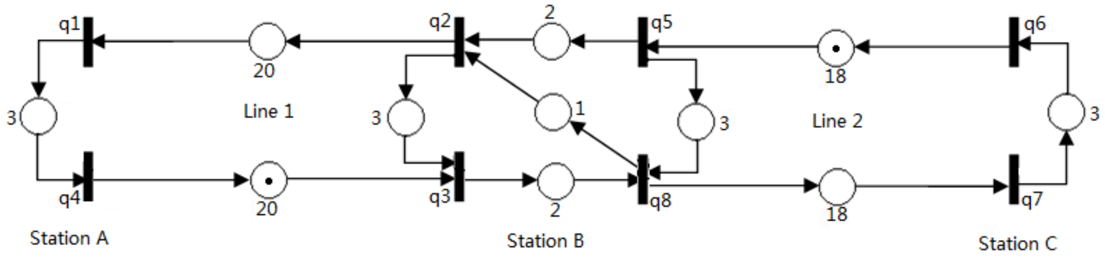


Figure 2-6: TEG model after event 4 fired

A time event graph can be represented by a max-plus linear system whereby an event occurs after the process time of the last predecessor events have finished. Computational algorithms exist for optimising the performance indicators of a max-plus linear system.

Goverde (2010) developed a bucked-based delay propagation algorithm based on the max-plus delay propagation model for minimising the propagation of train delays [59]. Schutter and van den Boom extended the model predictive control (MPC) to the class of max-plus linear system with soft and hard synchronization constraints for a railway network and this method is used to minimise the sum of deviations from the original schedule and number of broken connections [61, 62].

MIP model

There are also a number of researchers who solve the train rescheduling problem based on a MIP model in a macroscopic level. For example, an MIP model which considers the reordering and rerouting of trains is presented by Törnquist and Persson (2007) [63]. This model contains binary variables to express whether or not an event occupied a track, and continuous variables to indicate the start times and end times of an event. The model assumes that the headway times between the trains and running times along the segments between stations are fixed. Since the sequence of trains on tracks is predefined by the original timetable and remains the same during the operations, there are only a few modifications which can be made to minimise the train delays. Four different strategies are investigated based on the MIP model of part of the Swedish railway network. In 2012, Törnquist developed a heuristic greedy algorithm for the same problem and this algorithm can quickly find a good solution according to a set of criteria [64].

Acuna-Agost et al. extended the model presented by Törnquist and Persson in 2007 and used the same idea in 2010. There are two main changes in the latter model: one is that unplanned stops will change the minimal travel time when considering acceleration and braking behaviour; and the other is some modifications on constraints to admit more than one train in the same block section running in the same direction [65]. Right-shift Rescheduling, the MIP-based Local Search Method and the Iterative MIP-based Local Search Algorithm are considered in this paper to solve the train rescheduling problem. The experimental results show that the best compromise is obtained with the Iterative MIP-based local search procedure.

Min et al. formulated the train-conflict resolution problem as an MIP model in 2011. They argue that the train-conflict resolution problem is NP-hard and proposed a column-generation-based algorithm that exploits the 'separability' of the problem. During tests of this approach on real data from the Seoul Metropolitan Railway network, the proposed algorithm managed to provide near-optimal conflict-free timetables in a few seconds for most cases [66].

Other models

Other macroscopic models include an integrated framework based on a pattern description language for automatic train rescheduling proposed by Hirai [67]. Here, a large number of rescheduling plans are prepared to help manage major disruptions caused by an incident which is likely to require more than an hour of suspended train operations.

Missikoff described a fast prototype of MINT (Manager on Integrated Network of Train traffic), which combines information management and decision-support functions in a single integrated system for railway traffic control. This integration has been realised by means of an object-oriented approach and all the developments have been supported by Mosaico, which is a software environment for analysing, designing and rapid prototyping of most object-oriented applications [68].

Barta et al developed a Markov-chain based model for delay propagation in the railway network to evaluate the evolution of train delays as a train visits successive terminals. The model is based on the examination of a large set of historical data and shows how to classify different terminals according to their ability to absorb or amplify delays [69].

Albrecht et al described the railway network with a Possession Plans On Demand (PPOD) system, which can simultaneously consider track maintenance and trains in order to produce good integrated timetables. The dispatch can be customized to model different operating procedures by altering a combination of the dispatch decision time, possessor selection rule and action selection rule. Problem Space Search (PPS) is used to quickly generate a range of feasible timetables and a user-defined objective may be used to rank these timetables [70].

Dündar and Şahin studied the train rescheduling problem for benchmarking purposes, they developed an artificial neural network (ANN) model to mimic the decision behaviour of train dispatchers when processing conflict resolutions. ANN is an artificial intelligence method, which can learn, remember and memorise, which is developed based on the human brain's basic operation principals [71].

2.4 Conclusion

Due to the complexity of railway operations, the growth of traffic demands and limited resources, the occurrence of disruptions may lead to a chain of delays. Timetabling and rescheduling approaches are demanded to minimise the effect of the delays. Although a good timetable has the ability to recover from minor disturbances by implementing the running time supplements and buffer times, major disruptions are hardly recovered without dispatching decisions.

In this chapter, the rescheduling problem in a railway network is detailed, defined and classified into three categories: real-time rescheduling, delay management and disruption management according to the response time to the initial or primary delay. Delays can be classified as initial, original and knock-on delays according to the source of generation, or disturbances and disruptions according to the duration of time.

A detailed description of the real-time train rescheduling problem is presented and some basic terms for modelling a train rescheduling problem are introduced. With the increasing attention which is currently being given to the rescheduling problem in the railway network, a lot of researches on recovery models and solution approaches have been done in recent years. Several recovery models for real-time train rescheduling are reviewed at a microscopic and macroscopic level. The microscopic models include the: AG model, SST model, decision tree model, MIP-based models and simulation models, and the macroscopic models including the discrete-event model, TEG model, MIP-based models and other undefined models. These recovery models offer a different understanding of the train rescheduling problem and some algorithms based on the models described are developed to carry out rescheduling solutions.

The development of computer technology makes the continuous simulation of the rail network available for practical real-time rescheduling. A good simulation can present the real-world operation very precisely and provides useful information for train dispatchers. In this research, a microscopic simulation model of railway network is used for the real-time rescheduling management, and recovery approaches are implemented to that model. With an accurate rail simulator, different rescheduling approaches can be verified and investigated by evaluating their performance in different delayed scenarios.

The next chapter will give a definition of railway system resilience, and a visual method based on the railway simulation is proposed to evaluate and analyse delay propagation in the rail network.

CHAPTER 3

METHODOLOGY OF DELAY PROPAGATION VISUALISATION

During railway operations, it is inevitable that trains are delayed. The propagation of train delays in a railway network reflects the resilience of the system to different types of delays. A visualisation of the delay propagation can give dispatchers a clear understanding of the traffic states so that adjustments to train movements can be made. In this chapter, an introduction to railway system resilience is given, followed by some visualisation examples. A methodology of delay propagation visualisation is illustrated and a discussion of delay propagation with junction controls is also included in this chapter.

3.1 Introduction to railway system resilience

System resilience usually means the capability of a system to deal with change and adapt itself continually to survive the threats and shocks within critical thresholds [73]. Unclear and blurred definitions have been given by previous researchers on resilience. In this section, research views on system elasticity to delays are presented and a series of generally used terms are explained. A system resilience definition and classification are set out, followed by a discussion of the influence of factors based on a Quality of Service (QoS) measure [74].

3.1.1 Views on general terms

Stability

Loosely speaking, a solution of a dynamical system is deemed to be stable if all perturbations to the system result in a new solution that stays “close” to the original solution for all time [75]. In the railway system, it is usually used to describe the ability of the railway system to compensate for delays, usually perturbations, and results in an asymptotically stable behaviour with respect to the original schedule/ service pattern. [76]

Robustness

In a general way, robustness can be defined as the ability of a system to withstand the variations of parameters or changes in operational condition. A system is defined ‘robust’ if it is able to maintain its functionality under the incidences [75]. The idea of robust timetable should tolerate a certain degree of uncertainly during operations, in other words, they should be able to absorb some dynamic variations caused by both external and internal factors.

Even though much research on robustness in railway systems has been pursued, the concept of robustness remains vague and uncertain. Some definitions have emphasized the ability to preserve some level of solution quality. for example, Carey and Carville [77], define the robustness of a solution with respect to its ability to preserve the solution quality, like computation time. Another definition of robustness describes a timetable as robust if the system can survive threats without significant modification [78]. This work also defines a number of robustness levels (see Figure 3-1). Considering the differences between the retrieved timetable and original timetable, the associated modifications that should be made to cope with incidents range from level 0 (do nothing) to level 5 (cancel trains). In most cases, the main disruptions occur in level 0 due to small perturbations. The higher the level, the more rescheduling strategies are needed.

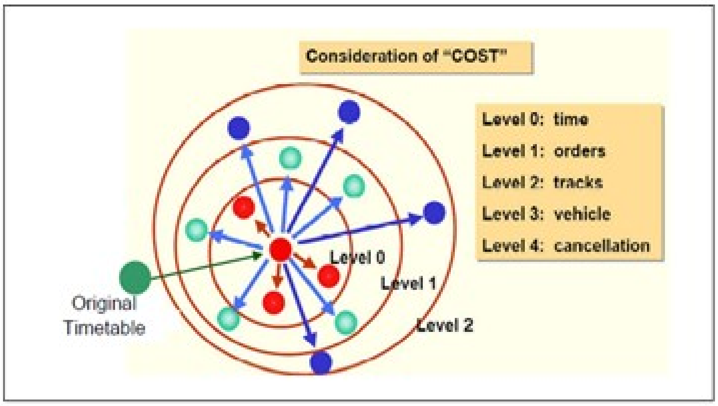


Figure 3-1: The main levels of robustness [78]

The robustness of the railway system indicates that the dynamic variations found in railway operation leads to an asymptotically stable behaviour of the railway system and the initial schedule plan/ service pattern is restored [79].

Recoverable robustness

Recoverable robustness is proposed based on a basic definition of robustness, with more flexibility added. Liebchen provides a precise concept of recoverable robustness for an optimisation problem [80]. Firstly, the original optimisation problem (O), the interrupted scenarios (S) and the limited recovery possibilities (R) need to be defined. A solution x for an optimisation problem O is recoverably robust if in all situations that may happen according to S, a feasible solution x can be found by one of the approaches given in R.

Several recoverable robustness models are built with boundaries for the action of recovery or limitations on computational power [80, 81]. For example, in [81], strict robustness models are the cases in which there are no recovery capabilities, however, the recoverable robustness models have possible new solutions computed by a recovery algorithm which must not deviate too much from the original solution, s, according to a distance, d.

The recoverable robustness of the railway system indicates that the imperfect situation found in railway operation leads to an asymptotically stable behaviour of the railway system with respect to a new schedule/service pattern with the same volume of traffic as the original solution.

Recoverability

Recoverability refers to the ability to restore a system with recovery actions when a disruption has occurred. For the railway, a recoverable timetable/service results in an asymptotically stable behaviour of the railway system with respect to a reduced service pattern with a lower volume of traffic than the original solution [75].

Table 3-1 shows the differences of four definitions. For each definition, the delays it can handle, the results after the recovery, and recovery strategies are represented and compared.

Definitions	**Delay situations**	**Results**	**Recovery strategies**
Stability	Perturbations	The original schedule/ service pattern.	No recovery actions.
Robustness	Minor disruptions	The original schedule/ service pattern.	Traffic management strategies.
Recoverable robustness	Disruptions	A new schedule/service pattern with the same volume of traffic as the original solution.	Traffic management strategies.
recoverability	Major disruptions	A reduced service pattern with a lower volume of traffic than the original solution	Operational management strategies.

Table 3-1: Differences of four definitions

3.1.2 Resilience classification and definition

In this research, we use resilience to describe the ability of a system to withstand stresses, pressures, perturbations, unpredictable changes or variations in its operating environment without loss of functionality [75]. Here, the term resilience is used to represent the ability of the railway system to deal with delays. Generally, railway systems can deal with delays in three ways: (1) systems can absorb delays automatically without active train rescheduling (i.e. no action being required by the traffic controller); (2) active train rescheduling or reordering strategies can be implemented to prevent the propagation of minor disruptions and aid the recovery of the system; and (3) adopt operational management measures, such as train cancellation, train re-allocation etc. when major disruptions occur.

In this work the term resilience is subdivided into three categories. This is based on the way in which the interacting services operating on the system respond to a given primary delaying incident (for example, a signal failure, dwell time delay, emergency speed restriction), and measures the return of the system to an on-time state. The definition of the on-time state will differ between and within railway administrations.

The definitions arrived at by the author and used throughout this thesis are [82]:

- A set of interacting services is said to be **stable** in response to a particular incident if the system recovers to an on-time state without the implementation of active traffic management measures;
- A set of interacting services is **robust** in response to a given delaying incident if active train rescheduling or reordering strategies must be implemented to allow the system to

return to an on-time state, without the requirement to cancel any of the trains involved in or affected by the incident;

- A set of interacting services is **recoverable** in response to an initial delaying incident if the system requires the introduction of major operational management measures, such as train cancellation or rolling stock reallocation, to return to an on-time state.

An important question is what kind of resilience strategy should be used to manage a transport network which is subject to a continued growing intensity of demand and a variety of disruptions? Taking into account the previous research that has considered a range rescheduling approaches to solve delay recovery problems, the specific aim of this research is to develop an approach that provides robustness.

3.1.3 Influence factors

The level of resilience should be related to the capacity of the system, the type of disruptions and the application of control strategies. Before a discussion of the influencing factors of railway resilience, a decomposition of the QoS (Quality of Service) concept is shown in Figure 3-2 below [74]. This general high level measure Quality of Service diagram is used as an indication of the overall performance of the railway system. This work has been presented in the "ON-TIME" project deliverables, and the author was enrolled in this project [74].

At the top it shows the key performance indicators (KPI) for QoS and the associated quantitative key measures. It covers transport volume, journey time, connectivity, punctuality, resilience, passenger comfort, energy and resource usage. It is desirable for the railway systems to be optimal in terms of all the indicators, however, trade-offs need to be made in practice due to the various constraints in real life railway operations. The bottom portion of the diagram shows the static and dynamic factors that influence the QoS. On the engineering side, the factors affecting Quality of Service can be broken down into capability and dependability. Capability covers all the "static" elements that are relatively hard to change, such as rolling stock, infrastructure, timetable and operational rules. Dependability includes all the dynamic components of the system: traffic management, operational management, human factors, system maintenance and environmental factors. These components can be modified over a shorter term in practice.

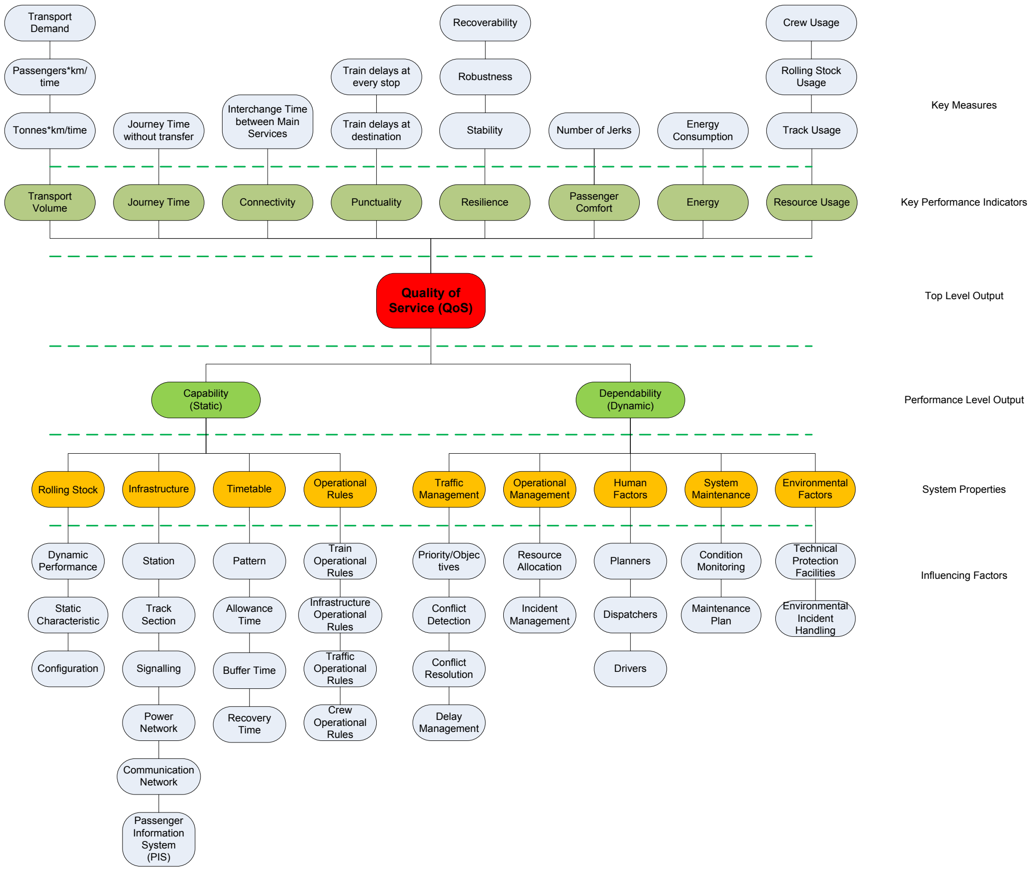


Figure 3-2: Quality of Service Diagram [74]

Based on the Quality of Service diagram (see Figure 3-2), there are many factors that may influence the system resilience. Different strategies to achieve different categories of resilience are also sought: to ensure the stability of the system, in this case train rescheduling activities are not required to recover perturbations, thus only off-line timetabling is needed. To achieve robustness, with regard to off-line timetabling, real-time traffic management is also required to help recover minor disruptions. For recoverability, real-time operational management is added to solve problems caused by major disruptions. In this research, the main aim is to manage with delays caused by minor disruptions through the use of real-time traffic management.

3.2 Existing delay propagation models

In recent decades, a considerable number of models to represent delay propagation have been proposed. These models reflect the impact of various factors on the propagation of train delays and provide a prediction of the effect of knock-on delays. These models can be divided into two categories: analytical methods and simulation-based methods.

3.2.1 Analytical models

Weigand described the variation of delays as either negative, zero or positive with an exponential distribution [83]. Based on the primary and secondary delay distributions, as well as the running time supplement and buffer time in the timetable, he derived a model for the distribution of the delays in a large-scale railway network. Based on this model, Muhlhans considers a generic cumulative probability distribution of delays and then computes the exit delay distribution by convoluting the primary delay and secondary delay distributions [84].

Carey and Kwiecinski computed the distribution of arrival and departure delays of trains at successive stations with recursive substitutions starting from the departure station with for a given departure delay [85]. Later work also evaluated the knock-on delays occurring on a single line caused by route conflicts by non-linear regression and heuristic approximations [27]. Considering the knock-on delays caused by insufficient headway, speed constraints and late connections, Higgins and Kozan presented an analytical model to quantify the expected delay for individual passenger trains and signal blocks in a railway network [86]. However, this model does not considers the knock-on delays caused by route conflicts and rerouting operations.

Later, Yuan proposed a probability model that provided a realistic estimate of knock-on delays [87]; this model can reflect the dependencies of dwell times and speed fluctuation. In 2008, D'Ariano evaluated delay propagation by decomposing a long time horizon into tractable intervals, and train conflicts are detected and solved in each time interval using advanced Conflict Detection and Resolution with Fixed Routes (CDRFR) algorithms [54].

3.2.2 Simulation-based models

Simulation techniques are widely used to study the primary delays and knock-on delays in a complex railway network. The interaction of these delays and the influence of different factors can be directly captured in the simulation models.

Petersen presented the first simulation model for rail lines in 1982. In his model, rail lines are divided into track segments between adjacent switches and algebraic relationships are developed to represent the model logic [88]. Then Dessouky used a simulation methodology to consider both single and double lines, making the approach insensitive to the size of the rail network [89]. This model considers headways, speed limits and actual train lengths in order to determine the track configuration that minimizes congestion delay to trains. In 1999, Krueger developed a simulation tool to support a regression model to allow the study of the relationship between train delay and traffic volume with network parameters, traffic parameters and operating parameters [90].

The analytical models usually make a lot of assumptions in order to simplify the complexity of the problem within solvable bounds. Thereby they are different to real-life operations to some extent. The simulation models, on the other hand, can describe the relationships between delays more accurately by capturing the interactions between the services and impacts from the operating parameters.

3.3 Delay propagation visualisation

In this section, a visualisation method with simulation techniques is adopted to describe and estimate the delay propagation. This work has been presented in the "ON-TIME" project deliverables [74], [82], [91].

3.3.1 Introduction to the visualisation method

The visualisation approach adopted here shows the lateness and change in lateness of all trains in the system, as well as the delays to individual trains, all with respect to time. These values are calculated as described in the following paragraphs.

Simulations which have been run with the HERMES simulator (see Appendix A) over a defined time window, *T*, are analysed as part of a post processing task. All trains operating within the system during *T* are considered. The times at which all trains under consideration pass each designated timing point are recorded; the time at which train i passes its jth timing point is defined as t_{ij}. For all train journeys the lateness of train i at its jth timing point, L_{ij} is calculated as the delay in seconds between the actual time and the scheduled time, t_{ij}^{s}:

$$L_{ij} = \max(0, \quad t_{ij} - t_{ij}^{s})$$

Equation 3-1

The Equation 3-1 results in a discrete set of observations of the system. A continuous function for each train service is defined here by using the most recent delay recorded along the journey to represent the current delay. At the time t, if the service i has passed the kth observation point but hasn't reached the $k+1$th observation point, the current delay for this service, $L_i^c(t)$, is the most recent delay value recorded, which is equal to L_{ik}.

$$L_i^c(t) = L_{ik}, \quad when\, t_{ik} \leq t < t_{i,k+1}$$

Equation 3-2

The scheduled time t_{ij}^s is calculated from an initial as-timetabled run of the simulator with no delays. The actual times at timing points are taken from the simulated delayed scenario under consideration. This is an event driven approach because the lateness is updated whenever a new timing point is passed. It is generally appropriate to plot both the lateness of individual trains (those that are delayed) against time, and the sum of lateness of all n trains within the system, $L(t)$.

$$L(t) \; = \sum_{i=1}^{n} L_i^c(t)$$

Equation 3-3

An increase or decrease in the delay for a train is considered from one timing point to the next, and then the change in lateness is introduced between timing points for each train journey:

$$C_{ij} \; = \; L_{ij} - L_{ij-1}$$

Equation 3-4

C_{ij}: The change in lateness of train i at its jth timing point;

L_{ij}: The lateness of train i at its jth timing point;

L_{ij-1}: The lateness of train i at its $j-1$th timing point.

A continuous function is defined by using the most recent value recorded along the journey. At the time t, the kth observation point is passed until the $k+1$th observation point is reached, the current change in lateness, $C_i^c(t)$, is the most recent value recorded:

$$C_i^c(t) = C_{ik}, \quad when\, t_{ik} \le t < t_{i,k+1}$$

Equation 3-5

We then consider the change in lateness in the system as a measure of whether the delay to the system is increasing or decreasing, *C(t).*

$$C(t) \ = \sum_{i=1}^{n} C_i^c(t)$$

Equation 3-6

Examples of these measures are given in Figure 3-3. The purple line illustrates the total lateness within the network, calculated as the sum of the delays from individual trains. The lateness of individual trains is also given. The red line shows the change of the total lateness. From the visualisation, the propagation of the delays and the rate of change can be seen, as well as the contribution of individual trains to the overall system lateness.

3.3.2 Key performance indicator [91]

There are three significant characteristics that can be used to evaluate the performance and quantify the delays that occurred during a delayed scenario derived from this visualisation method. They are described as the key performance indicators (KPIs) and can be derived from the visualisation graph described above (see Figure 3-3 (a)). They are maximum lateness, time to recover and integral of delay.

Maximum lateness indicates the maximum total lateness of all journeys running in the network within the time window T; it indicates the worst delay situation. **Time to recover** measures how quickly the system returns back to a situation with no delays. The third main measure, **Integral of delay**, measures the area under the lateness curve. Combined with the other two KPIs, this measure gives a quantitative summary of the delayed situation.

It can also be given as a proportion of the maximum possible integral of delay as: **Integral of Delay proportion** = Integral of delay/ (Maximum lateness* Time to recover). This value gives an indication of how severe the delays were relative to the maximum lateness throughout the period of delay.

3.3.3 Visualisation examples

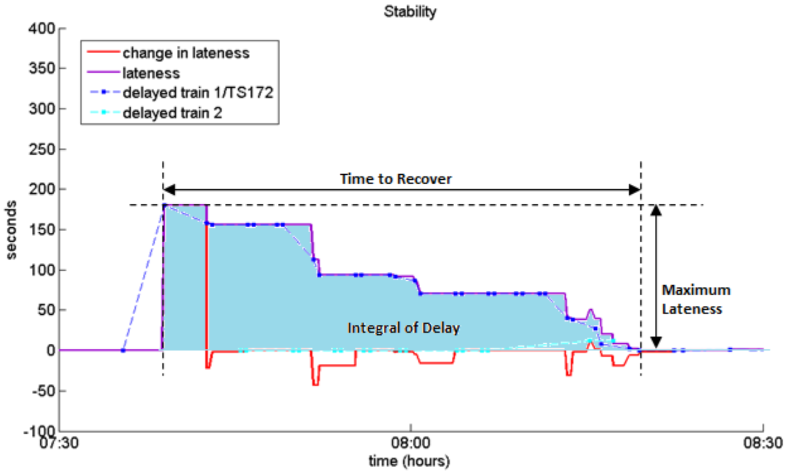
Figure 3-3 shows the response of a simple example system to three different input delays using the visualisation method described above. This example is taken from trains running over a short section of the East Coast Mainline. Further details of this part of the network are given in Section 3.4.2 where a case study is analysed in detail; here the objective is to demonstrate examples of the three levels of resilience defined previously.

In the first case, train TS172 is delayed at Welwyn Garden City station for 180 seconds due to an extended dwell time, in the second a 360 second delay is caused by a signal failure, and in the final graph an additive delay occurs along the journey of one train due to a partial engine breakdown and consequent loss of power. From the visualisation, the propagation of delays and the rate of change can be observed. In this simulation case an on-time state is said to be when there are ≤0 seconds of lateness.

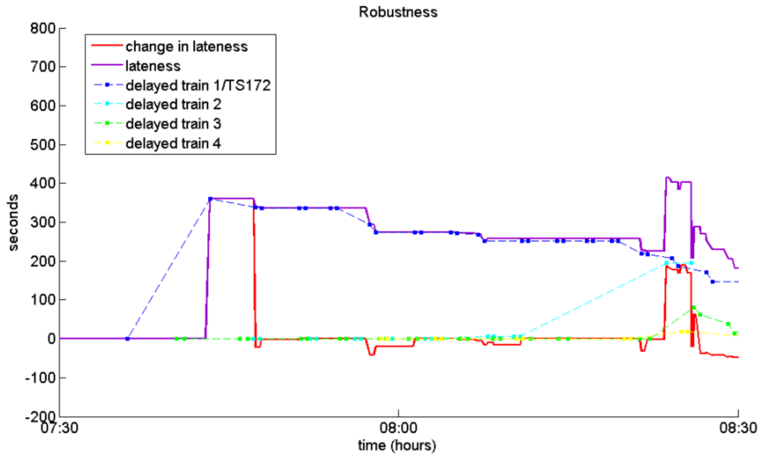
In Figure 3-3 (a) the 180 second delay is recovered and absorbed quickly by the system, causing only a minimal delay to one following train. No active recovery strategy is applied and the system recovers from the disruption by itself. According to the resilience classification proposed in the previous section, this is an example of a stable response to the initial extended dwell time delay.

In Figure 3-3 (b), an initial 360 second delay takes approximately 1 hour for the system to recover. The initially delayed train TS180 causes knock-on delays to three following trains, hence enlarging the total delays in the system. To improve the resilience of the system and allow the system to recover to an on-time state more quickly, a reordering strategy could be applied to allow this train to pass the junction before the initially delayed train, thus reducing the total delays in the system using a real-time rescheduling method. Such a system response to the input delay could be considered to be robust.

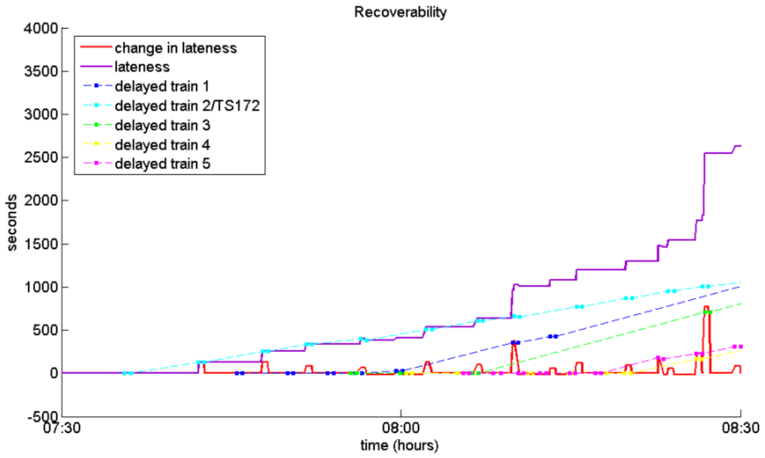
Lastly, in Figure 3-3 (c), an engine breakdown on train TS172 causes cumulative knock-on delays. The delays severely influence the following trains on the same line and the system cannot recover from the situation without the application of operational train management, in this case the cancellation of service TS172, which would allow the system to recover. The requirement for a train cancellation places this system response to the underpowered train in the recoverable category of system resilience.


Stability
change in lateness
lateness
delayed train 1/TS172
delayed train 2
Time to Recover
Maximum Lateness
Integral of Delay
seconds
400
350
300
250
200
150
100
50
0
-50
-100
07:30
08:00
08:30
time (hours)

(a)


Robustness
change in lateness
lateness
delayed train 1/TS172
delayed train 2
delayed train 3
delayed train 4
seconds
800
700
600
500
400
300
200
100
0
-100
-200
07:30
08:00
08:30
time (hours)

(b)



(c)

Figure 3-3: Visualisation of system resilience for the scenarios.

3.4 Case study of delay propagation with junction controls

3.4.1 Introduction to algorithms at junctions

A railway junction is a place where two or more rail routes meet or diverge, facilitated by points and signals. When delays occur in a rail network, conflicts at junctions are common, especially at complex junctions and / or on busy networks, leading to a propagation of the initial delay. In a delayed situation, the use of an appropriate junction control method (JCM) can help to assign the best possible train sequence through the junction and thus improve the quality of service of the railway. It should be noted that any change of sequence at a given junction will have consequences for other services since trains are moving out of their timetabled order. Thus, the selection of the best JCM in a delayed scenario is a complex task. A simple case study is selected for investigation here to demonstrate the usefulness of the categorisation of resilience and the new visualisation method in analysing train control approaches.

Four fundamentally different JCMs are chosen for study they are: First Come First Served (FCFS), Timetable Order Enforced (TTE), Fast Line First (FLF) and Busy Line First (BLF). In FCFS the first train to arrive at the junction is allowed to pass first, TTE forces all trains to pass the junction strictly in the timetabled order. In the case of a conflict, FLF allows the faster train (running a faster schedule with fewer stops on the fast line) to pass the junction first, while BLF prioritises the train that has the greatest number of following trains within 10 km of the junction. A summary of this four JCMs is presented in Table 3-2.

JCMs	**Description**
FCFS	First Com First Served: Prioritise the first train arriving at the last signal before a junction.
TTE	Timetable Order Enforced: Strictly enforce the timetabled order for passing a junction.
FLF	Fast Line First: Prioritise a train travelling on the fast line over a train travelling on the slow line.
BLF	Busy Line First: Prioritise the busiest line, that is, the train from the line with the greatest number of following trains within a given distance.

Table 3-2: JCMs used in this chapter

3.4.2 Design of experiments

Model description

The experiment simulates trains running on a part of the East Coast Main Line (ECML), which is an electrified and high speed (up to 125 mph) railway linking London with Yorkshire, the North East and Scotland [5, 6]. The experiment is carried out with a simplified version of the timetable published by Network Rail [92]. The aim is to demonstrate the visualisation of the system resilience and show the different behaviours that can be observed when different junction control methods are applied.

The simulation is commenced from the first train of the day, but in order to observe the influence during peak time, the period 7:00 am to 9:30 am was chosen as the observation period. The map of the case study area presented in Figure 3-4 (a) is chosen from part of the ECML, bounded by Welwyn Garden City and Hertford (on the Hertford loop) in the north, and the terminus London stations at King's Cross and Moorgate [93]. Due to the fact that some services will leave the section of the network being studied at the point between Finsbury Park station and London King's Cross station, five directions must be defined within the network to reflect the different routes that different services may take, as shown in Figure 3-4 (b). Direction No. 5, marked with a dotted line, indicates the direction where some trains leave and enter the system from the Midland Road Junction.

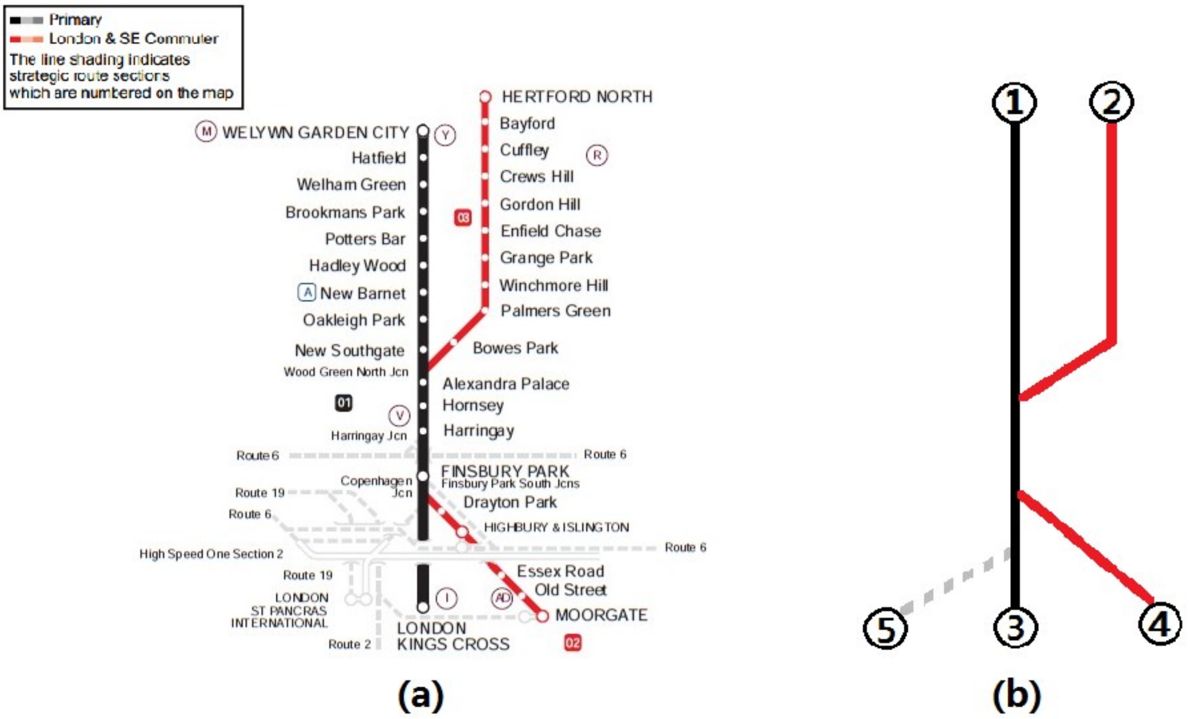


Figure 3-4 (a): Map of the simulated area [93]; (b): Five defined directions.

The experimental timetable used here runs ten services from Welwyn Garden City and Hertford to London and Moorgate. Table 3-3 shows the timetable and the service description of the sample case study. The maximum speed allowed for the fast train is 100km/h and for the slow train it is 75km/h. Four out of the ten services begin on the fast line and the rest are on the slow line. Figure 3-5 shows which track each of the services take and the locations of junctions.

Service No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Service No.**	**8**	**9**	**10**
Train ID	TS152	TS157	TS158	TS172	TS173	TS190	TS203	**Train ID**	TS156	TS175	TS202
Type	Fast	Slow	Fast	Slow	Fast	Fast	Slow	**Type**	Slow	Slow	Slow
Passing junctions	J3	J1, J3	J1, J3	J1, J2	J3	J1, J3	J1, J2	**Passing junctions**	J2	J2	J2
Start time	07:17:00	07:21:00	07:26:00	07:33:00	07:35:00	07:54:00	08:03:00	**Start time**	07:23:00	07:38:00	08:01:00
								Hertford North	07:25:00	07:40:00	08:05:00
Welwyn Garden City	-	07:26:00	-	07:36:00	-	-	-	Bayford	07:30:00	07:45:00	-
Hatfield	-	07:30:15	-	07:40:30	-	-	08:10:30	Cuffley	07:34:39	07:49:39	08:13:00
Welham Green	-	-	-	07:44:00	-	-	08:14:00	Crews Hill	07:37:40	07:52:40	-
Brookmans Park	-	-	-	07:46:30	-	-	08:16:30	Gordon Hill	07:41:01	07:56:01	08:18:00
Potters Bar	-	07:37:00	-	07:50:38	-	-	08:20:38	Enfield Chase	07:43:29	07:58:29	08:20:30
Hadley Wood	-	-	-	07:54:00	-	-	08:24:02	Grange Park	07:45:29	08:00:29	-
New Barnet	-	-	-	07:57:09	-	-	08:27:11	Winchmore Hill	07:47:29	08:02:29	08:23:05
Oakleigh Park	-	-	-	07:59:38	-	-	08:29:40	Palmers Green	07:49:59	08:04:59	08:26:00
New Southgate	-	-	-	08:02:46	-	-	08:32:48	Bowes Park	07:52:29	08:07:29	-
Alexandra Palace	-	07:45:00	-	08:05:30	-	-	08:35:31		07:55:29	08:10:29	08:30:30
Hornsey	-	-	-	08:07:38	-	-	08:37:39		-	-	08:32:42
Harringay	-	-	-	08:09:33	-	-	08:39:35		-	-	08:34:33
Finsbury Park	-	07:49:30	07:39:21	08:13:11	07:52:09	08:09:21	08:43:16		08:01:59	08:16:59	08:38:38
Drayton Park				08:16:06			08:46:11		08:04:58	08:19:58	08:41:37

Highbury & Islington				-			-		-	-	-
Essex Road				08:19:30			-		08:08:29	08:23:29	08:44:31
Old Street				08:23:12			08:53:14		08:12:11	08:27:11	08:48:13
Moorgate				08:27:01			08:57:03		08:17:00	08:32:00	08:52:02
London King’s cross	07:36:40	07:58:00									

Table 3-3: The timetable and service description of the sample case study

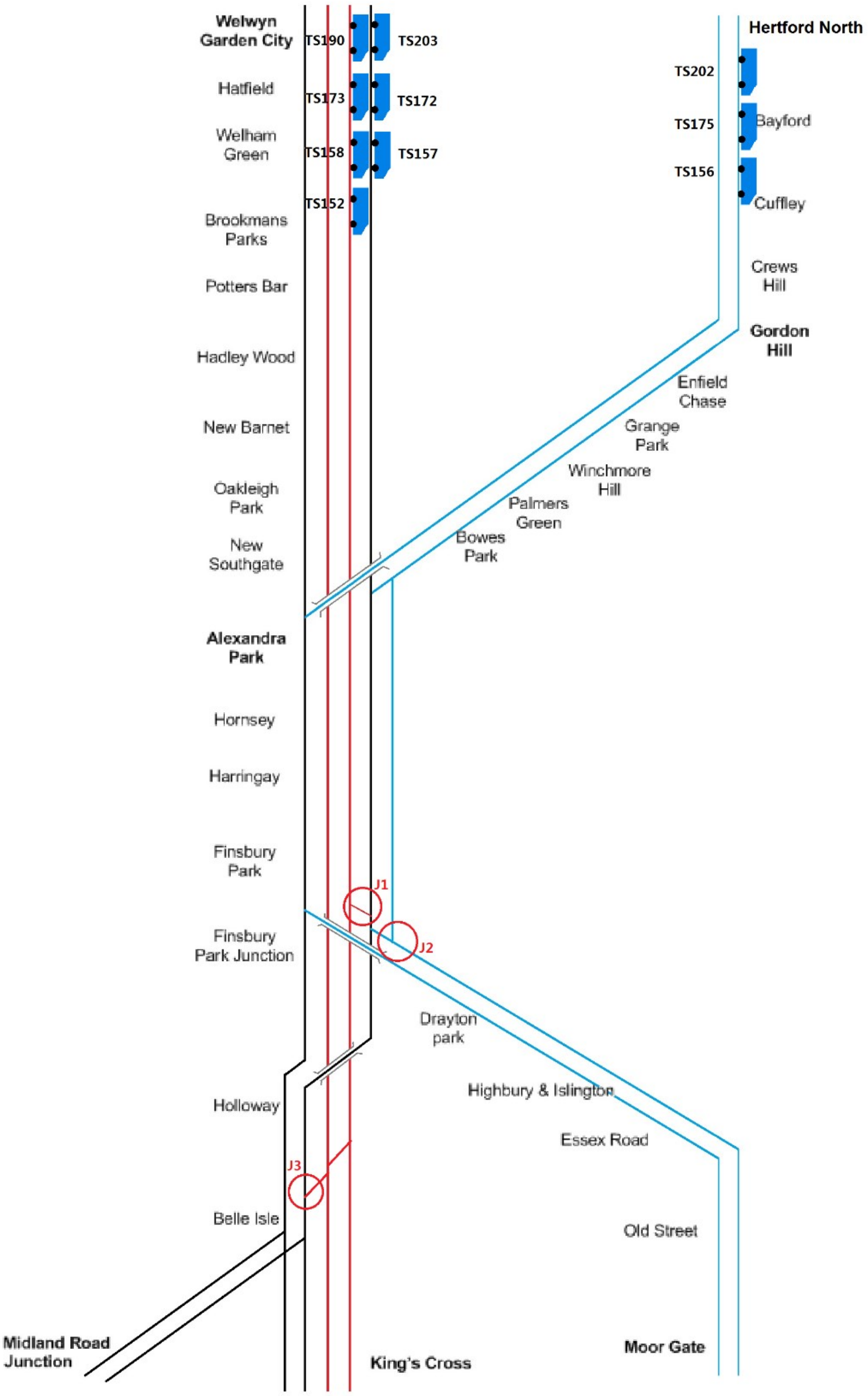


Figure 3-5: Infrastructure with junction specification

Scenario design

In the delayed scenario a single train TS158 heading for London King's Cross is delayed for 10 minutes from 07:27 to 07:37 at Welwyn Garden City because of signal failure. This size of delay is selected to make trains conflict at the junctions and allow the effect of different JCMs to be visualised. Figure 3-6 shows the effect of the disrupted scenario and the propagation of delays with four different JCMs: TTE, FCFS, FLF and BLF in place.

3.4.3 Analysis of Results

As can be seen from all the Figure 3-6(a) – (d), the 10 minute delay experience by the fast train TS158 is recovered slightly before it arrives at the junction area, and it therefore delays the subsequent service TS157.

Figure 3-6 (a) and 3-7 (c), which are using TTE and FLF junction control methods respectively, show that the trains follow very similar trajectories at the beginning of the delay period. However, towards the end of the observation period the FLF strategy recovers more quickly as TS173, the faster service, is allowed to pass junction 2 before TS157.

Figure 3-6 (b) shows that the FCFS strategy provides a good control strategy. A significant difference with this strategy is that only two trains are delayed in the network when applying a FCFS strategy, rather than three trains delayed using the other approaches. This results in the maximum total delay being least for the FCFS case.

Figure 3-6 (d) representing the BLF junction control strategy provides the worst performance, with the recovery time, maximum lateness and integral delay all larger than in the other cases. In this case the BLF strategy causes TS152 to be significantly delayed as it is given a very low priority in passing junction 3 due to the other lines being busier at the time it arrives at this junction.

The example shows that the resilience of the system is affected by changing the JCMs and demonstrates the usefulness of the visualisation approach to assess the performance of each strategy.

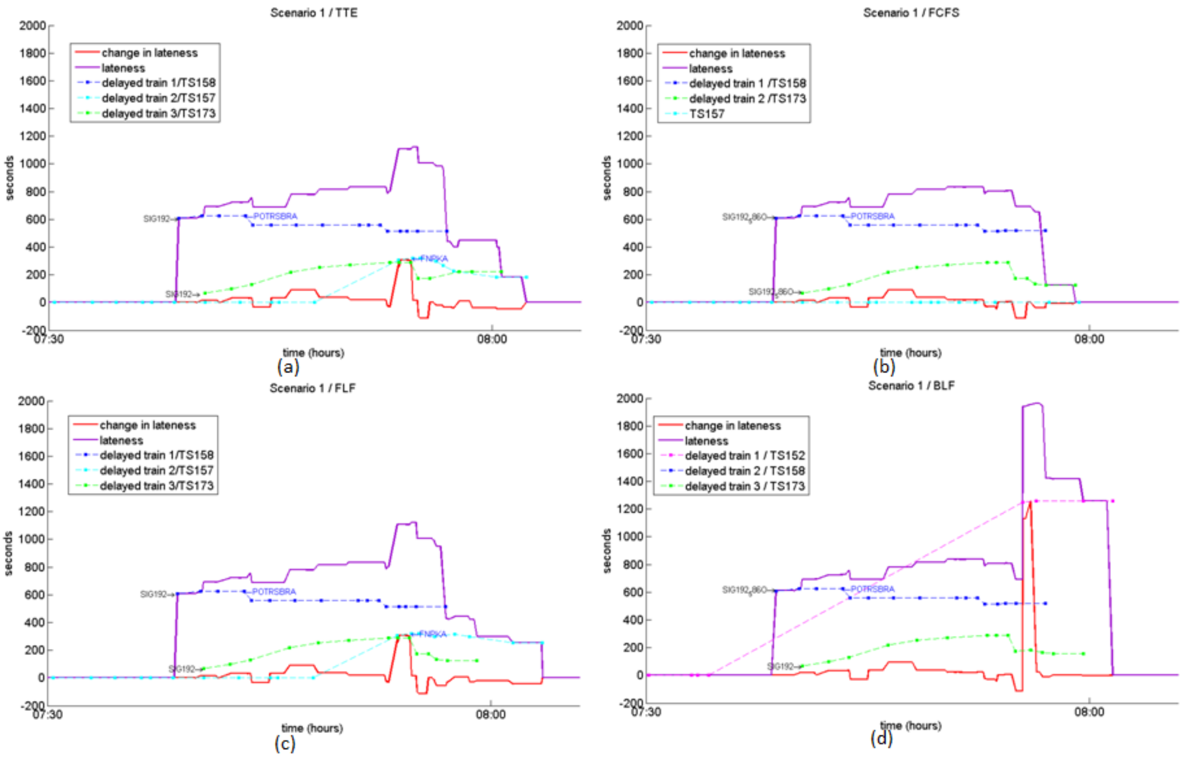


Figure 3-6 (a): The propagation of delays with TTE; (b): The propagation of delays with FCFS; (c): The propagation of delays with FLF; (d): The propagation of delays with BLF.

3.4.4 Summary

From each of the graphs in Figure 3-6, the three KPIs - maximum lateness, time to recover and integral of delay - are quantified and used to construct a high level representation of the performance of each strategy in terms of the KPIs; these are shown in Figure 3-7. Table 3-4summarises these performance measures which include the three KPIs and the proportion of integral of delay. It gives an estimation of the shape of the delay graph, and allows the three KPIs to summarise the characteristics of the visualisation.

In order to more easily compare the integral of delay values, the integral of delay normalisation is also given.

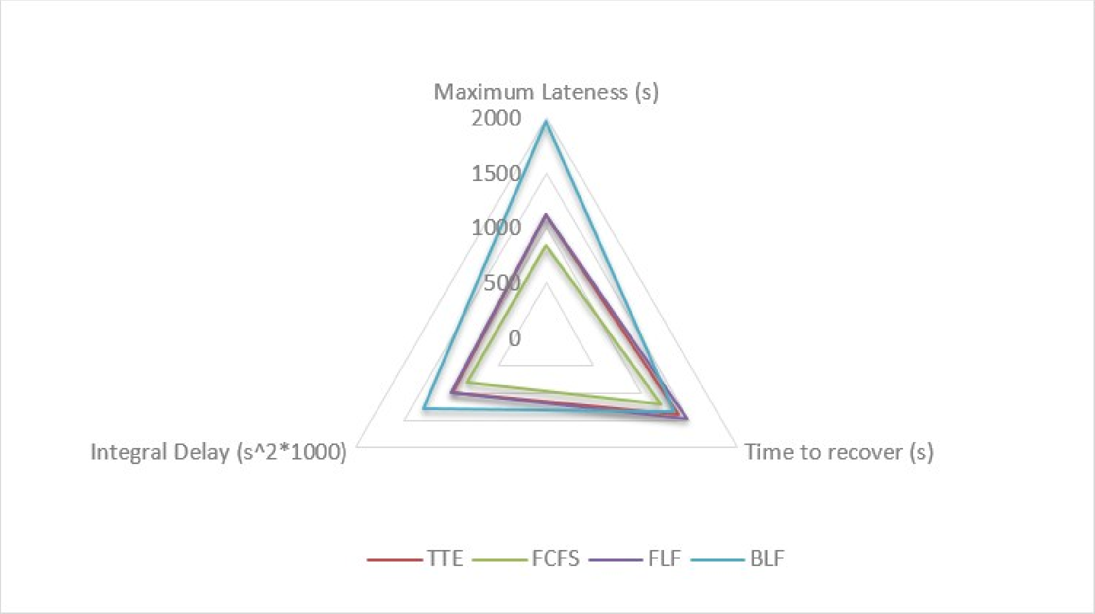


Figure 3-7: Triangle representations with three KPIs for the four JCMs studied in the case study

JCM	Maximum Lateness (s)	Time to recover (s)	Integral Delay (s^2) (normalised value (%))	Proportion of Integral Delay (%)
TTE	1121	1390	987066 (76.73)	63.4
FCFS	834	1203	822198 (63.92)	82.0
FLF	1121	1477	999273 (77.68)	60.4
BLF	1964	1342	1286344 (100)	48.8

Table 3-4: Three KPIs and ID proportion of four JCMs

The triangle representation allows a direct comparison of the KPIs for each of the junction control methods. There is a significant difference in performance between all the methods, except FLF and TTE which are very similar to one another in terms of the KPIs. The smallest triangle representation, shown in green is produced by the FCFS strategy, indicating the best recovery from the initial delay. The largest triangle representation is produced by BLF, which indicates the worst service with the most significant effect to the train services, although the time to recover is comparable with TTE and FLF. The KPI triangle representation shows a summary of the propagation of delays and affords an easy comparison of the performance of the junction control recovery strategies in response to the input delay.

3.5 Conclusion

In this chapter, the resilience of a railway system can be defined according to the response of a system to a delay, and it is allocated one of three levels: stable, robust and recoverable. Stability indicates that the system can absorb delays automatically without active train rescheduling; robustness shows that active train rescheduling or reordering strategies must be implemented to prevent the propagation of minor disruptions and aid the recovery of the system; and recoverability represents the situation where the system needs active operational management intervention, such as train re-allocation, to handle some major disruption. Considering the use of rescheduling approaches to solve delay recovery problems, the aim of this research is to achieve robustness of the system.

In the recent decades, analytical methods and simulation methods are used to model the propagation of train delays. Those models reflect the impact of various factors and give a direct prediction of knock-on delays. An overview of the existing models is given and a visualisation and quantification method is introduced with simulation techniques to monitor the delay propagation in the rail network. Examples of the three levels of resilience are shown in the graphs created using the visualisation method.

Using the Graffica HERMES simulator, a delayed scenario was simulated and compared to an as-timetabled run. Then the propagation of delays with respect to time was produced by

MATLAB to visualise the resilience of the system. Lateness and change in lateness of all trains in the system with respect to time, as well as the individual lateness of delayed trains, are shown in the visualisation. Three Key Performance Indicators (KPIs) are introduced for the measurement of the propagation of delays and resilience of the system, which are maximum lateness, time to recover, and integral of delay.

This visualisation method can also be used to analyse and evaluate the different control strategies. A comparison of the response of the different JCMs to a delayed scenario is given as a demonstration of the visualisation method introduced here. Using this approach, information can be derived about the propagation of delays and overall resilience of the system in response to the different strategies. The comparison shows the different effects of junction controls on the resilience of the system. Furthermore, it shows that we can change the resilience of the system by changing the JCM.

CHAPTER 4

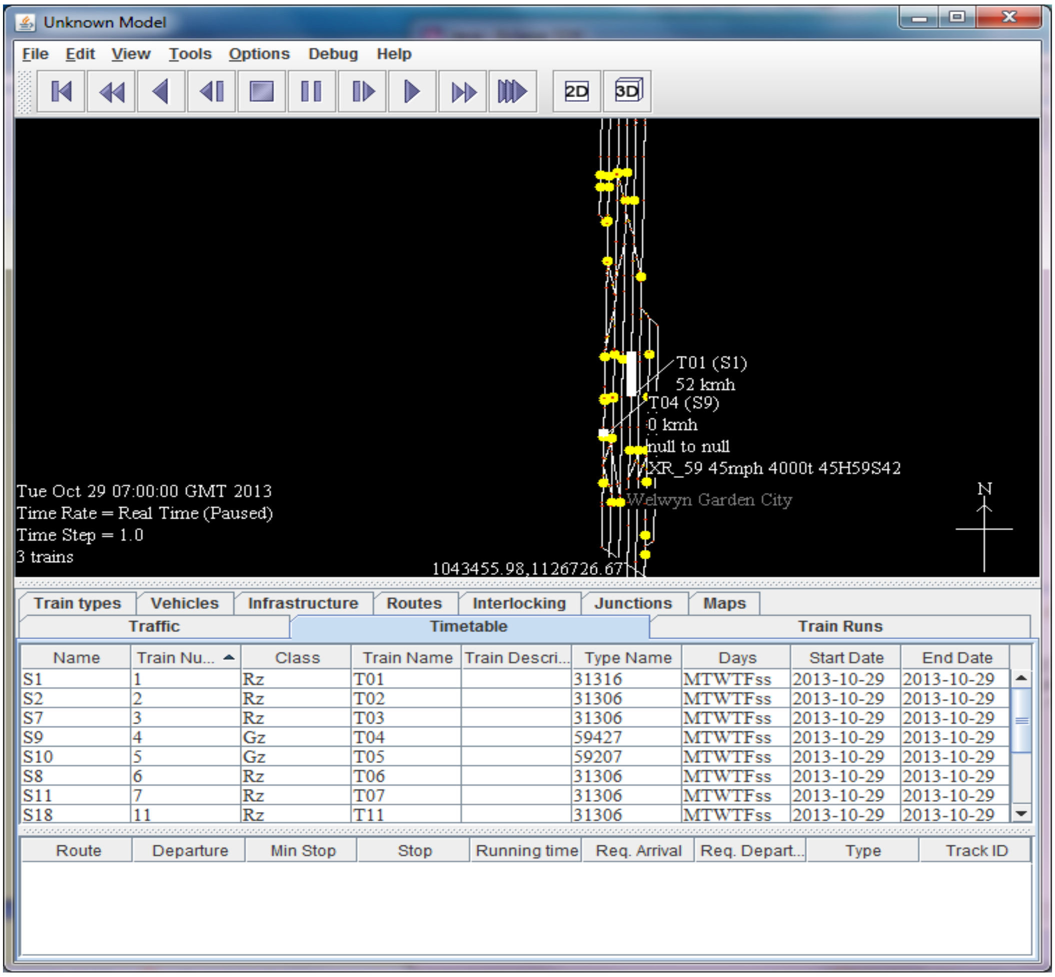
IMPLEMENTATION OF SIMULATION MODEL

4.1 Introduction to the simulators

4.1.1 BRaVE [94]

The BRaVE simulator is a railway network simulator written in Java at the University of Birmingham. BRaVE is an acronym for Birmingham Railway Virtual Environment. The simulator forms the core of the virtual railway laboratory at the Birmingham Centre for Railway Research and Education. Figure 4-1 shows the user interface of the simulator, which includes two main panels: the graphical panel and the information panel. The top side of the display in Figure 4-1 is the graphical panel, which shows a graphical view of the simulation model including infrastructure, signals and the trains running on the network. The graphical panel is updated when the simulation is running. The information panel is shown on the lower side of the display. The panel displays database tables that detail the current state of all of the entities in the simulation database. A tabbed window is contained in the panel, with tabs across the top allowing the different tables of data to be displayed. The information includes train types, vehicles, infrastructure, routes, interlocking, junctions, maps, traffic, timetable and current train runs.

BRaVE is highly flexible, allowing the modification of many parameters. For example, timetable information, trains and routing information are all editable in the simulation. Simulations may be run as per a timetable, or with a delay that can be inserted artificially. The main outputs of BRaVE are described in the following section; based on the outputs, a direct view of train movements and service performance can be shown to users.



```
Tue Oct 29 07:00:00 GMT 2013
Time Rate = Real Time (Paused)
Time Step = 1.0
3 trains
```

Name	Train Nu...	Class	Train Name	Train Descri...	Type Name	Days	Start Date	End Date
S1	1	Rz	T01		31316	MTWTFss	2013-10-29	2013-10-29
S2	2	Rz	T02		31306	MTWTFss	2013-10-29	2013-10-29
S7	3	Rz	T03		31306	MTWTFss	2013-10-29	2013-10-29
S9	4	Gz	T04		59427	MTWTFss	2013-10-29	2013-10-29
S10	5	Gz	T05		59207	MTWTFss	2013-10-29	2013-10-29
S8	6	Rz	T06		31306	MTWTFss	2013-10-29	2013-10-29
S11	7	Rz	T07		31306	MTWTFss	2013-10-29	2013-10-29
S18	11	Rz	T11		31306	MTWTFss	2013-10-29	2013-10-29

Route	Departure	Min Stop	Stop	Running time	Req. Arrival	Req. Depart...	Type	Track ID

Figure 4-1: Platform of simulator (example)

4.1.2 Single Train Simulator (STS)

The STS, developed at the University of Birmingham, is a single train simulator which is used for the study of the movement of a single train along the railway. In the current analysis, whether under automatic train operation (ATO) or driver control, four possible modes of single-train movement are assumed: powering, cruising, coasting and braking [95]. A typical journey between two stopping points can be defined by a sequence of the four modes. The coasting mode is usually used for energy saving, which is not considered in this research. To simplify the complexity of the problem, the trains are set to run as fast as possible, at maximum acceleration, maximum cruising speed within the speed limit and service braking.

The simulator can output the train trajectory, running diagram and energy consumption when some static parameters and dynamic parameters are given as the inputs. The static parameters, which are not changed during the modelling process, include the information of the infrastructure (e.g. line length, velocity limit, gradient), parameters of the train (e.g. train top speed and maximum acceleration), and station characteristics (e.g. station position, dwell time). The dynamic parameters are the acceleration and deceleration of the trains and the location of coasting points along the journey. By changing the inputs (e.g. infrastructure, timetable), an estimated train profile is given for the running estimation.

An example is given to compare the speed profiles (Figure 4-2) and the running diagrams (Figure 4-3) of the same train running in two different simulators based on the same timetable. Train S8 is selected from the case study shown later in Chapter 5. In Figure 4-2, the red line indicates the speed limit of the line, the black line indicates the speed profile obtained by BRaVE, and the blue line represents the speed profile obtained by the STS.

In Figure 4-3, the black line indicates the running distance according to the time obtained by BRaVE, and the blue line represents the running diagram obtained from the STS. It can be seen that the running diagrams obtained from the two simulators are very similar.

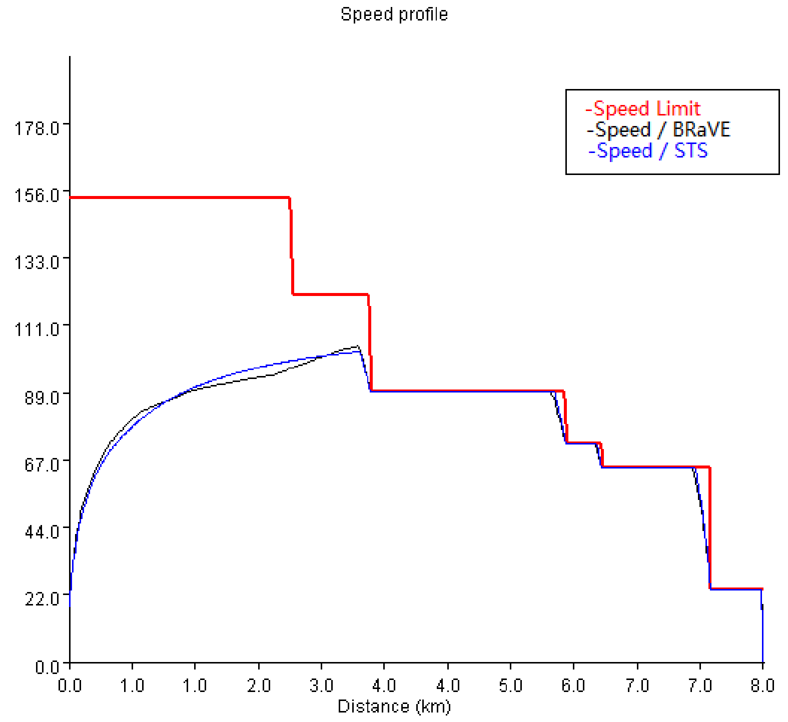


Figure 4-2: Speed profile of the train S8 in BRaVE and STS

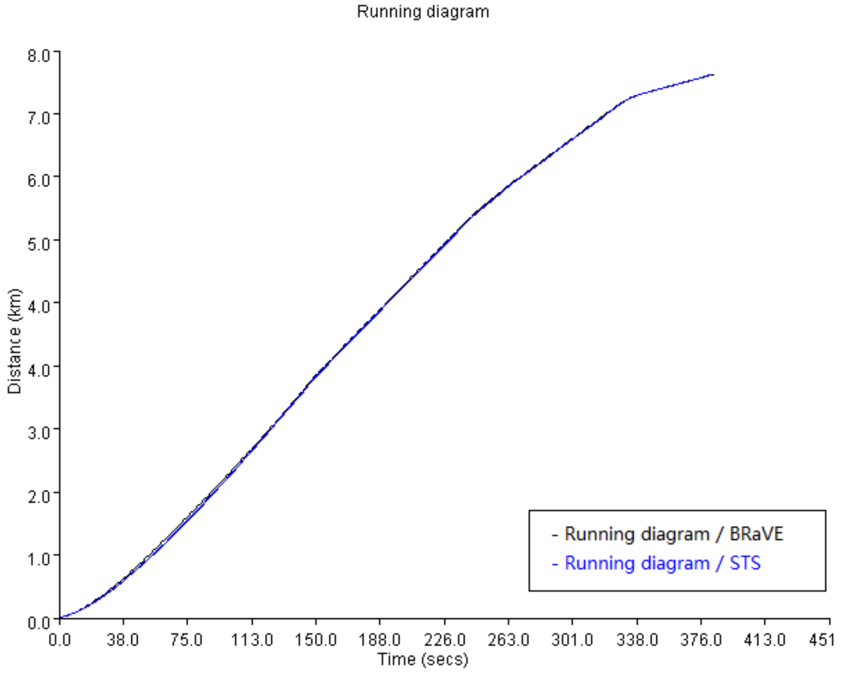


Figure 4-3: Running diagram of the train S8 in BRaVE and STS

Table 4-1 summaries the differences, advantages and disadvantages of these two simulators.

	BRaVE	**STS**
Difference	• Reference step BRaVE uses time (seconds) as its reference step; • Signalling BRaVE has signal timing points on the route setting, points switching and signal switching. • Vehicle model BRaVE models vehicles' performance with their given traction system characteristics.	• Reference step The STS uses length (metres) as its reference step; • Signalling The STS does not use signalling. Instead of signalling, the stopping points along the journey are known by the trains. • Vehicle model The STS models vehicles' performance with their given configuration parameters.
Advantages	• The simulation process of BRaVE is more close to reality, and the simulation results are more reliable. • The simulator can output various train running information to give a comprehensive view of train movement.	• The simulation process of the STS is simpler and quicker. The results are less accurate but still reliable. • The vehicles are easy to construct.
Disadvantages	Time consuming when simulating the railway network with a lot of trains running at the same time.	It generates a knock-on effect of many small timing differences; therefore, later trains have a greater difference.

Table 4-1: Comparison of BRAVE and STS

In summary, BRaVE is chosen as the simulation platform in this study for implementation of rescheduling approaches, due to its accurate and reliable simulation process and its various outputs. The STS is chosen for the train running estimation by the rescheduling algorithms because of its short calculation time; the results can be very close to the results obtained from BRaVE (the difference is less than 60 seconds) when the vehicles are well defined.

4.2 Implementation of algorithms

A Java interface in BRaVE has been provided to enable third party developers to use the real time information captured in the entity data model to drive new algorithms and to generate requests which can be fed back into the simulator.

The developer has provided a class document which implements an interface that is installed as a plug-in object to perform the necessary initialisation of the plug-in and to carry out regular updates as required by the implementation. The plug-in architecture is illustrated in Figure 4-4 below. BRaVE provides real-time data to the traffic manager and also receives traffic management requests from the traffic manager. The traffic manager (TM) creates a plug-in interface which is accessible to the entity data models, including train detection (TD) section, route, signal, point, service and train path. Algorithms provided by third party developers can be used as plug-in objects in the junction area, which is defined in the network model.

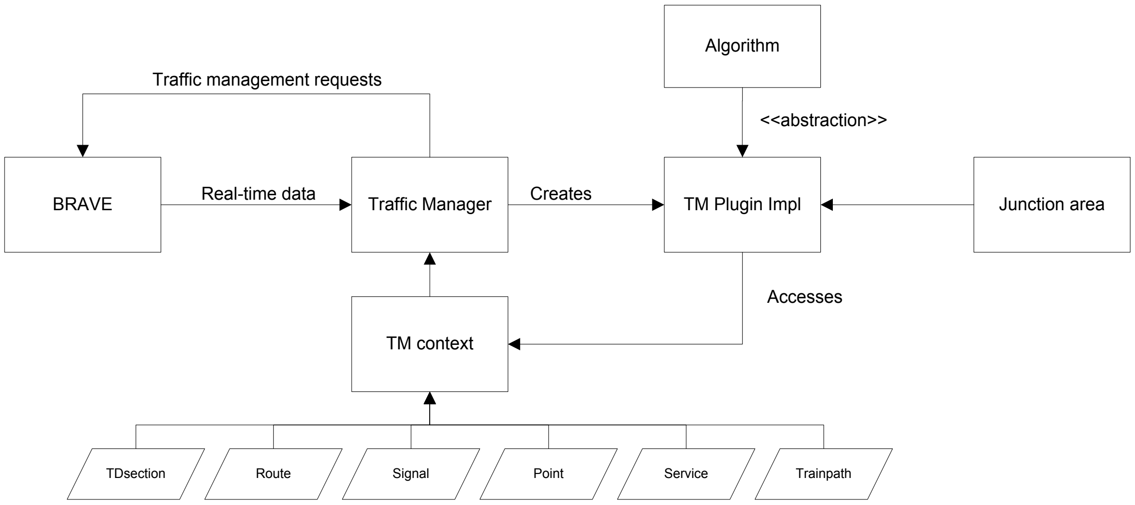
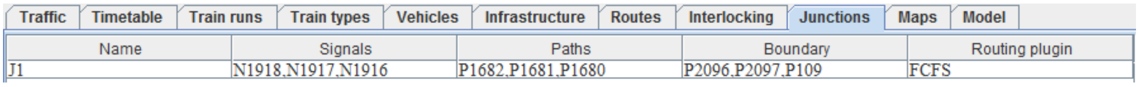


Figure 4-4: Traffic Management Plug-in Architecture

The junction panel (see Figure 4-5) is used to specify areas in the simulation model where an external signalling plug-in will control the signals in the network, via the API described above.

Junction definitions include junction name, junction signals, junction edge paths, junction boundary paths and routing plug-in.



Name	Signals	Paths	Boundary	Routing plugin
J1	N1918.N1917.N1916	P1682.P1681.P1680	P2096.P2097.P109	FCFS

Figure 4-5: Junction panel (example)

4.3 Train running estimation

All rescheduling methods consider the sequence of trains to pass into the junction area, so it is important to keep the estimated running time as close as possible to reality. In this thesis, BRaVE is used to represent the real world, and the STS is used for the train running estimation by the algorithms. An estimation process is constructed to manage the train running estimation by using the STS, which is essential for the rescheduling approaches. Unlike BRaVE, the STS only analyses one train at one time. When given the information of junctions, timetables and the sequences of passing trains, it can calculate the arrival time of a particular train at a particular station. The estimation process is described in the following steps:

Step 0: Define the signals in the network: $n \epsilon N = \{n_1, n_2, \dots, n_n\}$.

Define the block sections in the network: $b \epsilon B = \{b_1, b_2, \dots, b_n\}$.

Define the junctions in the network: $j \epsilon J = \{j_1, j_2, \dots, j_n\}$.

Set the sequence of passing trains: $S = [s_1, s_2, \dots, s_n]$.

Set the block clearance time, $T_B = \{ t_b | \, t_b = 0; \; b = b_1, b_2, \dots, b_n\}$, and junction route set time, $T_J = \{t_j | t_j = 0; \; j = j_1, j_2, \dots, j_n\}$.

The simulation results obtained from the STS are related to the sequences of passing trains. When the sequence changes, the simulation result changes as well. In this step, all signals, block sections and junctions in the network are defined, and a sequence of trains passing the junction area is also determined. For each block section, the block clearance time indicates the time when the block section is available for trains to enter, and the junction route set time indicates the time when the junction is available for a junction route to be set. t_b denotes the clearance time for the block section b, and t_j denotes the junction route set time for the junction j. All block clearance times and junction route set times are set to zero at the beginning.

Step 1: Run the first train of the sequence $s = s_1$ in STS;

Since no train is in front of the first train of the sequence, this train is simulated in the STS with all scheduled stops known (based on a nominal timetable).

Step 2: Record the block clearance time, $T_B^s = \{t_b^s \,|b = b_1,\ b_2,\ \dots,\ b_n, s \in S\}$, and junction route set time, $T_J^s = \left\{t_j^s \,\middle|j = j_1,\ j_2,\ \dots,\ j_n, s \in S\right\}$ when train s finishes its journey.

When a train passes a series of block sections, according to the block and interlocking principles, the blocking time for a running train in the simple two-aspect signalling system consists of the following parts, see Figure 4-6 [96].

1. Approach time,
2. Running time in the block,
3. Clearing time (running time over the train length to clear the block), and a
4. Release time to release the route in the block.

Block clearance time, t_b^s is regarded as the clear time of block section b when train s leaves this block section. For example, in Figure 4-6, $t_{b_1}^{s_1}$ means the block clearance time of block section $b_1$when train s_1 passes.

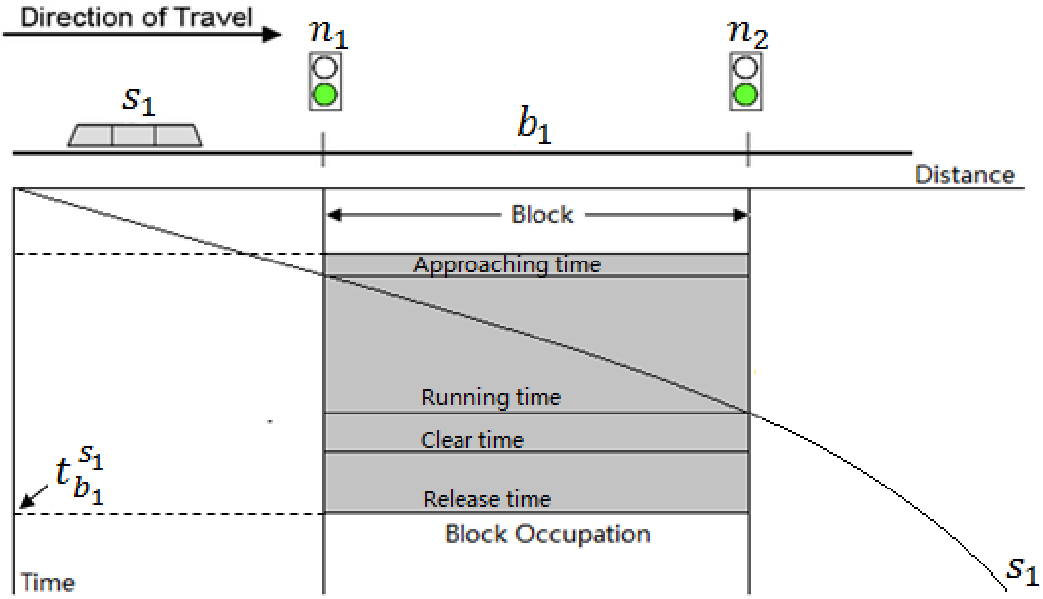


Figure 4-6: Railway block occupation [96]

If a block section has not been occupied by train s along its journey, the block clearance time for this block section keeps the previous value when the last train passed through. Thus, the block clearance time for the train s can be calculated by the equations below:

When $s = s_1$,

$$T_B^s = \{t_b^{s_1} \mid t_b^{s_1} = \begin{cases} t_b^{s_1}, & s_1 \text{ passes the block section } b \\ 0, & \text{otherwise} \end{cases}; b = b_1,\ b_2,\ \dots,\ b_n, s_1 \in S\}$$

Equation 4-1

When $s = s_n \epsilon\, S, n > 1$,

$$T_B^s = \{t_b^{s_n} \mid t_b^{s_n} = \begin{cases} t_b^{s_n}, & s_n \text{ passes the block section } b \\ t_b^{s_{n-1}}, & \text{otherwise} \end{cases}; b = b_1,\ b_2,\ \dots,\ b_n, s_n \in S\}$$

Equation 4-2

The same for the junction route set time, if a train passes a junction area, the route set time for this junction is recorded, if not, the junction route set time for this junction keeps the previous value. Thus, the junction route set time of the train s can be calculated by the equations below:

When $s = s_1$,

$$T_J^s = \{t_j^{s_1} \mid t_j^{s_1} = \begin{cases} t_j^{s_1}, & s_1 \text{ passes the junction } j \\ 0, & \text{otherwise} \end{cases}; j = j_1,\ j_2,\ \dots,\ j_n, s_1 \in S\},$$

Equation 4-3

When $s = s_n \epsilon\, S, n > 1$,

$$T_J^s = \left\{t_j^{s_n} \mid t_j^{s_n} = \begin{cases} t_j^{s_n}, s_n \text{ passes the junction } j \\ t_j^{s_{n-1}}, \text{otherwise} \end{cases}; j = j_1,\ j_2,\ \dots,\ j_n, s_n \in S\},$$

Equation 4-4

Step 4: Calculate signal clearance time: $T_N^s = \{t_n^s \mid n = n_1,\ n_2,\ \dots,\ n_n, s \in S\}$, when train s finishes its journey.

Each block section has at least one approaching signal, for example, in Figure 4-6, n_1 is the approaching signal of block section b_1. On the other hand, one signal can lead at least one block section. If signal n is the approaching signal of block section b, we claim that block section b is relevant to signal n. Similarly, if signal n is the approaching signal of junction j, this junction j is defined to be relevant to signal n. The signal clearance time T_N^s is presented by the equation below:

When $s \epsilon S$,

$$T_N^s = \{t_n^s \mid t_n^s = Max(t_b^s, \quad t_j^s);\ \forall b \text{ is relevant to } n\ ;\ \forall j \text{ is relevant to } n;\ n = n_1, n_2, \ldots, n_n, s \in S\},$$

Equation 4-5

Step 5: Run the next train in sequence S in the STS with the signal clearance times of the previous train applied.

The signal clearance times indicate the earliest release time for a signal, which means signal n is deemed to be closed (red) until the time reaches t_n^s, which can lead to some deceleration and stops which are not scheduled in the timetable to the next train.

Step 6: Repeat Step 2 to Step 5 until all trains have reached their destination.

The arrival times of all trains in this sequence at their destinations are obtained, and these values can be used to evaluate the sequence with a well-defined cost function.

4.4 Rescheduling process of the simulation

In reality, railway network rescheduling is a complex, multi-stage process, as described by Laube and Schaffer [97]. The details of each conceptual level are illustrated in Figure 4-7 [98]. The process begins from collecting network condition information (e.g. infrastructure status, train positions and states), and compares this information to pre-defined thresholds (i.e. deviation, disturbance, service change) to determine if a rescheduling process is needed. Once a rescheduling process is triggered (i.e. the rescheduling plans are usually periodically generated, e.g. every 3 minutes), the operator must prepare a new scheduled plan. After an appropriate prediction, one or more rescheduling algorithms must be run to actually develop the new production plan. The choice of method depends on the type of problem, data availability and urgency of the need for a new production plan. These refresh schedules are then transmitted to dispatchers to execute.

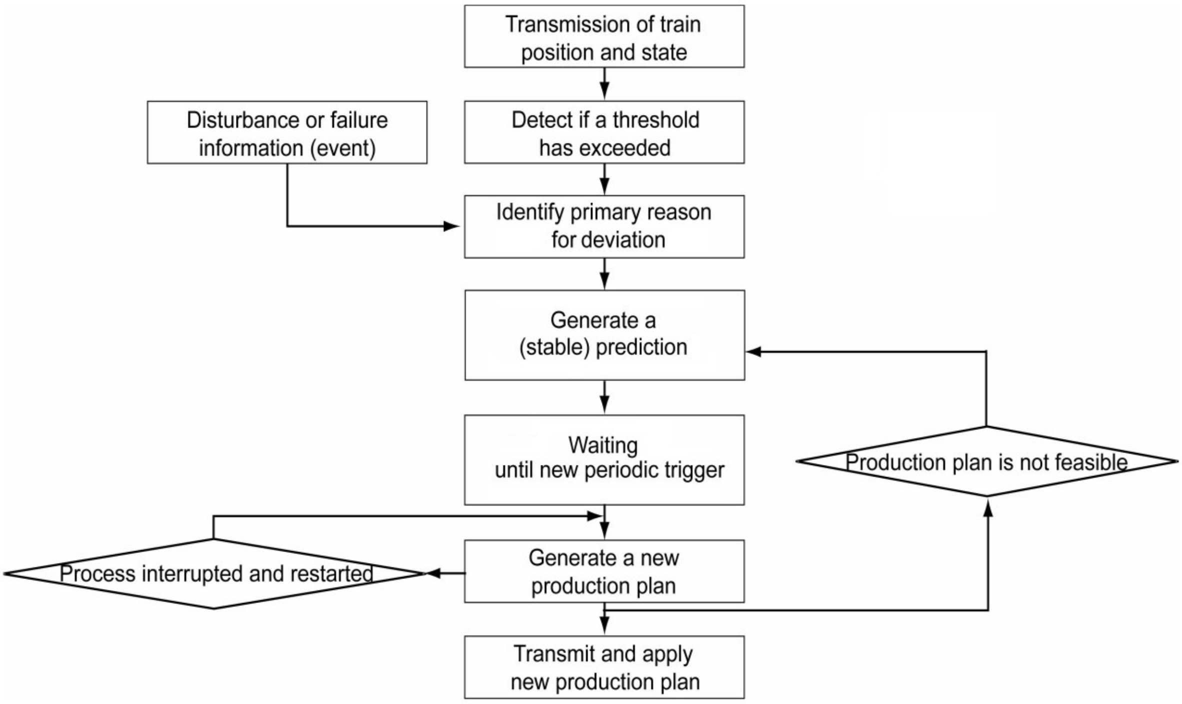

Transmission of train position and state
Disturbance or failure information (event)
Detect if a threshold has exceeded
Identify primary reason for deviation
Generate a (stable) prediction
Waiting until new periodic trigger
Production plan is not feasible
Process interrupted and restarted
Generate a new production plan
Transmit and apply new production plan

Figure 4-7: Railway network rescheduling process in reality [98]

In simulation models, the network condition information is known by the simulator, and all disturbances or failures are predefined. In this study, only reordering at the junctions is considered for the network rescheduling. The rescheduling process of the simulation model is shown in Figure 4-8. Firstly, run the network in BRaVE simulator, and observe the network conditions. Once the current conditions have met some pre-defined thresholds, a rescheduling command is sent to the BRaVE. Once this rescheduling command is received, BRaVE simulator is paused and all network information is transmitted to the STS. Combining the train running estimations provided by the STS and an appropriate algorithm, an optimised reordering plan is generated (according to a pre-defined cost function), and this plan is transmitted back to the simulator and the routing sequence as junctions is updated. The simulation resumes and runs the timetable with new junction routing sequence.

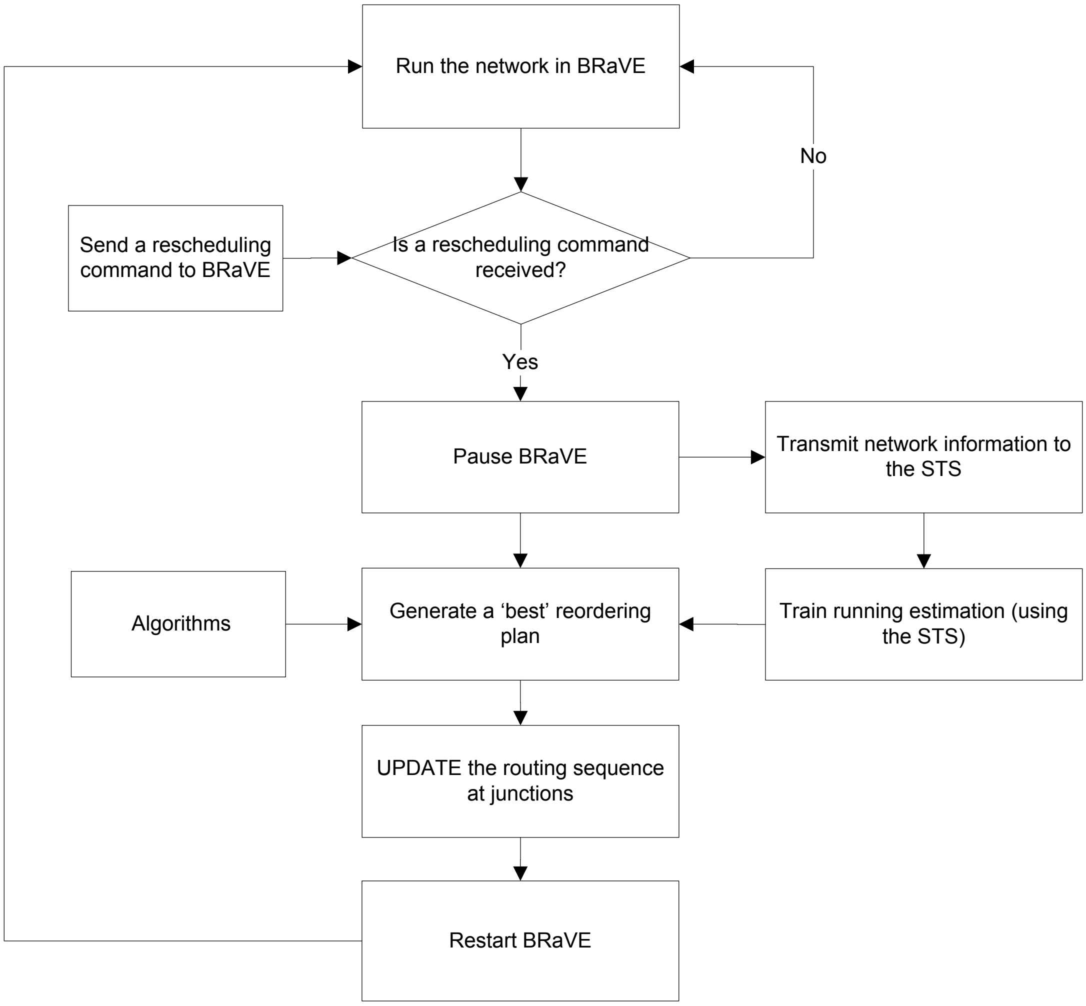

Run the network in BRaVE
No
Send a rescheduling command to BRaVE
Is a rescheduling command received?
Yes
Pause BRaVE
Transmit network information to the STS
Algorithms
Generate a 'best' reordering plan
Train running estimation (using the STS)
UPDATE the routing sequence at junctions
Restart BRaVE

Figure 4-8: Railway network rescheduling process in simulation

4.5 Conclusion

The simulation platform used in this thesis is the BRaVE simulator, which is developed by the University of Birmingham. The capability of BRaVE is briefly described, which are compatible with the visualisation tool described in Chapter 3. To speed up the rescheduling algorithms, the

STS developed by the University of Birmingham is used for the estimation of train movements for some advanced algorithms.

The algorithm implementation progress is explained with a plug-in architecture and the train running estimation process is described step by step. The values of predicted arrival times at the destinations can be used to evaluate the sequence with an appropriate cost function.

The rescheduling processes in reality and in the simulation model are both described. The prediction part of the rescheduling process in the simulation model is not needed, because when the simulator pauses, no time is added to the plan generation. However, considering the practical situation, the calculation time for the algorithm still needs to be considered when we evaluate the performance of the algorithm in simulation.

CHAPTER 5

EXPERIMENTS ON SINGLE JUNCTION RESCHEDULING USING DIFFERENT ALGORITHMS

In this chapter, a practical rescheduling problem in the single junction area is defined and a number of representative rescheduling approaches are applied to solve the problem. These approaches are investigated and tested on a series of delay scenarios in microscopic simulation, and rescheduling solutions are compared and analysed. The case study investigates how different levels of delays and numbers of constraints may affect the performance of algorithms for network-wide rescheduling in terms of quality of solution and computation time. A recommendation for using these approaches is given based on their performance on different delay scenarios, and can be used as references for rescheduling.

5.1 Introduction to the approaches in the experiments

As a verification and supplement to the benchmark problem presented in the thesis 'Railway Traffic Rescheduling Approaches To Minimise Delays In Disturbed Conditions' [72], the experimental algorithms: Brute Force (BF), Dynamic Programming (DP), Decision Tree Based Elimination (DTBE), Tabu Search (TS), Local Search (LS), Simulated Annealing (SA), Genetic Algorithms (GA) and Ant Colony Optimisation Algorithm (ACO) were chosen. Additionally, Timetable-Order-Enforced (TOE), First-Come-First-Served (FCFS), and First-Leave-First-Served (FLFS) have been added as they are commonly used strategies in practice.

A brief description of the different optimisation algorithms for train rescheduling which are used in this work is presented and classified here. To address the characteristics of each classification, a high level description is provided in Table 5-1. The introduction and application of each algorithm is discussed later in this chapter.

Algorithms	**Classification**	**Description**
Timetable-Order-Enforce (TOE)	Rule-based methods	These methods are parametric methods, based on a set of predefined rules [99]. No complex calculation is needed.
First-Come-First-Served (FCFS)		
First-Leave-First-Served (FLFS)		
Brute Force (BF)	Exhaustive methods	An entire search algorithm that systematically examines all possible solutions. Usually very simple to program and requiring no specific knowledge [100].
Dynamic Programming (DP)	Graphic methods	A scientific search method of entire search methods by analysing the relations between numbers, by means of figures (e.g. the SST) [101].
Decision Tree Based Elimination (DTBE)	Branch and cut methods	A combination of a cutting plane with a branch-and-bound algorithm. The cutting plane methods improve the convergence of the problem, and branch-and-bound algorithms proceed with an accurate divide and conquer activities[102].
Ant Colony Optimisation Algorithm (ACO)	Evolutionary methods	Evolutionary Algorithms (EA) are an optimisation method that use the computational models of natural evolution, such as mutation, crossover, natural selection and survival of the fittest [103].
Genetic Algorithms (GA)		
Simulated Annealing (SA)		
Tabu Search (TS)		
Local Search (LS)		

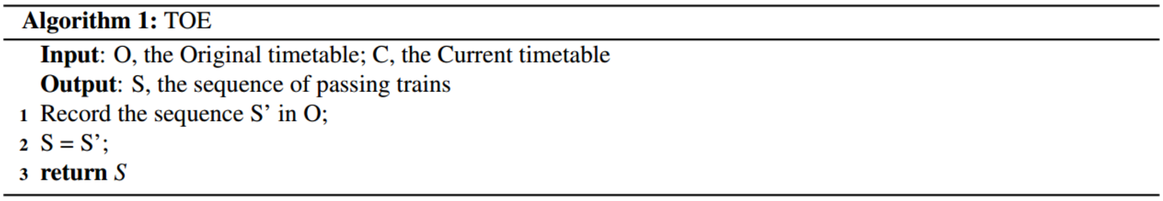
Table 5-1: Algorithm classification

5.2 Application of algorithms

5.2.1 Rule-based methods

Rule-based algorithms are usually adopted to solve very complex problems in a limited time. In this chapter, three rule-based algorithms, which are intuitively used on the operational railway by signallers, are used to solve the train rescheduling problem. They are: TOE, FCFS, and FLFS.

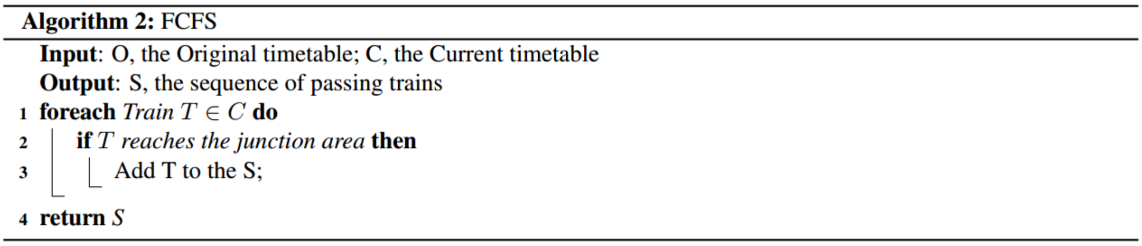
Timetable-Order-Enforce (TOE) is used as a standard method for comparison of all the other algorithms. No effective reordering strategy is applied, and the original order will be enforced in all scenarios. The method is predicted to lead to an unsatisfactory result. The application of TOE is done by recording the original passing order with no delay, and enforcing this order for any delayed situations. The pseudo-code of TOE is given in Figure 5-1.



```
Algorithm 1: TOE
  Input: O, the Original timetable; C, the Current timetable
  Output: S, the sequence of passing trains
  Record the sequence S' in O;
  S = S';
  return S
```

Figure 5-1: Pseudo-code of TOE

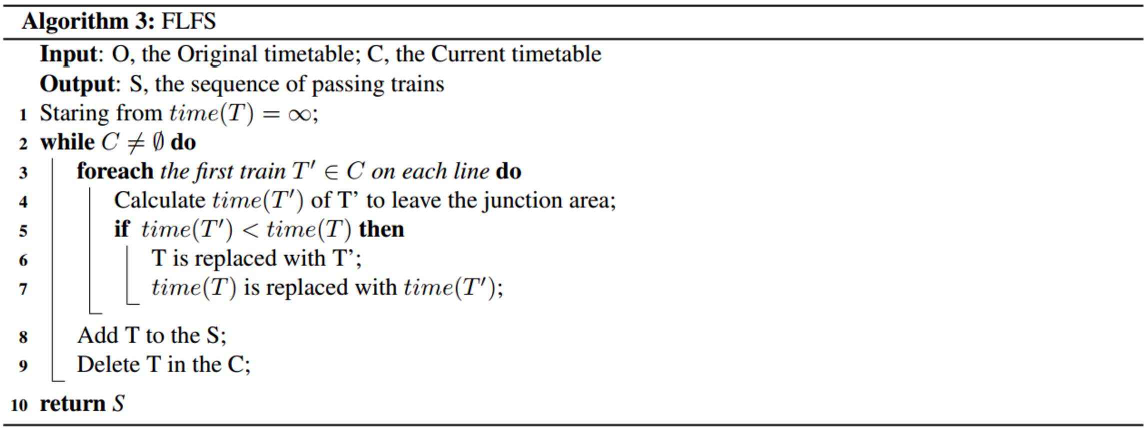
First-Come-First-Served (FCFS) is where a simple 'rule-of-thumb' rule is commonly adopted in railway practice. It consists of giving precedence to the train arriving first at a junction area. FCFS computes the arrival time of the trains at each junction, sequencing the trains in the order of arrival [104]. The pseudo-code of FCFS is given in Figure 5-2.



```
Algorithm 2: FCFS
  Input: O, the Original timetable; C, the Current timetable
  Output: S, the sequence of passing trains
  foreach Train $T \in C$ do
      if $T$ reaches the junction area then
          Add T to the S;
  return $S$
```

Figure 5-2: Pseudo-code of FCFS

First-Leave-First-Served (FLFS) is used to compare the first arriving trains from different lines which may have a conflict at the junction, and gives precedence to the train that is able to leave the block section first [104]. FLFS is a compromise between two commonly used dispatching rules of: (i) Fast train served first; and (ii) FCFS. The pseudo-code of FLFS is given in Figure 5-3.



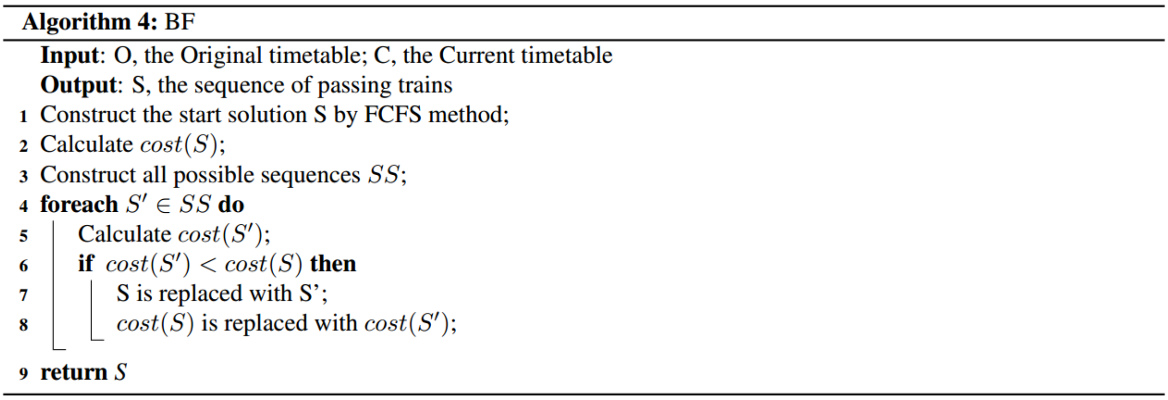
```
Algorithm 3: FLFS
  Input: O, the Original timetable; C, the Current timetable
  Output: S, the sequence of passing trains
  Staring from $time(T) = \infty$;
  while $C \neq \emptyset$ do
      foreach the first train $T' \in C$ on each line do
          Calculate $time(T')$ of T' to leave the junction area;
          if $time(T') < time(T)$ then
              T is replaced with T';
              $time(T)$ is replaced with $time(T')$;
      Add T to the S;
      Delete T in the C;
  return $S$
```

Figure 5-3: Pseudo-code of FLFS

The implementation of the rule-based method can be simply described as making a decision as to whether a train can pass or not when it requests passing authority at the junction area, based on whether the rule is satisfied.

5.2.2 Exhaustive method

Brute Force (BF) is a straightforward approach to solving a problem. It suggests that all possible solutions in the problem's domain are generated, selecting those of them which satisfy the specific constraints and finding the optimal one (the one which fits with an objective function) [105]. When BF is used to solve a train rescheduling problem with many possible sequences, it will take a long time to evaluate every possible solution with the objective function. An exhaustive search is impractical, although as BF enumerates all possible sequences, it always finds the best solution to the train rescheduling problem. The pseudo-code of BF is given in Figure 5-4.

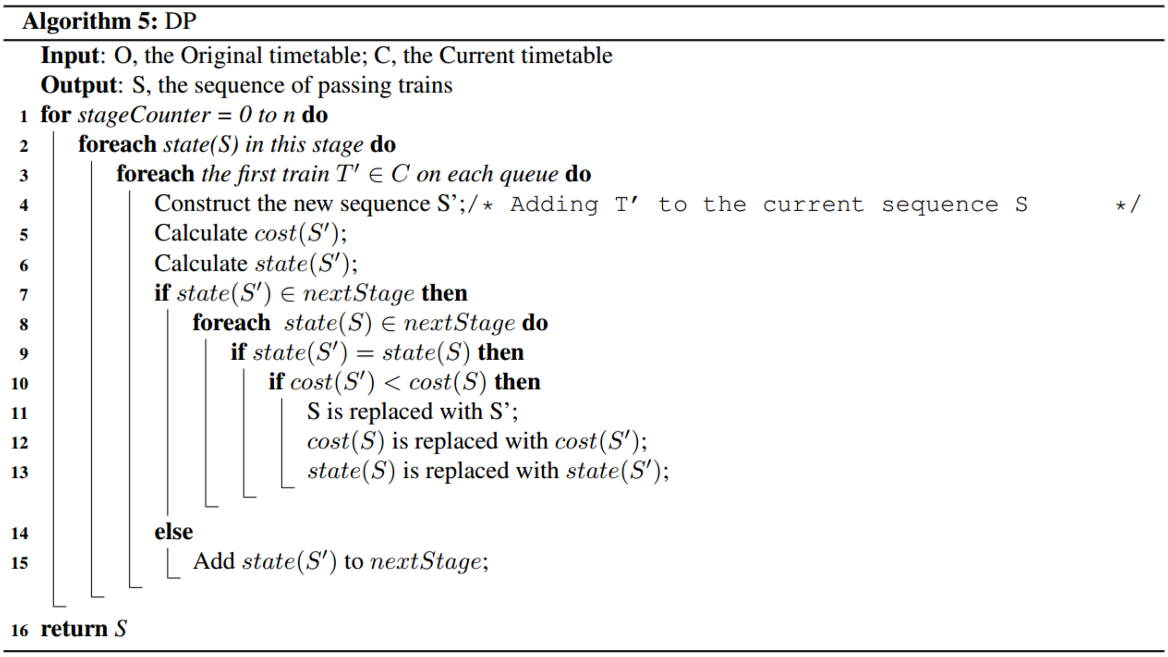


```
Algorithm 4: BF
  Input: O, the Original timetable; C, the Current timetable
  Output: S, the sequence of passing trains
  Construct the start solution S by FCFS method;
  Calculate $cost(S)$;
  Construct all possible sequences $SS$;
  foreach $S' \in SS$ do
      Calculate $cost(S')$;
      if $cost(S') < cost(S)$ then
          S is replaced with S';
          $cost(S)$ is replaced with $cost(S')$;
  return $S$
```

Figure 5-4: Pseudo-code of BF

5.2.3 Graphic method

Dynamic Programming (DP) is one of the most commonly used algorithms for solving the shortest path problem and it uses Stage-to-Stage Transformations (SST). It resolves a problem into a collection of sub-problems and tackles them one by one. The smallest one is taken first, using the solutions to help figure out a bigger one, until all sub-problems are solved. When solving a problem by dynamic programming, the most crucial question is what the sub-problems are. In the problem considered here, the train rescheduling problem is modelled by a forward stage-to-stage model [106]. Dynamic Programming can also be considered as a kind of exhaustive algorithm; it will give an optimal solution if the original problem is completely decomposed. The pseudo-code of DP is given in Figure 5-5.



```
Algorithm 5: DP
Input: O, the Original timetable; C, the Current timetable
Output: S, the sequence of passing trains
for stageCounter = 0 to n do
    foreach state(S) in this stage do
        foreach the first train $T' \in C$ on each queue do
            Construct the new sequence S';/* Adding T' to the current sequence S */
            Calculate $cost(S')$;
            Calculate $state(S')$;
            if $state(S') \in nextStage$ then
                foreach $state(S) \in nextStage$ do
                    if $state(S') = state(S)$ then
                        if $cost(S') < cost(S)$ then
                            S is replaced with S';
                            $cost(S)$ is replaced with $cost(S')$;
                            $state(S)$ is replaced with $state(S')$;
            else
                Add $state(S')$ to $nextStage$;
return $S$
```

Figure 5-5: Pseudo-code of DP

5.2.4 Branch and cut method

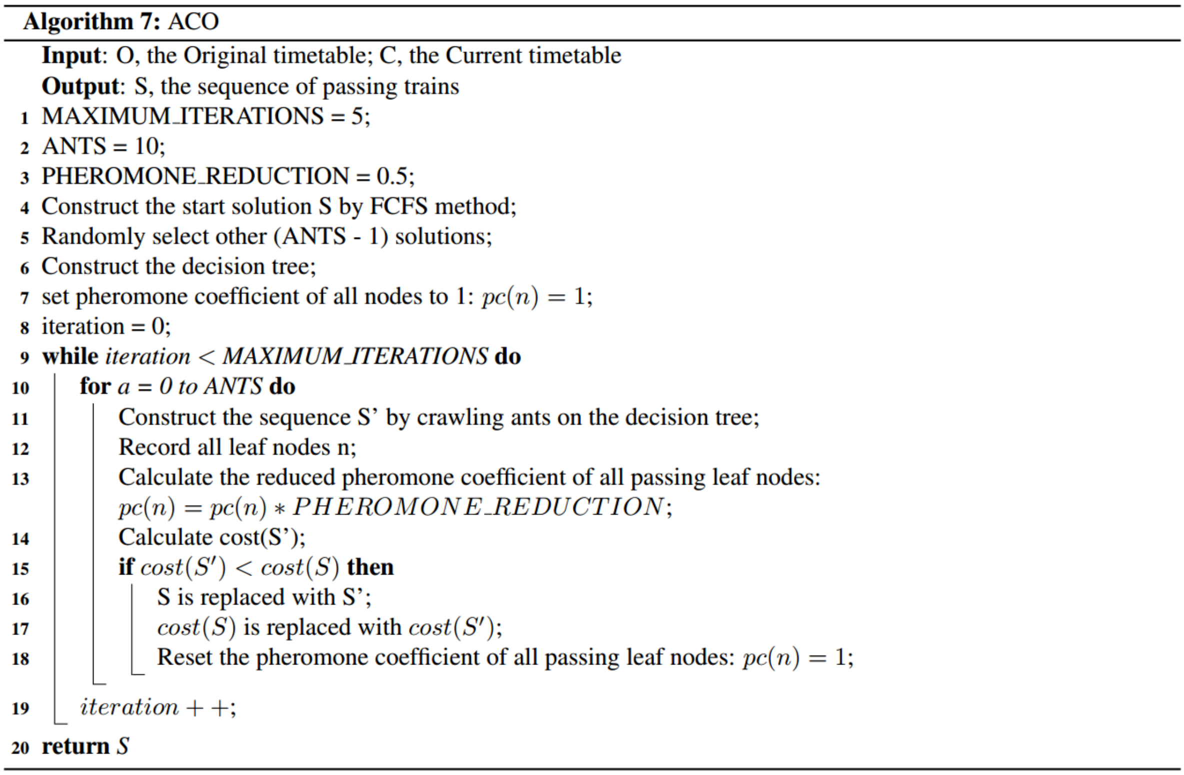
Decision Tree Based Elimination (DTBE) uses a greedy, top-down recursive partition approach to induce a decision tree from data. The approach is designed to reduce the search time by pruning the decision tree according to some heuristic rules [72]. The pseudo-code of DTBE is given in Figure 5-6.

```
Algorithm 6: DTBE
Input: O, the Original timetable; C, the Current timetable
Output: S, the sequence of passing trains
FIRST_PRUNE_LEVEL = 3;
PRUNE_RATIO = 0.3;
Construct the decision tree;
for treeLevel = 0 to n do
    foreach branches S in this treeLevel do
        Calculate $cost(S)$;
    if treeLevel >= FIRST_PRUNE_LEVEL then
        Range all branches according to the ascending order of costs;
        Calculate the number of branches;
        Keep first (number of branches)*(1−PRUNE_RATIO) branches left;
    if treeLevel < n then
        foreach branches S in this treeLevel do
            foreach the first train $T' \in C$ on each queue do
                Grow the branch S';/* Adding T' to the current branch S */
                Add S' to next treeLevel;
    if treeLevel = n then
        Find the best solution S according to the minimum $cost(S)$ ;
return S
```

Figure 5-6: Pseudo-code of DTBE

5.2.5 Evolutionary algorithms and their stability

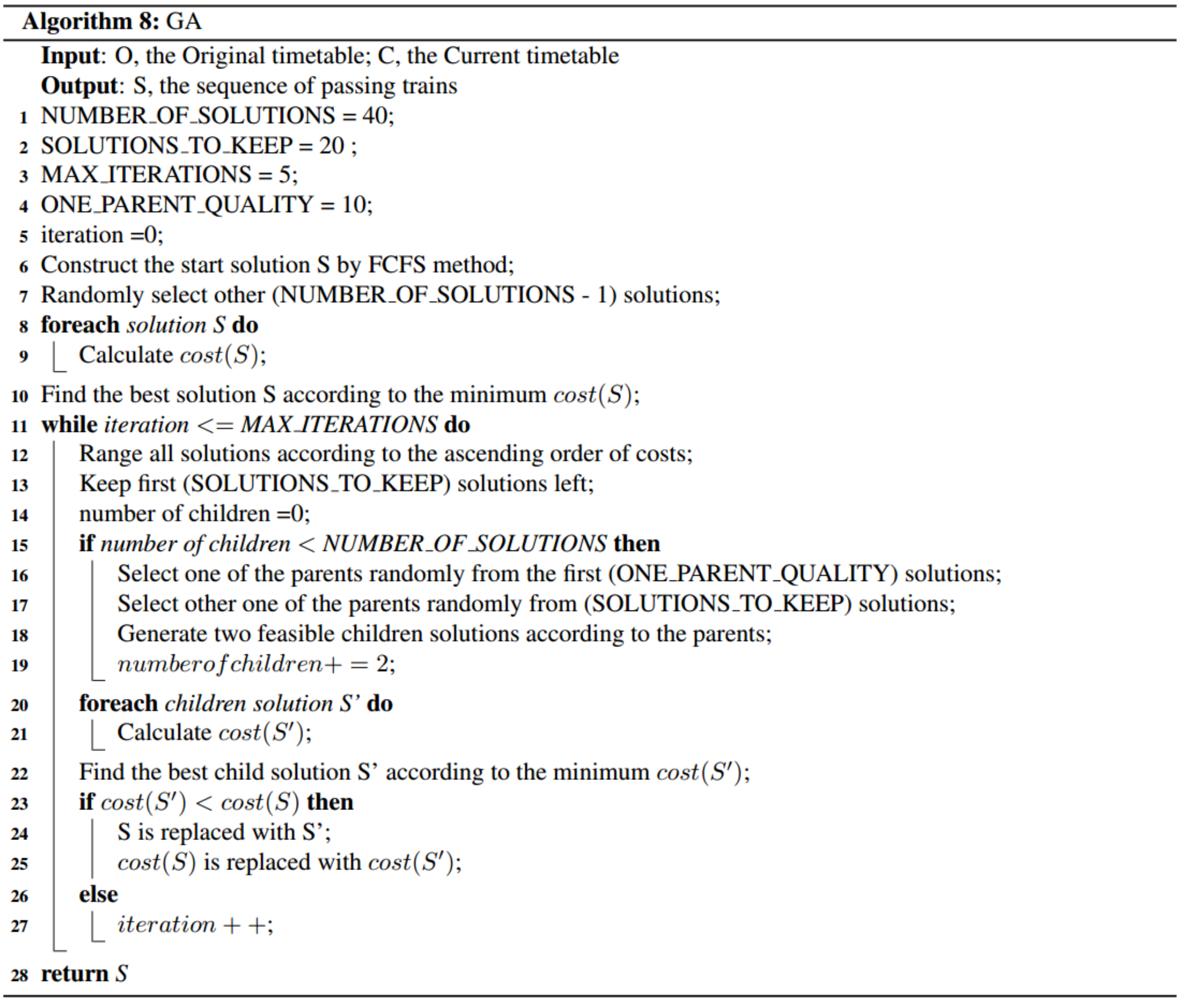
Ant colony optimisation (ACO) emulates ants searching for food. In ACO algorithms (first proposed by Dorigo in his PhD thesis [107]), in virtual, 'ants' construct solutions by probabilistically making a set of local decisions to create paths on a graph. At each node of a path, the ant can extend the current partial solution by keeping crawling one possible branch [108]. Each node is assigned a 'pheromone' coefficient value according to the cost of a solution which need pass through the node. The pseudo-code of ACO is given in Figure 5-7.



```
Algorithm 7: ACO
Input: O, the Original timetable; C, the Current timetable
Output: S, the sequence of passing trains
MAXIMUM_ITERATIONS = 5;
ANTS = 10;
PHEROMONE_REDUCTION = 0.5;
Construct the start solution S by FCFS method;
Randomly select other (ANTS - 1) solutions;
Construct the decision tree;
set pheromone coefficient of all nodes to 1: $pc(n) = 1$;
iteration = 0;
while iteration < MAXIMUM_ITERATIONS do
    for a = 0 to ANTS do
        Construct the sequence S' by crawling ants on the decision tree;
        Record all leaf nodes n;
        Calculate the reduced pheromone coefficient of all passing leaf nodes:
        $pc(n) = pc(n) * PHEROMONE\_REDUCTION$;
        Calculate cost(S');
        if $cost(S') < cost(S)$ then
            S is replaced with S';
            $cost(S)$ is replaced with $cost(S')$;
            Reset the pheromone coefficient of all passing leaf nodes: $pc(n) = 1$;
    $iteration++$;
return $S$
```

Figure 5-7: Pseudo-code of ACO

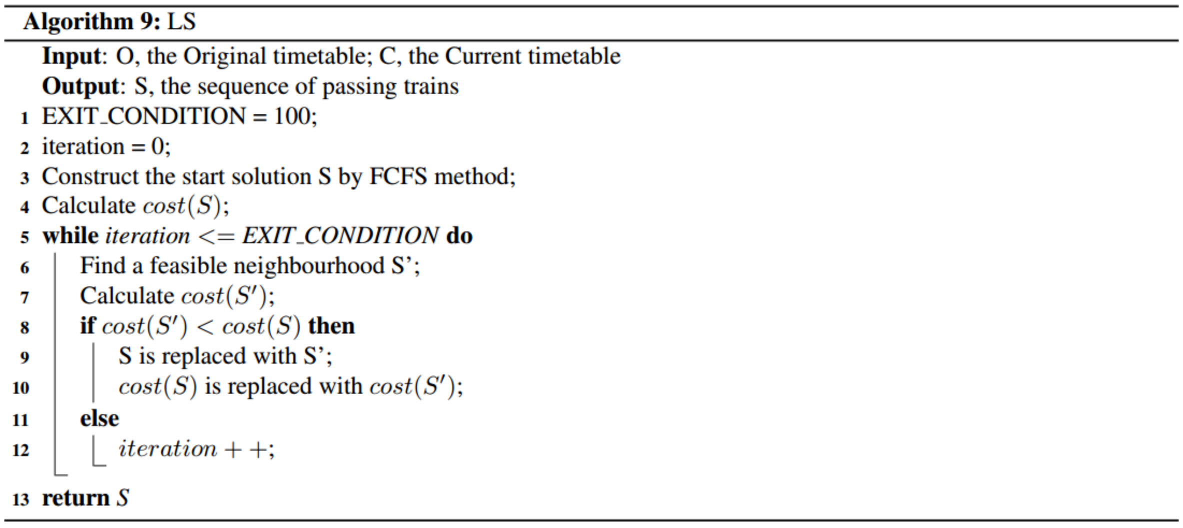
Genetic Algorithms (GA) were rapidly developed and widely used over the last 20 years. They are simulated on the principles of evolution via natural selection, which means a population of candidates is produced through competition, mating and variation activities [109]. A pre-defined cost function is used to evaluate these individuals, and next generation success varies with the 'cost'. The pseudo-code of GA is given in Figure 5-8.



```
Algorithm 8: GA
Input: O, the Original timetable; C, the Current timetable
Output: S, the sequence of passing trains
NUMBER_OF_SOLUTIONS = 40;
SOLUTIONS_TO_KEEP = 20 ;
MAX_ITERATIONS = 5;
ONE_PARENT_QUALITY = 10;
iteration =0;
Construct the start solution S by FCFS method;
Randomly select other (NUMBER_OF_SOLUTIONS - 1) solutions;
foreach solution S do
    Calculate $cost(S)$;
Find the best solution S according to the minimum $cost(S)$;
while iteration <= MAX_ITERATIONS do
    Range all solutions according to the ascending order of costs;
    Keep first (SOLUTIONS_TO_KEEP) solutions left;
    number of children =0;
    if number of children < NUMBER_OF_SOLUTIONS then
        Select one of the parents randomly from the first (ONE_PARENT_QUALITY) solutions;
        Select other one of the parents randomly from (SOLUTIONS_TO_KEEP) solutions;
        Generate two feasible children solutions according to the parents;
        $numberofchildren+ = 2$;
    foreach children solution S' do
        Calculate $cost(S')$;
    Find the best child solution S' according to the minimum $cost(S')$;
    if $cost(S') < cost(S)$ then
        S is replaced with S';
        $cost(S)$ is replaced with $cost(S')$;
    else
        $iteration + +$;
return S
```

Figure 5-8: Pseudo-code of GA

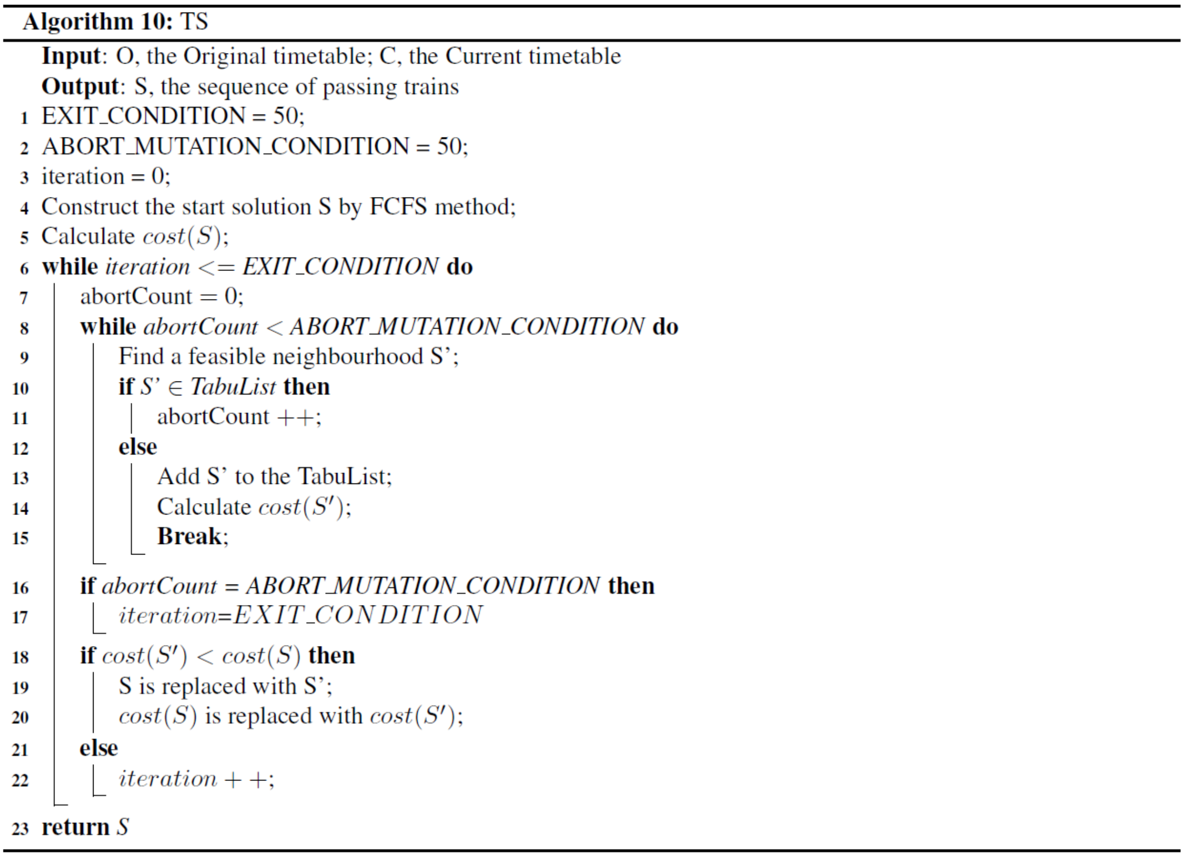
Local Search (LS) is one of elementary evolutionary algorithms. The principle of local search is to keep a simple 'current best' state, and try to improve it by modifying repeatedly within a local area, until some termination conditions are reached, such as the maximum number of iterations or until no further improvement can be found over a specific number of iterations [110]. The pseudo-code of LS is given in Figure 5-9.



```
Algorithm 9: LS
  Input: O, the Original timetable; C, the Current timetable
  Output: S, the sequence of passing trains
  EXIT_CONDITION = 100;
  iteration = 0;
  Construct the start solution S by FCFS method;
  Calculate $cost(S)$;
  while $iteration <= EXIT\_CONDITION$ do
      Find a feasible neighbourhood S';
      Calculate $cost(S')$;
      if $cost(S') < cost(S)$ then
          S is replaced with S';
          $cost(S)$ is replaced with $cost(S')$;
      else
          $iteration++$;
  return $S$
```

Figure 5-9: Pseudo-code of LS

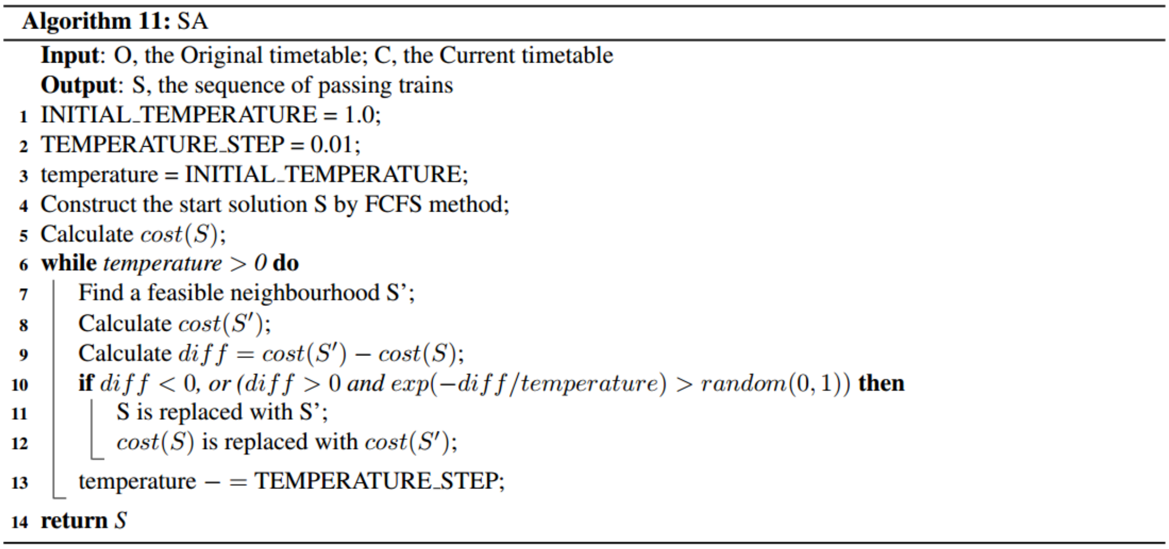
Tabu Search (TS), presented by Glover [111, 112], is considered to be a more efficient method than the local search as it uses memory structures: a ‘tabu’ list is built by recording a potential solution or swap, and it is forbidden to repeat this is in the next certain number of iterations. The pseudo-code of TS is given in Figure 5-10.



```
Algorithm 10: TS
Input: O, the Original timetable; C, the Current timetable
Output: S, the sequence of passing trains
EXIT_CONDITION = 50;
ABORT_MUTATION_CONDITION = 50;
iteration = 0;
Construct the start solution S by FCFS method;
Calculate $cost(S)$;
while $iteration <= EXIT\_CONDITION$ do
    abortCount = 0;
    while $abortCount < ABORT\_MUTATION\_CONDITION$ do
        Find a feasible neighbourhood S';
        if $S' \in TabuList$ then
            abortCount ++;
        else
            Add S' to the TabuList;
            Calculate $cost(S')$;
            Break;
    if $abortCount = ABORT\_MUTATION\_CONDITION$ then
        $iteration = EXIT\_CONDITION$
    if $cost(S') < cost(S)$ then
        S is replaced with S';
        $cost(S)$ is replaced with $cost(S')$;
    else
        $iteration++$;
return $S$
```

Figure 5-10: Pseudo-code of TS

Simulated Annealing (SA) is a further development of a local search to solve the combinatory optimisation problem as well as Tabu search [113]. The algorithm is based on the annealing process used in statistical mechanics, where heating and controlled cooling is used. Begin with a 'current best' state, a new candidate solution is found based on a local search, and this candidate solution will be selected as it has been improved or satisfies a probabilistic function. The pseudo-code of SA is given in Figure 5-11.



```
Algorithm 11: SA
Input: O, the Original timetable; C, the Current timetable
Output: S, the sequence of passing trains
INITIAL_TEMPERATURE = 1.0;
TEMPERATURE_STEP = 0.01;
temperature = INITIAL_TEMPERATURE;
Construct the start solution S by FCFS method;
Calculate $cost(S)$;
while $temperature > 0$ do
    Find a feasible neighbourhood S';
    Calculate $cost(S')$;
    Calculate $diff = cost(S') - cost(S)$;
    if $diff < 0$, or $(diff > 0$ and $exp(-diff/temperature) > random(0,1))$ then
        S is replaced with S';
        $cost(S)$ is replaced with $cost(S')$;
    temperature $-=$ TEMPERATURE_STEP;
return $S$
```

Figure 5-11: Pseudo-code of SA

The Evolutionary Algorithm (EA) is an optimisation method which uses the computational models of natural evolution, such as mutation, crossover, natural selection and survival of the fittest [103]. Thus, the three main components for an evolutionary process are:

- Selection: Some environmental factors (i.e. the cost function used in the train rescheduling problem) must favour certain speciality over others.
- Variation: Individuals arise some characteristics which are significantly different from their ancestors.
- Heritability: Generally, children must inherit almost traits from their parents to insure that selected traits survive in the next generation.

Based on the characteristics of the evolutionary algorithms, there is no guarantee that the problem will be optimised to one optimal solution every time. When an evolutionary algorithm is applied to a railway system, the stability of using the algorithm must be ensured by determining certain algorithm parameters.

The range ω and the coefficient of variation C_v in statistics are used to determine the stability of using evolutionary algorithms. When they are within a reasonable range of values (ω is set to range from 0 to 1000 and C_v is set to range from 0 to 0.001 in this work), the stability of the evolutionary algorithm is acceptable.

Arithmetically, the range ω of a set of data is the difference between the largest and smallest values [114]:

$$\omega = X_{max} - X_{min}$$

Equation 5-1

the coefficient of variation C_v, known as unitised risk, is a normalised measure of dispersion of a probability distribution or frequency distribution. It is defined as the ratio of the standard deviation σ to the mean μ. It shows the extent of variability in relation to the mean of the population [115]:

$$C_v = \frac{\sigma}{\mu}$$

Equation 5-2

The evolutionary algorithms are tested with different parameters in place until stability is reached, then the parameter values will be kept when the evolutionary algorithms are addressed into different scenarios.

5.3 Development of a benchmark scenario for algorithm assessment

5.3.1 Cost function for the benchmark problem

To find an optimal solution, a rescheduling ‘objective’ for a particular network should be defined in the railway traffic management system, for example, minimising overall delay, maximising the track usage, or minimising journey times. There are a variety of objectives of rescheduling that have been considered in previous research, for instance, to minimise train delay (sum, max, average, etc.) [8, 11, 29], to minimise passenger delay [10], to minimise energy consumption [9], etc. In Fan’s thesis [72], the rescheduling objective is measured by cost, whereby Network Rail pays a fine to train companies for network control delays, which is calculated from delay minutes. To make the algorithm more sensitive to different solutions, the total delay cost $Cost(S)$ for the sequence S in this thesis is defined as the delay penalty calculated in seconds for all considered trains; the cost function is thus represented as:

$$Cost(S) = \sum_{i=1}^{n} DT_i(S) \times DP_i$$

Equation 5-3

n: The total number of trains;

S: The sequence of trains through the junction

DT_i: Delay time for the ith train in seconds at its last stop in the control zone;

DP_i: Delay Penalty per second of the ith train;

5.3.2 Introduction to the infrastructure and rolling stocks

A network layout (Figure 5-12 (a)) considered in the research is the same as the one in section 3.4.2. A simplified track map with the experimental junction area marked is presented in Figure 5-12 (b) with its boundary from Welwyn Garden City and Hertford in the north to the terminal stations at London King’s Cross and Moorgate in the south. The layout includes two fast lines (red lines), two slow lines (black lines) and one parallel line (blue line), which is from Hertford to Moorgate.

The junction area studied in this chapter is presented in detail in Figure 5-13. In this work 14 trains coming from Welwyn Garden City and Hertford are considered over identical two time periods. In the first time period the trains are numbered 01 to 07 and then all services are repeated in the second time period (numbered 11 to 17). Inspired by the real timetable published by Network Rail, an idealised model with high frequency services (services repeat every 5 minutes) is simulated in BRaVE. These services include two types of rolling stock: fast train (vehicle type Class 313) and slow train (vehicle type Class 59).

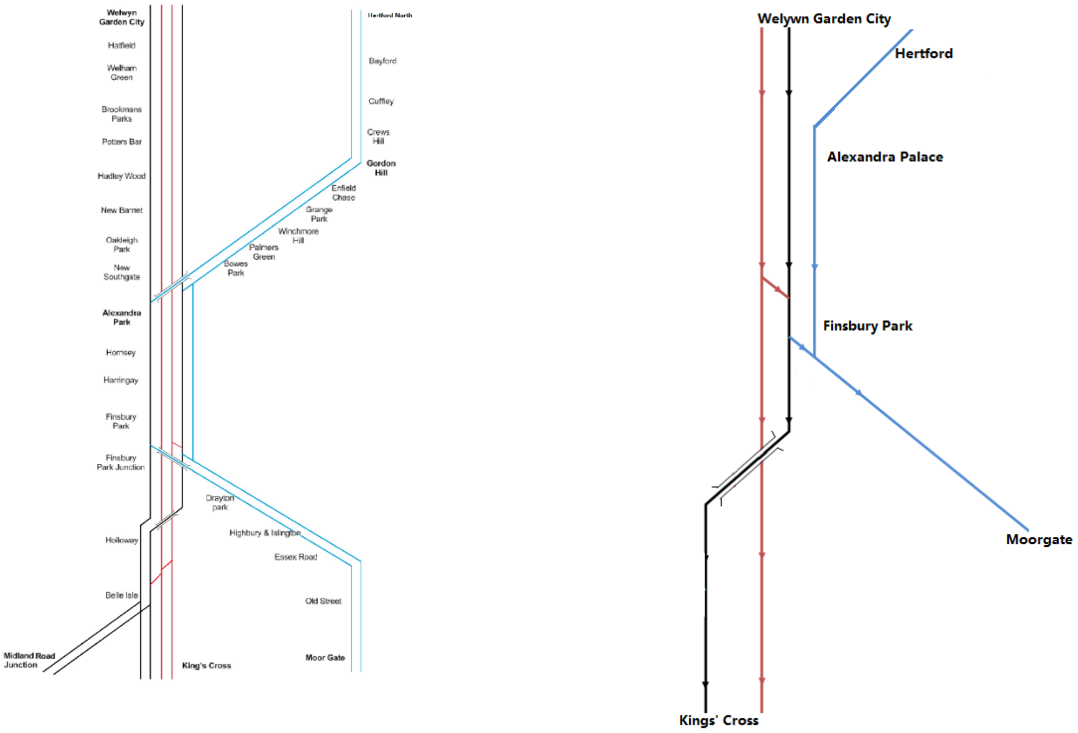


Figure 5-12(a): A track map of the network; (b): A simplified track map of the experimental network in this chapter

Junction region

The experiment junction area consisting of two simple adjacent flat junctions is located near Finsbury Park Station, and is shown in Figure 5-13. The layout includes one fast line (red line F), one slow line (black line S) from Welwyn Garden City (W) to King's Cross (K), and one parallel line (blue line P), which comes from Hertford (H) then goes towards Moorgate (M). The distance between Welwyn Garden City to the north edge of the junction area is about 28.6km and the distance between Hertford to the north edge of the junction area is about 27.5km; the distance between the junction edge to the junction signals is 870m and the distance between two flat junctions is 556m; the distance between the south edge of the junction area to King's Cross is about 4km and the distance between the south edge of the junction area to Moorgate is about 5.8km [116]. These two flat junctions can be regarded as one junction control region, and all algorithms are investigated in this junction region.

All services come from the same direction, and are numbered 01 to 07 in the first time period and then repeated (numbered 11 to 17) in the second time period. The letters (shown in brackets) are of the form [(Line, origin) - (Line, destination)], for example, "01[(F, W) - (F, K)]" means train 01 coming from Welwyn Garden City on the Fast line, terminating at King's Cross on the Fast line. In Figure 5-13, services in the first period from 01 to 07 are shown to approach the junction area.

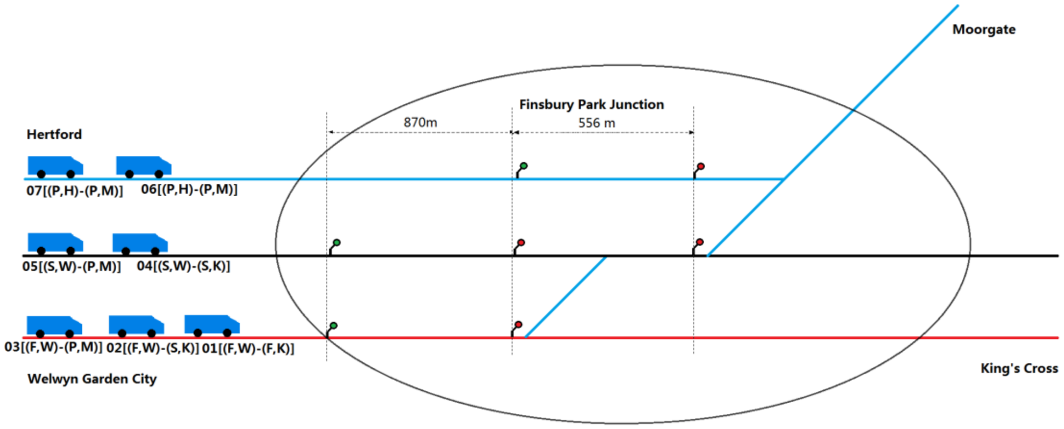


Figure 5-13: Junction area layout and approaching services of first period

Original Timetable

Regarded as the main rescheduling area, all services in this network are arranged to pass this area according to a conflict-free timetable. The timetable in Table 5-2 shows all services in two time periods with scheduled start times, arrival times, and types of vehicles; to give significant observation outputs, the service frequency is compressed to be every 5 minutes.

Period 1							
Train No.	**01**	**02**	**03**	**04**	**05**	**06**	**07**
Vehicle Type	Class 313	Class 313	Class 313	Class 59 loco	Class 59 loco	Class 313	Class 313
Start	(F, W)	(F, W)	(F, W)	(S, W)	(S, W)	(P, H)	(P, H)
Destination	(F, K)	(S, K)	(P, M)	(S, K)	(P, M)	(P, M)	(P, M)
Welwyn Garden City	07:00:00	07:02:00	07:04:00	07:00:00	07:02:00		
Hertfort						07:00:00	07:02:00
King’s Cross	07:19:00	07:21:00		07:30:00			
Moorgate			07:29:00		07:38:00	07:25:00	07:27:00
Period 2							
Train number	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Vehicle Type	Class 313	Class 313	Class 313	Class 59 loco	Class 59 loco	Class 313	Class 313
Start	(F, W)	(F, W)	(F, W)	(S, W)	(S, W)	(P, H)	(P, H)
Destination	(F, K)	(S, K)	(P, M)	(S, K)	(P, M)	(P, M)	(P, M)
Welwyn Garden City	07:05:00	07:07:00	07:09:00	07:05:00	07:07:00		
Hertfort						07:05:00	07:07:00
King’s Cross	07:24:00	07:26:00		07:35:00			
Moorgate			07:36:00		07:40:00	07:31:00	07:33:00

Table 5-2: Original conflict-free timetable

Rolling stock

The rolling stock configuration and train-specific delay penalties of different vehicle types are presented in Table 5-3; this data has been obtained from industry contacts. All this information will be used to simulate train running and calculate the cost function in the following sections.

Type	**Class 313**	**Class 59 loco**
Maximum speed (km/h)	121	72
Maximum braking rate (ms^{-2})	0.78	0.45
Max acceleration (ms^{-2})	0.588	1
Total mass (tonnes)	220	132
Train length (m)	118	21.4
Number of seats	231	188
Penalty (£s^{-1})	0.1	0.02

Table 5-3: Vehicles parameters

5.3.1 Benchmark scenario design

With the aim of building an understanding of how to select a particular algorithm to address different delay situations, the author has designed a series of benchmark scenarios to test and evaluate the performance of different rescheduling approaches. In this section, a series of disturbed scenarios are designed and tested with different rescheduling algorithms in place.

In a set of experiments, there are several variables which need be defined before starting. Generally, the numbers of initial delays, delay type, number of trains in total and delay size are used to design different scenarios. The diagram in Figure 5-14 gives the structure of the classification of different scenarios. Train 01 is delayed to test the single train delay scenarios and trains 01 and 04 are delayed to test the multiple train delay scenario. Delay type includes initial delay and additive delay; for initial delay tests, a train is stopped at Alexandra Palace Station for a defined time period due to signal failure, meanwhile, the trains behind it cannot proceed because the track is blocked by the delayed train. For additive delay tests, the maximum train speed is reduced due to simulate engine failure at the first station, and the delay increases along the whole journey until the train arrives at its destination.

To observe the relationship between the computation time and the number of trains in the network, the total number of trains is considered when designing the benchmark scenarios. In the 7 train scenario, trains 01 to 07 (all trains in the first time period) are considered; in the 10 train scenario, three more trains (trains 11, 14, 16) are added because they are the first three trains in the second time period; then in the 13 train scenario, three more trains (trains 12, 15, 17) appear, as the time window which is considered is enlarged by 2 minutes; finally, the whole network with 14 approaching trains in two time periods is considered, and this is also used in the multiple trains delay experiments.

For 7-train, 10-train and 13-train networks, only a 2-minute delay is inserted to test the performance of different algorithms. Scenarios 1.1 to 1.3 are designed with 2 minutes of departure delay occurring to a single train, train 01, with an increasing number of trains in the network.

Scenario 1.1: Train 01 is delayed for 2 minutes with 7 trains in the network.

Scenario 1.2: Train 01 is delayed for 2 minutes with 10 trains in the network.

Scenario 1.3: Train 01 is delayed for 2 minutes with 13 trains in the network.

For the single delay tests, 2, 5, 10, 15, 20 and 30-minute departure delays are given to train 01 in scenarios 1.4.1 to 1.4.6 with the 14-train network model (see below).

Scenario 1.4.1: Train 01 is delayed for 2 minutes at Alexandra Palace.

Scenario 1.4.2: Train 01 is delayed for 5 minutes at Alexandra Palace.

Scenario 1.4.3: Train 01 is delayed for 10 minutes at Alexandra Palace.

Scenario 1.4.4: Train 01 is delayed for 15 minutes at Alexandra Palace.

Scenario 1.4.5: Train 01 is delayed for 20 minutes at Alexandra Palace.

Scenario 1.4.6: Train 01 is delayed for 30 minutes at Alexandra Palace.

To generate the delay scenarios 2.1 to 2.3 which have single additive delays, the following reductions in speed are given to train 01 in the additive delay region: 80% top speed remains; 50% top speed remains; and 20% top speed remains.

Scenario 2.1: Train 01 is limited to 80% full speed.

Scenario 2.2: Train 01 is limited to 50% full speed.

Scenario 2.3: Train 01 is limited to 20% full speed.

Scenarios 3.1 to 3.5 refer to the different initial delay which affect different trains (trains 01 and 04), as below:

Scenario 3.1: Train 01 is delayed for 2 minutes and Train 04 is delayed for 2 minutes.

Scenario 3.2: Train 01 is delayed for 2 minutes and Train 04 is delayed for 5 minutes.

Scenario 3.3: Train 01 is delayed for 2 minutes and Train 04 is delayed for 10 minutes.

Scenario 3.4: Train 01 is delayed for 5 minutes and Train 04 is delayed for 2 minutes.

Scenario 3.5: Train 01 is delayed for 10 minutes and Train 04 is delayed for 2 minutes.

Scenarios 4.1 to 4.3 test different additive delays which affect different trains (trains 01 and 04):

Scenario 4.1: Train 01 is limited to 50% full speed, and train 04 is limited to 50% full speed.

Scenario 4.2: Train 01 is limited to 80% full speed, and train 04 is limited to 50% full speed.

Scenario 4.3: Train 01 is limited to 20% full speed, and train 04 is limited to 50% full speed.

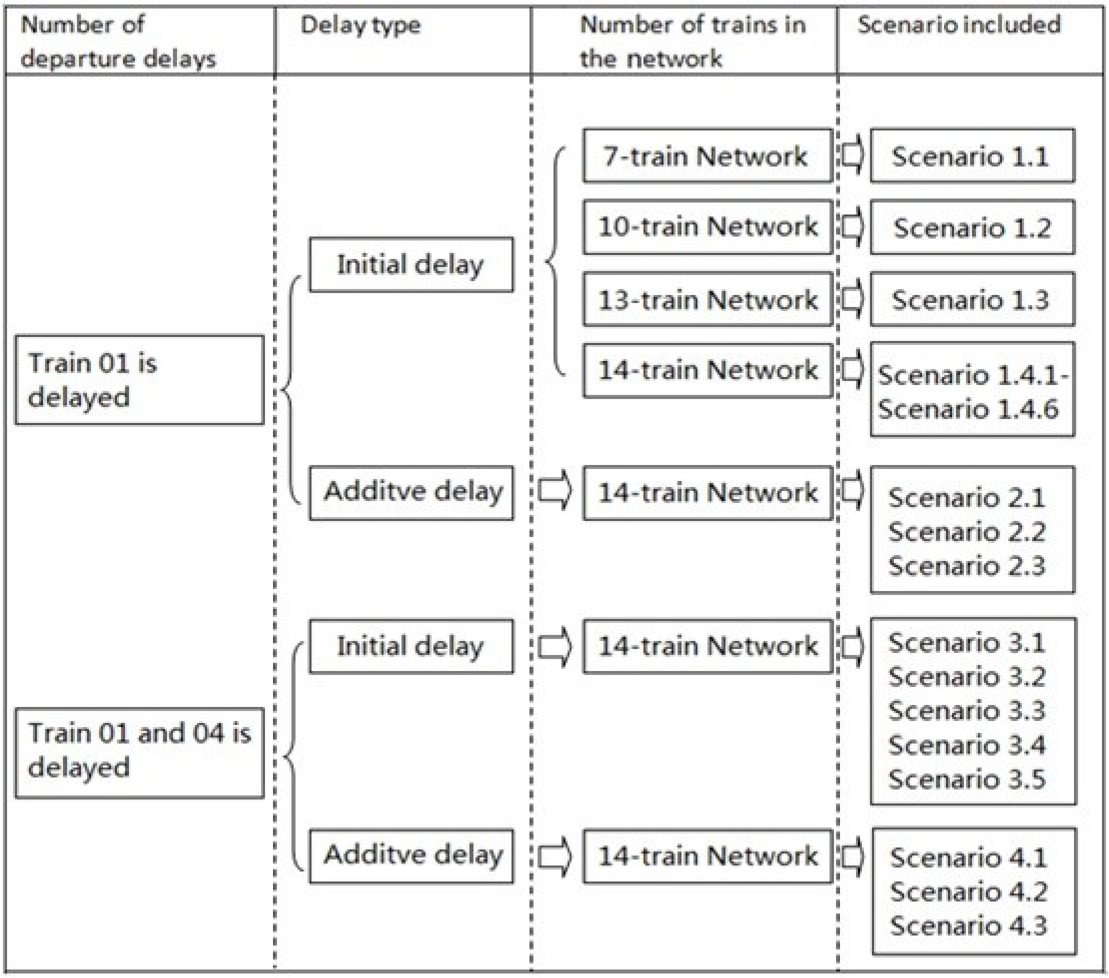


Figure 5-14: Benchmark scenarios classification diagram

A set of scenarios are designed to represent different realistic delay situations; by implementing different algorithms at the junction region. The rescheduling results are compared and analysed in the next sections.

5.4 Analysis and comparison of solutions

5.4.1 Key performance indicator

A series of benchmarking experiments are carried out and the results are discussed in this section. By evaluating different rescheduling algorithms with different delayed scenarios, a summary of the characteristics of each algorithm is presented according to the experimental results. Some assumptions are proposed for an automatic rescheduling system. To assess the performance of each algorithm, two indicators are used to analyse the experiment results, which are: Total Delay Penalty (TDP), Computing Time (CT). The visualisation of the delay propagation is observed and compared with the best algorithm addressed.

TDP

As a key indicator of the quality of the solution, TDP will be used to evaluate different algorithms. Considering the variation of the evolutionary algorithms, an average value from 100 results is taken as the main TDP for the evolutionary algorithms.

The Optimisation Rate (OR) is used to represent how much the TDP is improved; the bigger the OR is, the better the improvement made by the algorithm j. The equation is shown as:

$$OR(j) = \frac{TDP_0 - TDP_j}{TDP_0};$$

Equation 5-4

$OR(j)$ represents Optimisation Rate when method j is addressed;

TDP_j indicates the Total Delay Penalty of the system when method j is addressed;

TDP_0 indicates the Total Delay Penalty of the system when no rescheduling method is addressed; here, the train passes the junction according to the original timetable, thus TDP_0 =TDP_{TOE}

CT

CT, as a main indictor of the performance of the algorithm, helps to define the area considered in the later research. According to the algorithm operation process, the algorithm starts computing by the time the first train enters the junction region, and gives the results when the algorithm computation completes. In the experiment, time is frozen when an algorithm is running, but in reality, trains are still running while an algorithm is operating. The junction signal is deemed to be red when no junction route has been assigned, which means, the train will decelerate and stop by the junction signal until a passing route is given. For the best results, a train should receive the junction routing sequence from the control centre before it starts to brake. Due to the buffer time inserted in the timetable, a 5-second delay which happens in the junction area will have no effect on the order of passing trains.

The algorithm detection area in the experiment is defined as X km from the edge of the junction region to the junction signals. There are a series of trains running in the network, I. The train $i \in I$ enters the junction region with speed U_i and brake rate d_i, and it should stop by the head of the junction signal if it does not receive a command from a dispatcher. The braking distance S_i for the train i can be calculated by the equation below [117]:

$$S_i = (-U_i^2)/2(d_i - g * tan\alpha) \ , \quad for\ \alpha < 0$$

Equation 5-5

The equation for the running time RT_i of train i from it entering the junction to it beginning to decelerate without receiving any command is given as:

$$RT_i = \frac{X - S_i}{U_i}, \qquad i \in I$$

Equation 5-6

Therefore, the acceptable computing time CT will be no more than the minimum running time of a series of trains I, and a reasonable buffer time such that the train can recover the delay itself before it reaches the junction signal is added as well:

$$CT \leq min(RT_i) + t_{buffer}, \qquad i \in I$$

Equation 5-7

The term "$-$ g $*$ tanα" is the gravitational acceleration, assumed as 0 in this case.

U_i is the speed of the train i when the brakes are applied.

d_i is the brake rate of the train i.

S_i is the braking distance for the train i.

X is the distance between the edge of the junction region to the junction signal.

RT_i stands for the running time of the train i from entering the junction to when it starts to decelerate.

t_{buffer} is the added buffer time for the algorithm calculation, assumed as 5 seconds in this case.

According to the rolling stock configuration and junction information given in the section 5.3.2, CT of less than 15 seconds is deemed to be acceptable in this thesis. The best method should give the solution in an acceptable CT, and lead to the minimum TDP/ maximum OR.

5.4.2 Results comparison across different algorithms

All experiments' results have been recorded in tables and ranked by their performance (see Appendix C). In this section, the bar charts are used to compare the optimisation rate with different rescheduling approaches addressed in different scenarios and the line graphs are used to indicate how computation time changes in different scenarios.

(a) Results analysis for scenarios 1.1 to 1.4.1

Figure 5-15 shows the optimisation performance in different scenarios (optimisation rate vs. number of trains), and Figure 5-16 shows how the computation time changes when the number of trains is increasing (computation time in log vs. number of trains).

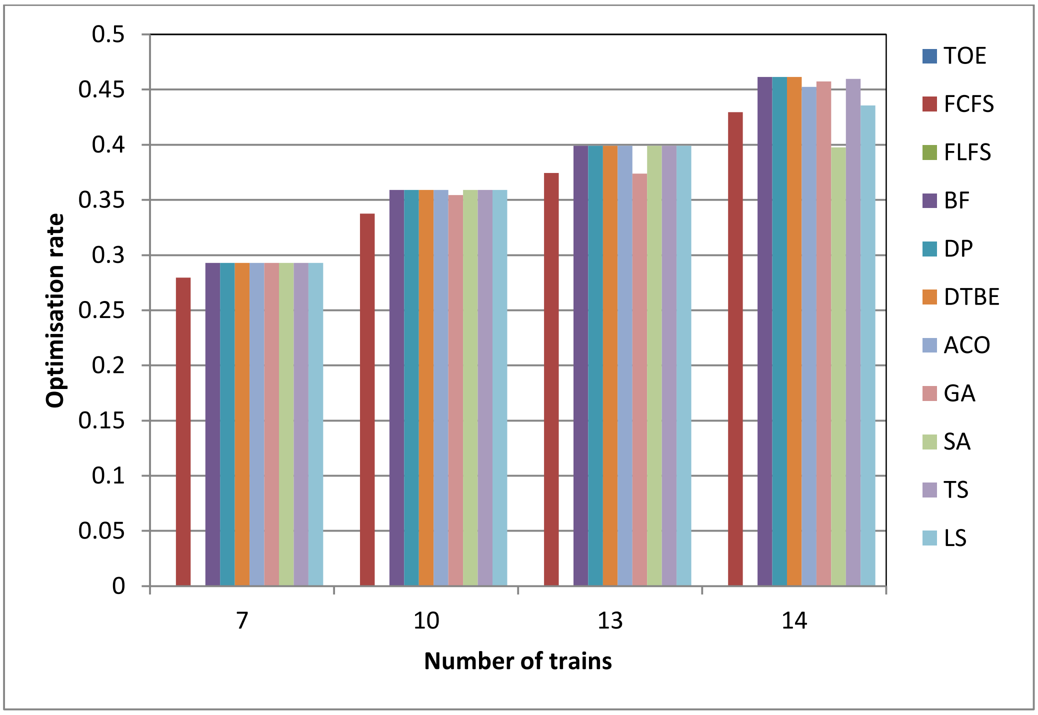


Figure 5-15 Optimisation rate vs. number of trains

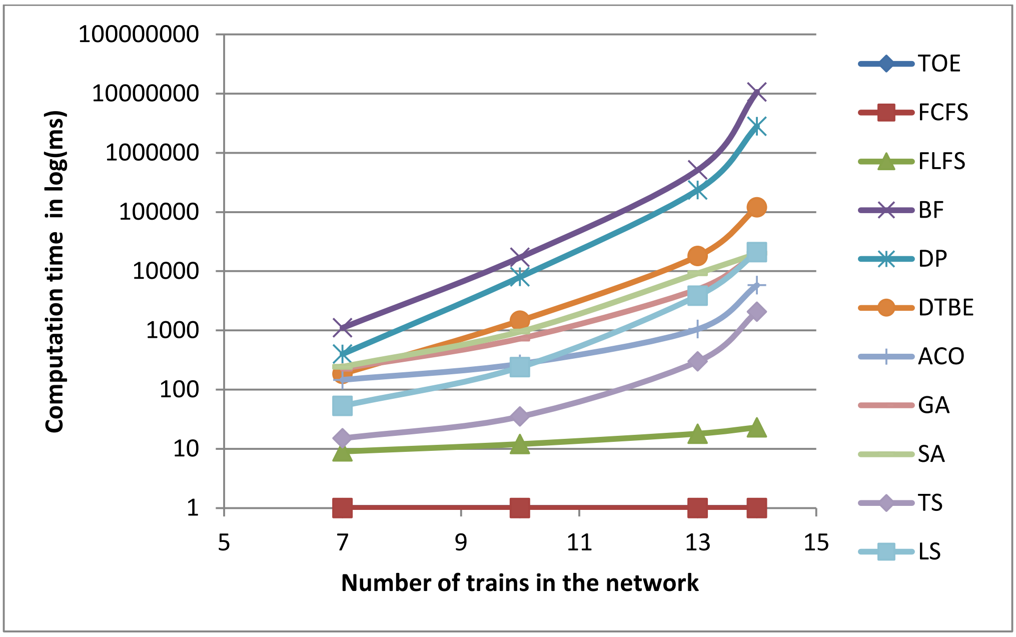


Figure 5-16 Computation time in log vs. number of trains

Analysis:

TOE and FLFS fail to find the optimal solution (optimisation rates stay at zero in Figure 5-15) due to the inability to change the original order of the passing trains. On the contrary, BF, DP and DTBE can always find the optimal solution (shown by a maximum optimisation rate) under a minor disruption situation in all scenarios. Evolutionary algorithms behave variably when the number of trains increases (for example, the GA performs worse than other evolutionary algorithms in 10-train and 13-train scenarios, but better in the 14-train scenario). When the number of trains is up to 14 in the network, all evolutionary methods failed to find the optimal solution. When the number of trains in the network increases, the optimisation rate is also slightly raised, and the OR of each of the algorithms does not appear to differ much under a minor disruption.

From Figure 5-16 it can be seen that the computation time of fundamental algorithms and evolutionary algorithms grows super exponentially when the number of trains in the network increases; in contrast, the computation time of TOE, FCFS and FLFS stays low while the number of trains increases.

(b) Results analysis for scenarios 1.4.1 to 1.4.6

Figure 5-17 shows the optimisation rates of different algorithms when the inserted delay increases and Figure 5-18 shows the relationship between the computation time in log and the size of the inserted delay.

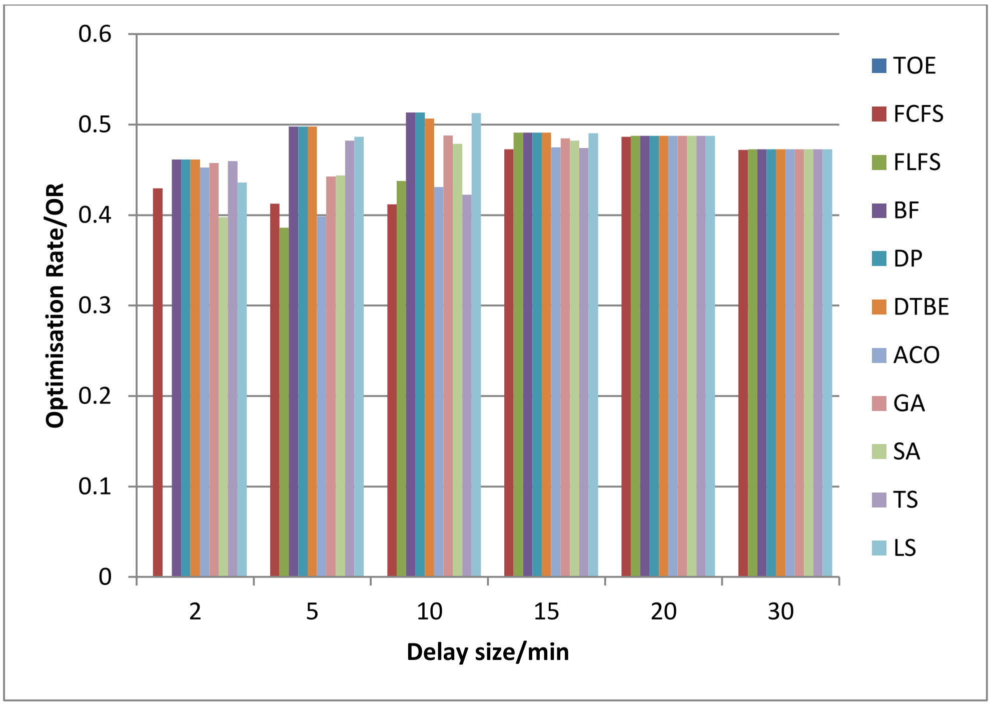


Figure 5-17 Optimisation rate vs. number of trains

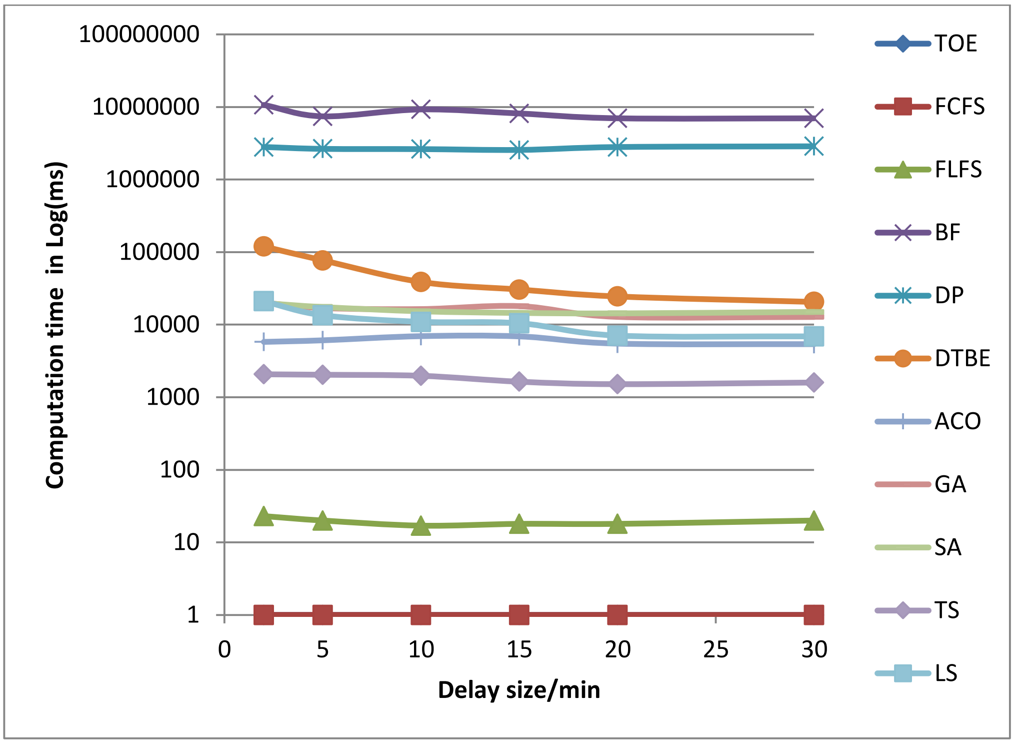


Figure 5-18 Optimisation rate vs. number of trains

Analysis:

When the delay is larger than 20 minutes, all rescheduling algorithms can provide the optimal solution. The reason is that the experimental network only runs a two time periods timetable, when the input delay on one train is too big, all trains that are running on the other two lines will pass the junction before the delayed train arrives at the junction, therefore there will be no conflict detected in the junction area. It is pointless to do the rescheduling if no conflict is present.

FLFS failed to find the optimal solution when the initial delay is small (OR is zero when initial delay is 2-minute), but the performance improves when the initial delay size increases. BF and DP can always find the best solutions, while for the other algorithms the performance is varied.

Evolutionary algorithms perform very differently and none of them can find the best solution in these scenarios and they provide solutions with ORs ranging from 0.4 to 0.5.

From Figure 5-18, it can be seen that CT is slightly declines when the delay increases, but there is not a significant reduction. An acceptable CT can be calculated with the information provided by the infrastructure and train configuration; with an acceptable CT the best algorithm can be chosen based on its OR.

(c) Results analysis for scenarios 2.1 to 2.3

Figure 5-19 shows the optimisation rate of different algorithms with different speeds of train. 01, and 0.8, 0.5, and 0.2 indicate the percentage of the original speed used. Figure 5-20 shows the relationship between computation time in log and the speed of train 01.

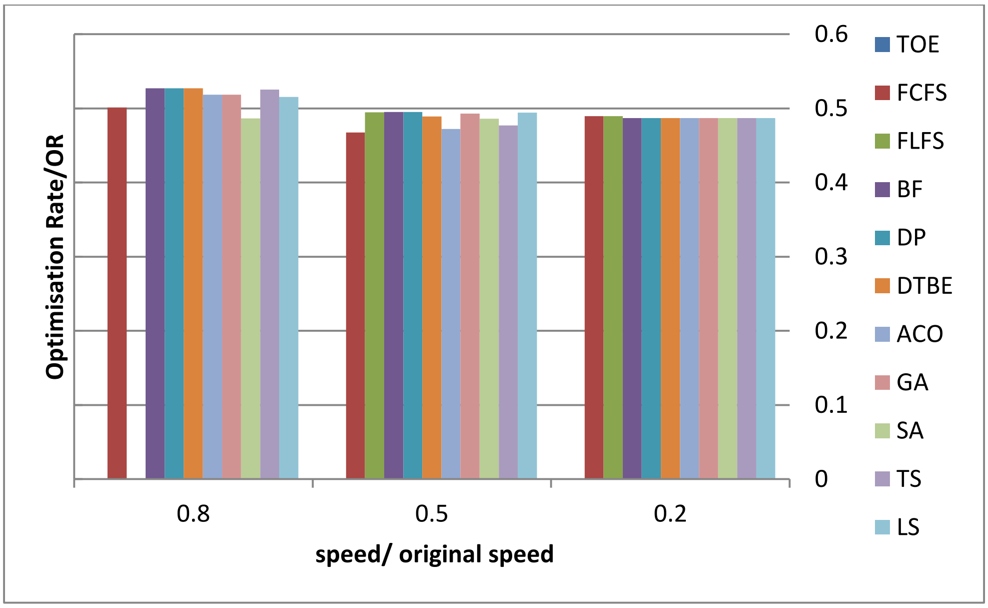


Figure 5-19 Optimisation rate vs. percentage remaining speed of train 01

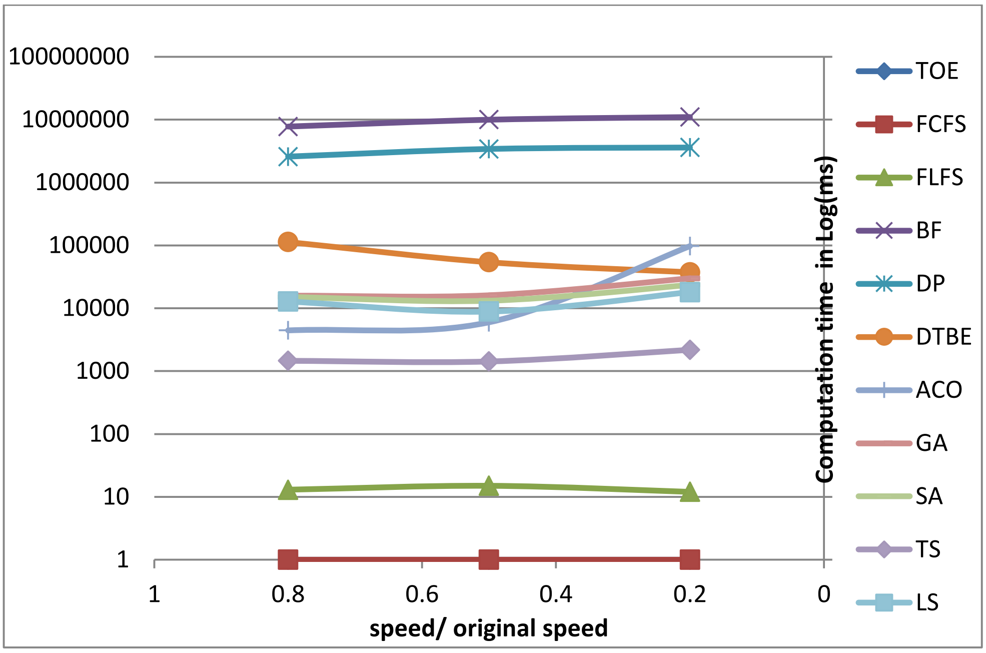


Figure 5-20 CT in Log (ms) vs. percentage remaining speed of train 01

Analysis:

The x axis speed/original speed ranges from 1 to 0, which means that the train speed drops down from full speed to zero, and the additive delay is increasing while the speed is reducing. Figure 5-19 shows that while additive delays increase, the optimisation rate is slightly decreased for all algorithms except FLFS. FLFS is poor when dealing with minor disruptions: In the 80% of full speed scenario, FLFS makes no improvement. BF and DP undoubtedly give the best result, although DTBE can also find the best solution and other algorithms give optimal solutions which are very close. When the speed is reduced to 50%, BF and DP can still find the best solutions, whereas other algorithms find their optimal solutions with a lower OR.

Because optimisation is undertaken using STS model whist evaluation is undertaken using BRaVE model, the BF and DP might not always yield optimal evaluation results. When speed is reduced even more, the estimation of the train running is less accurate, which means the algorithms provide a worse solution due to a less accurate prediction. For example, when speed is reduced to 20% of the original speed, BF and DP fail to find the best solution compared to FCFS and FLFS, due to poor prediction. Under this type of situation, where prediction is poor, simple rule-based methods can be more effective.

The computation time increases when the speed of the train is reduced, except for DTBE, which can be observed from Figure 5-20. The computation time for DTBE drops a little, while remain the same or even rise slightly.

(d) Results analysis for scenarios 3.1 to 3.5

Figure 5-21 shows the optimisation rates of different algorithms in scenarios 3.1 to 3.5 when two trains are delayed. On the horizontal axis, 1 to 5 represents the scenario 3.1 to 3.5. Figure 5-22 shows the relationship between computation time in log and these scenarios.

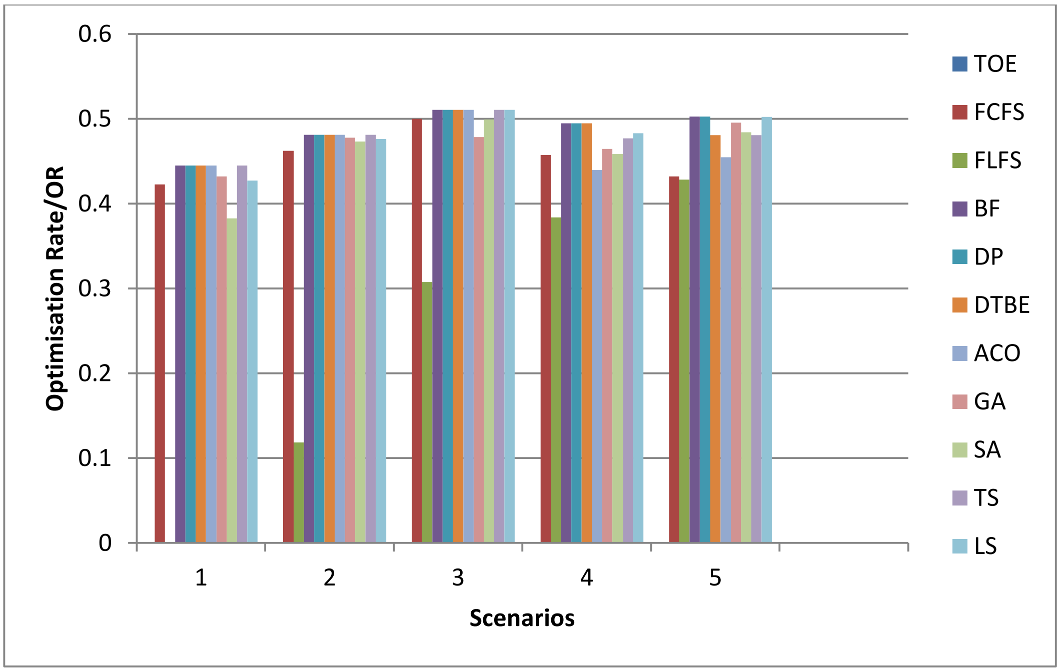


Figure 5-21 Optimisation rate vs. different scenarios

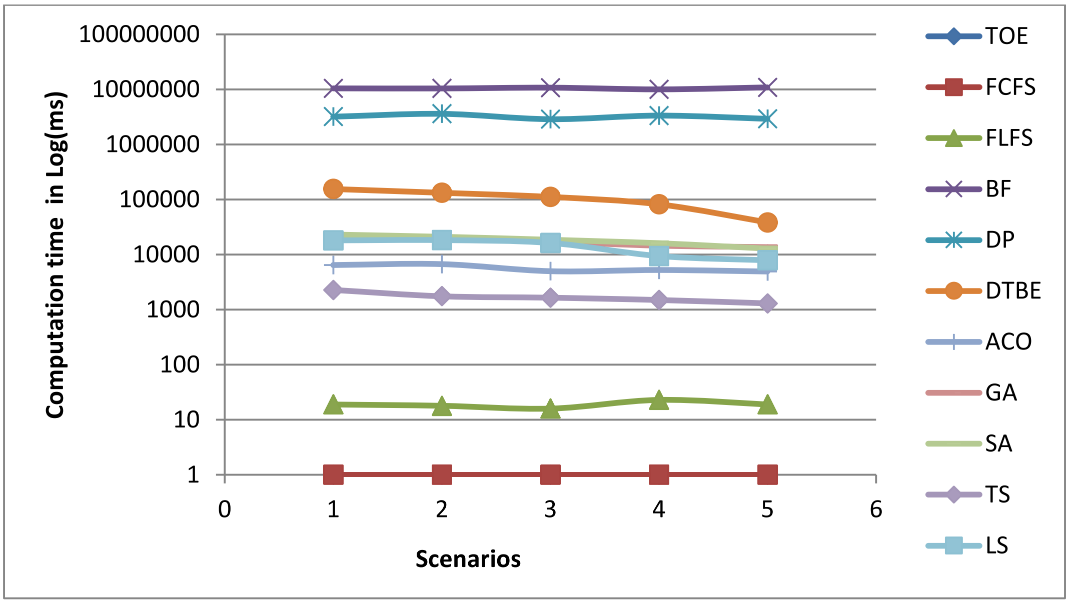


Figure 5-22 CT in log (ms) vs. different scenarios

Analysis:

To test multiple inserted delays, five representative scenarios with two trains which are delayed at the beginning are tested in this section. The OR of FLFS is increases along the five scenarios and the ORs of other algorithms increase in the first three scenarios and decrease in the fourth, moving back up in the fifth. Scenarios 2 and 4 and scenarios 3 and 5 share the same total inserted delays respectively, and this may be the reason why the ORs for the algorithms are similar to each other in the two scenarios.

In Figure 5-22, BF takes the longest time to search for the optimal solution, and for others, CTs are different to each other, but show no large differences when the same algorithm is used, but for different delay scenarios

(e) Results analysis for scenarios 4.1 to 4.3

Figure 5-23 shows the optimisation rate of different algorithms in scenarios 4.1 to 4.3 with two additive delays occurring at the same time, and Figure 5-24 shows the computation time in log for the different scenarios.

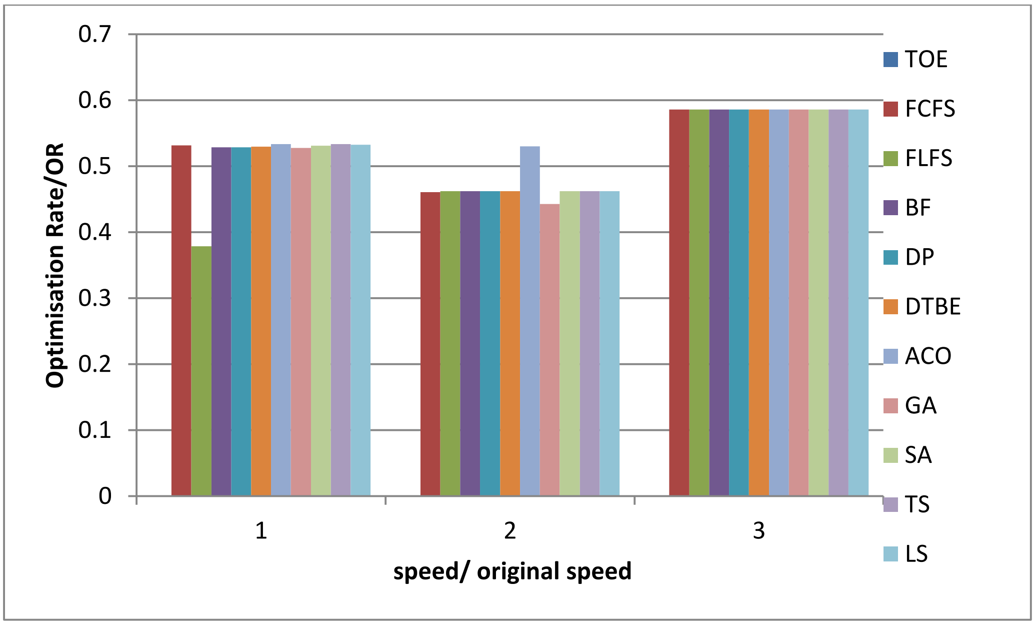


Figure 5-23 Optimisation rate vs. different scenarios

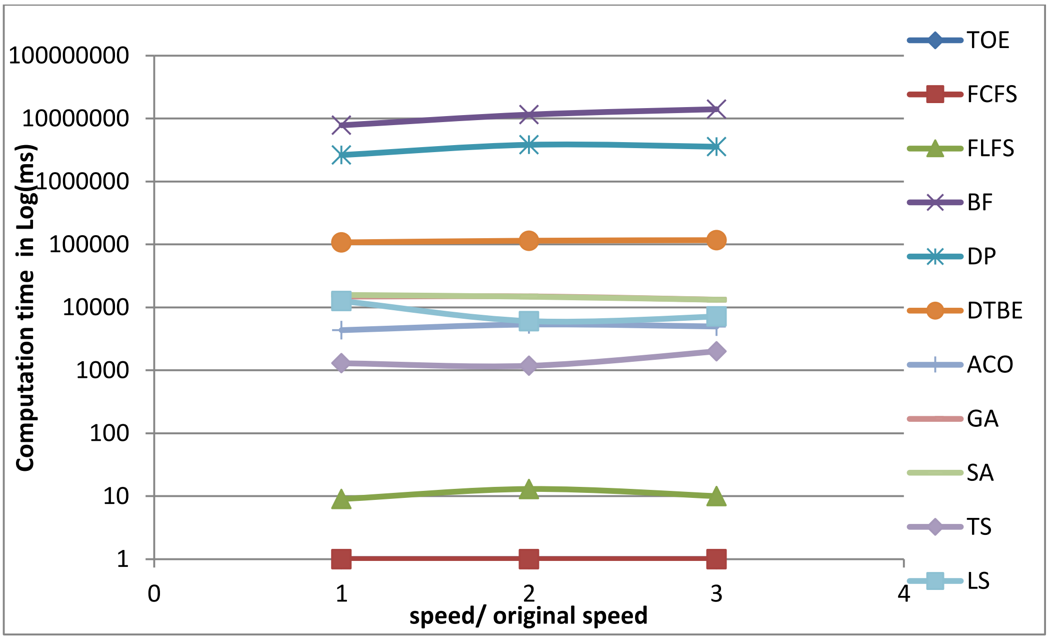


Figure 5-24 Computation time in log (ms) vs. different scenarios

Analysis:

To show algorithm performance in a scenario with multiple inserted additive delays, three representative scenarios with two trains which are delayed at the beginning because of engine failure are tested in this part. Train 01 retained 80%, 50% and 20% of its speed in these three scenarios respectively and train 04 retained 50% of its speed in all scenarios.

With increasing additive delays input into the system, the performance of FLFS improves along the three scenarios. In scenario 1, due to a minor estimation error, all algorithms, except FLFS and TOE, give similar optimal solutions with the OR around 0.53. In scenario 2, ACO performs relatively better than the other algorithms, which have ORs of around 0.46, dropping down slightly compared with scenario 1. In scenario 3, due to a big loss on the speed of train 01, the line is blocked for a long time, which leads to no conflict being detected in the junction area. All algorithms give the same rescheduling solution, except TOE. From Figure 5-24, the computation times are observed to be very close to the single additive delay scenario in 2.1 to 2.3.

5.4.3 Summary

After analysing the performance of the different algorithms across different delay scenarios, a summary of the application and characteristics of each algorithm in Table 5-4 below (note that this conclusion is restricted to the experiments in this chapter):

TOE:	Trains pass the junction with an enforced order with no flexibility, leading to a chain of delays since the on time trains need to wait at the junction for the delayed trains. The OR stays at zero because no rescheduling action is taken with TOE, which means that no improvement is made by executing this approach.

FCFS	FCFS only considers the junction arrival time for each train without considering the delay penalty at the final destination. It can be calculated quickly but it is not accurate. Although it is a simple way to avoid conflict, it does not consider the whole network, therefore it is not recommended in most scenarios.
FLFS	The FLFS algorithm considers the destination arrival time for each train. When the delay is small, it will give an order very close to the timetable order, since the predicted destination arrival time is close to the timetable arrival time, which means that little or no improvement is made when FLFS is applied in a small disruption scenario. When the delay increases, the performance of FLFS improves. However, it still does not consider the delay penalty for each train, therefore it cannot calculate the best solution in most minor disruption situations.
BF	As an exhaustive method, BF can find the best solution in most situations with a good prediction. However, the computation time is obviously bigger than other approaches. In scenario 1.1 and 1.2, the CT is within an acceptable region, but in other scenarios, it is too big to be practicable.
DP	As for BF, DP can find the best solution in most situations with a good prediction. It takes less time to compute the solution than BF, but the computation time is still too big to accept in 13-train and 14-train scenarios.
DTBE	DTBE can find the best, or a relatively good solution, in most situations. It is a very stable algorithm which can be applied to solve most cases if the CT is acceptable. According to the complexity of the different scenarios, the computing time varies from 20 s to 100 s in 14-train scenarios.
ACO	ACO takes a relatively short time (around 10s) for the computation in 14-train scenarios. It performs well when the delay is small, however, when delay increases, the performance deteriorates.
GA	The GA algorithm performs averagely in all scenarios; it was neither the best nor the worst performing. It spends nearly an equal amount of time on computation (around 15 s) in almost all situations.
SA	In the minor disruption scenario, SA performs very badly, although it performs better when the delay increases.
TS	This takes the least time of all evolutionary algorithms, the performance of TS deteriorates when the delay increases. In the minor disruption scenarios (2-minute

	initial delay scenario and 80% of full speed additive delay scenario), it can always find the best solution, or a relatively good solution in a short time.
LS	Using LS, the performance improves and the computation time is reduced when the delay increases. It gives a satisfactory solution in an acceptable time when the delay is not too small, and it is recommended in most scenarios.

Table 5-4 Summary of the application

5.5 Conclusion

In this chapter, the author introduced the application of ten algorithms and one standard method (TOE) to a set of benchmark problems. These algorithms include rule-based methods (FCFS, FLFS), an exhaustive algorithm (BF), a graphic method (DP), a branch and cut method (DTBE) and evolutionary methods (ACO, GA, SA, TS LS). The rescheduling approaches used for the experiment tests and the innovation of the application are generally introduced and classified in Section 5.1.

A cost function is defined to evaluate the performance of different algorithms in Section 5.2. In Section 5.3, a benchmark scenario is developed for assessment of the algorithms in order to estimate their performance under different disturbed conditions. The junction area studied in this chapter is Finsbury Park junction on the ECML in the UK, with 14 trains passing through in 30 minutes. The experimental methodology of simulation, train running estimation and algorithm implementation are also explained.

In Section 5.4, a series of scenarios are designed and tested with different algorithms addressed. Analysis and comparison of different solutions is presented, using two performance indicators: Computation Time (CT) and Total Delay Penalty (TDP). The characteristics of each algorithm are derived from the summaries of each group for comparison. Currently, the developed methods have been tested mainly in an experimental setting, showing promising results, both in terms of their solution quality and in terms of their computation times. A summary of the performance of these algorithms is given at the end of this section, and can be used as references in further operation in this junction region.

To conclude, BF and DP are the most appropriate algorithms in the majority of delay scenarios because they can always find the optimal solution and the computation time is acceptable due to the limited scale of the network and scale of the rescheduling area. However, in simulation, the capacity is enhanced by the computer, which leads to an enormous amount of computation time when BF and DP are used. As a result, BF and DP are no longer appropriate for these delay scenarios. DTBE is also a good method to use because it can find the best solution in most scenarios, the disadvantage of this algorithm is that it still needs a long time to find the solution. FCFS and FLFS are commonly used approaches for junction rescheduling, as they are both quick and simple. However, in most scenarios, the results they found are not as good as other methods. ACO, GA, SA, TS and LS need to be well constructed to ensure the stability of the evolutionary algorithms. According to CTs, these algorithms rank as: TS $< ACO < LS < GA \approx SA$ when the number of trains in the network are the same. According to the TDPs, these algorithms perform discrepantly according to different delay conditions.

The characteristics of the applications for each algorithm can be concluded from their comparison, and some suggestions for choosing algorithms to use in the dynamic rescheduling can be formulated. For example, Table 5-5 shows an example of choosing algorithms for different networks and delay conditions after considering the characteristics of application for each algorithm.

Network layout $(\boldsymbol{L})$ / The number of trains in the network (**N**) The number of delays ($\boldsymbol{n}$) / Delay type (**Departure/ Additive**) / Delay size $(\boldsymbol{d}/\ \boldsymbol{a})$			$\boldsymbol{L}$ = ECML			
			$\boldsymbol{N} \leq 9$	$\boldsymbol{N} = 10{,}11$	$\boldsymbol{N} = 12$	$\boldsymbol{N} = 13{,}14$
$\boldsymbol{n} = 1$	Departure (d_1 is the delay time of Train 01)	$\boldsymbol{d_1} < 5\ mins$	BF, DP	DP	DTBE	TS
		$5\ mins \leq \boldsymbol{d_1} < 15\ mins$	BF, DP	DP	DTBE	LS
		$\boldsymbol{d_1} \geq 15$ mins	BF, DP	DP	DTBE	FLFS
	Additive (a_1 is the percentage of the remaining speed of Train 01)	$\boldsymbol{a_1} \geq 80\%$	BF, DP	DP	DTBE	TS
		$80\% > \boldsymbol{a_1} \geq 50\%$	BF, DP	DP	DTBE	LS
		$50\% > \boldsymbol{a_1} \geq 20\%$	BF, DP	DP	DTBE	FLFS
		$\boldsymbol{a_1} < 20\%$	BF, DP	DP	DTBE	FLFS, FCFS
$\boldsymbol{n} = 2$	Departure (d_1 is the delay time of Train 01, d_2 is the delay time of Train 04)	$\boldsymbol{d_1} + \boldsymbol{d_2} \leq 4\ mins$	BF, DP	DP	DTBE	ACO, TS
		$4\ mins < \boldsymbol{d_1} + \boldsymbol{d_2} \leq 7\ mins$	BF, DP	DP	DTBE	TS, LS
		$7\ mins < \boldsymbol{d_1} + \boldsymbol{d_2} \leq 12\ mins$	BF, DP	DP	DTBE	LS
	Additive (a_1 is the percentage of the remaining speed of Train 01, a_2 is the percentage of the remaining speed of Train 04)	$\boldsymbol{a_1} = 80\%$ $\boldsymbol{a_2} = 50\%$	BF, DP	DP	DTBE	TS
		$\boldsymbol{a_1} = 50\%$ $\boldsymbol{a_2} = 50\%$	BF, DP	DP	DTBE	ACO
		$\boldsymbol{a_1} = 20\%$ $\boldsymbol{a_2} = 50\%$	BF, DP	DP	DTBE	FLFS

Table 5-5: Algorithm recommendation table

CHAPTER 6

STUDY OF SINGLE-JUNCTION RESCHEDULING USING PERFORMANCE-BASED SUPERVISORY CONTROL

In chapter 5, a number of rescheduling approaches are investigated on a single-junction area with idealised the timetable and compressed periods. However, in real working conditions, railway operations are highly uncertain and complex, and delays are hard to predict and avoid due to technical failures, weather conditions and passenger performance. These result in operational dynamics such as uncertain delays, changing traffic flows, unpredictable changes in traffic, etc. In this chapter, in order to deal with these operational dynamics, supervisory control is adopted to provide rescheduling decisions over a wider time window through the application of different rescheduling strategies in appropriate sequences.

In order to explain the fundamentals of the approach, initially a brief overview of supervisory control is provided. A methodology for using supervisory control as part of a junction rescheduling problem is developed and demonstrated. A single-junction case study is designed to demonstrate how this process is realised.

6.1 Supervisory control overview

Supervisory control is a common method to select between different control strategies within a single system. It is particularly useful in applications where it is beneficial to try different control strategies to find solutions over a broad solution space, and therefore avoid local minima, or where a single control strategy is unlikely to be able to find a satisfactory solution in all possible situations [118]. Supervisory Control usually takes one of two forms: In the first, ‘switched supervisory control’ the control process proceeds autonomously and is monitored by an ‘observer’; when necessary, the observer intervenes to select an alternative control strategy. In the other, ‘schemed supervisory control’’, the process follows a set of predefined control instructions, which can be changed by an operator if required.

In this thesis, switched supervisory control is discussed and an approach to address the railway junction rescheduling problem is developed. A more detailed introduction to this type of supervisory control is presented in the following section, along with a demonstration of the effect of using different types of supervision.

6.1.1 Introduction to supervisory control

A standard architecture for supervisory control is shown in Figure 6-1. A key feature of the approach is the bank of alternative controllers, and the ability to switch between them in real-time, based on the current measured outputs. When an uncertain disruption w is detected by a controlled process, the measured output y is sent to a supervisor where it is combined with the control signal u from the current controller, and the supervisor provides a switching signal σ to select expected to be the most appropriate controller from the bank of candidate controllers [119].

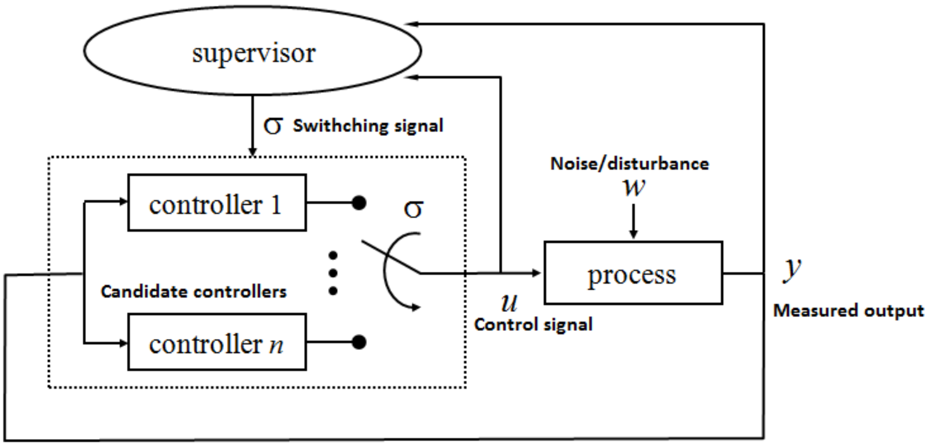


Figure 6-1: Supervisory control architecture

6.1.2 Type of supervision

Three types of supervision are generally used in switched supervisory control; they are pre-routed supervision, performance-based supervision and estimator-based supervision.

Pre-routed supervision:

In pre-routed supervision, the supervisor tries one controller after another in a pre-defined sequence, and stops when the performance becomes acceptable against a cost function (See Figure 6-2) [120]. Algorithms can be used to help determine the searching sequence. However, in practice pre-routed supervision is usually restricted to a small number of candidate controllers, so as to minimise the time taken to find an acceptable controller.

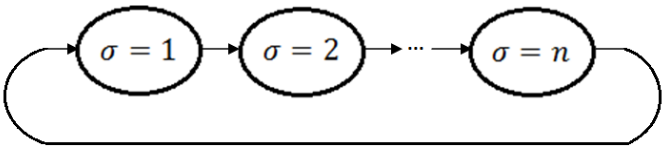


Figure 6-2: Pre-routed supervision

Performance-based supervision:

Performance-based supervision attempts to directly assess the performance of each candidate controller, but it does not attempt to estimate the model of the process in order to achieve this [121]. The principle of performance-based supervision is to retain the existing controller while observed performance continues to be acceptable, however as soon as the performance becomes unacceptable an alternative controller is selected that will provide improved performance based on the available data provided from the measured output and the feedback from the controlled signal.

Figure 6-3 shows a block diagram that represents performance-based supervision. For all candidate controllers $C_q \in C$, π_{q1} to π_{qn} represents the expected performance indicators of controller C_q by using the measured output, y, and control signal, u, as inputs into a performance monitor. The decision logic is responsible for generating the switching signal, so if σ: π_q is acceptable, then the current controller is retained; if it is not, an alternative controller, C_q, is selected corresponding to the best π_q.

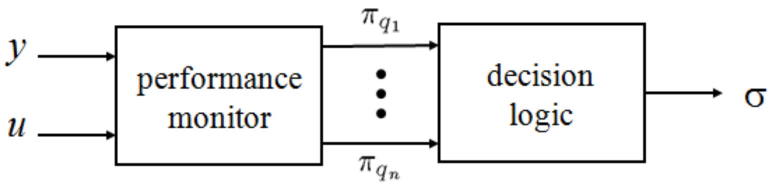


Figure 6-3: Performance-based supervision

Estimator-based supervision:

Estimator-based supervision is an indirect supervision approach that estimates a control model from observed data, then selects a controller based on the current estimation. This type of supervision was developed to improve the performance obtained by pre-routed supervision. To select an appropriate estimate model, estimator-based supervisors continuously compare the outputs of the process with the outputs of several process models to determine which model most closely matches the actual process [119]. A control process which has uncertain parameters is assumed to be in a family of systems and for each process in this family at least one candidate controller, $\mathrm{C_p}$, with a defined controller selection function can provide adequate performance.

The structure of estimator-based supervision can be represented by the diagram in Figure 6-4. It includes a multi-estimator which is responsible for determining which admissible model best describes the actual process: y_p is the estimate of the output y, and e_p is the estimation error between the estimated output y_p and the actual output y. e_p is small if the estimate process matches the actual process. A decision logic that generates a switching signal, σ, is used to select the most appropriate candidate controller.

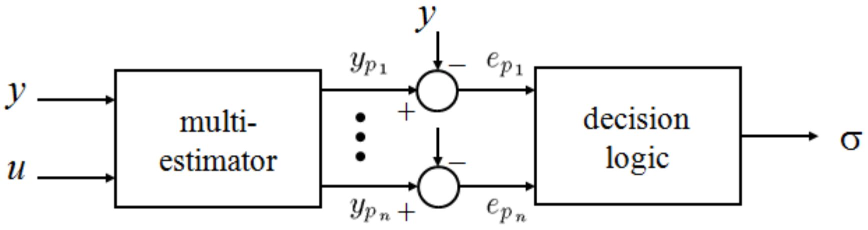


Figure 6-4: Estimator-based supervision

6.2 Performance-based supervisory control for junction control

6.2.1 Modelling framework

A general supervisory control architecture was introduced in the previous section. In this section the unique structure of a supervisory control process for railway rescheduling is developed (see

Figure 6-5). When a railway network is disturbed by a perturbation w, the output y from the network performance assessment is detected and analysed by the global control centre, and a calling signal τ is sent to the individual supervisor from the global control centre.

The switching signal σ is given by each supervisor to select a control mode in the control region. When σ is "1", all trains in the control region will be rescheduled through intelligent local rescheduling control, which is supported by an advanced algorithm chosen by the control centre; when σ is "0", it is as long as no rescheduling happens before the trains reach the junction and all trains will pass the junction following the current timetable order. After a reordering/retiming process, the rescheduled result u is sent back to the network to execute. The network database will update while the results are processing. The network output, y, is considered and observed by a global control module and each local control module every time step.

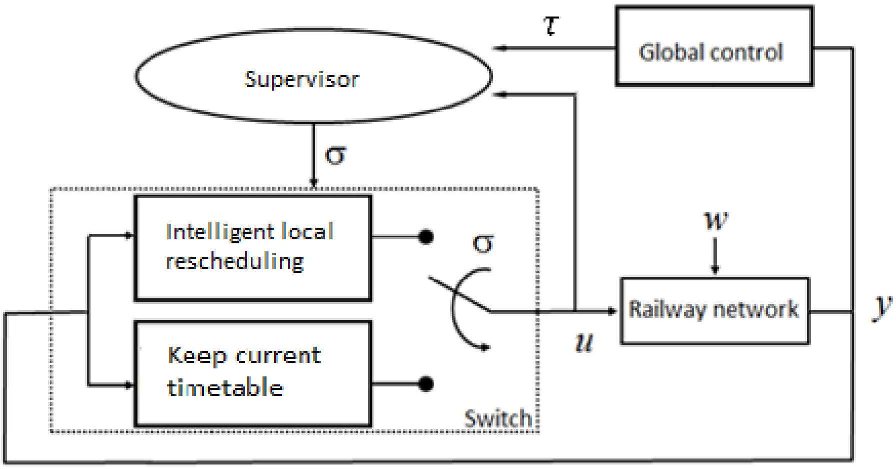


Figure 6-5: Supervisory control for railway rescheduling

Differing from traditional performance-based supervisory control, the supervision has been divided into two steps:

Step 1: Global control supervision

Step 2: Local control supervision

Step 1: Global control supervision

The global control module adopts this type of supervision to control different supervisors. Figure 6-6 shows the structure of global control supervision. The measured output y is performed and evaluated in the global performance monitor (G-PM). π_q equals the measure of the expected performance of the controller inferred from the current data.

The global decision logic (G-DL) is relatively simple: if π_q is acceptable then the current controller is retained; if it is not, a rescheduling calling signal τ equals "1" is sent to the supervisor. Otherwise, the calling signal τ remains "0".

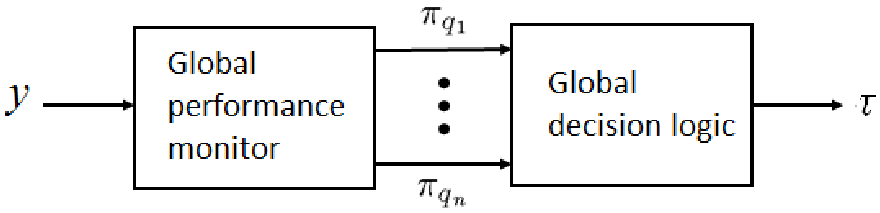


Figure 6-6: Global control supervision

Step 2: Local control supervision

Local control supervision is adopted by an individual supervisor to directly control each decision centre. By receiving signals from the global control centre and feedback from the decision centre, the supervisor should make decisions on which controllers to retain. Similar to global control supervision, in Figure 6-7, the inputs for the local control supervision include the rescheduled results, u, from the decision centre and the calling signal, τ, from the global control centre. The local performance monitor (L-PM) measures the performance of the rescheduling results and passes these measures (π_l) to the decision logic, and the calling signal, τ, is directly delivered to the local decision logic (L-DL) part. The decision logic module makes a decision as to whether the value of the switching signal, σ, is "1" or "0".

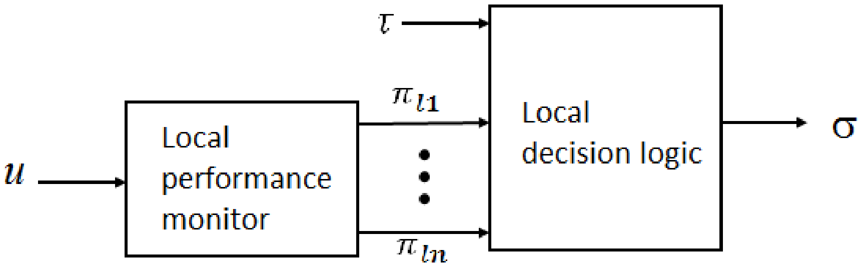


Figure 6-7: Local control supervision

Considering these two types of supervision, the architecture for the abstract supervision of the whole system can be shown, as in Figure 6-8.

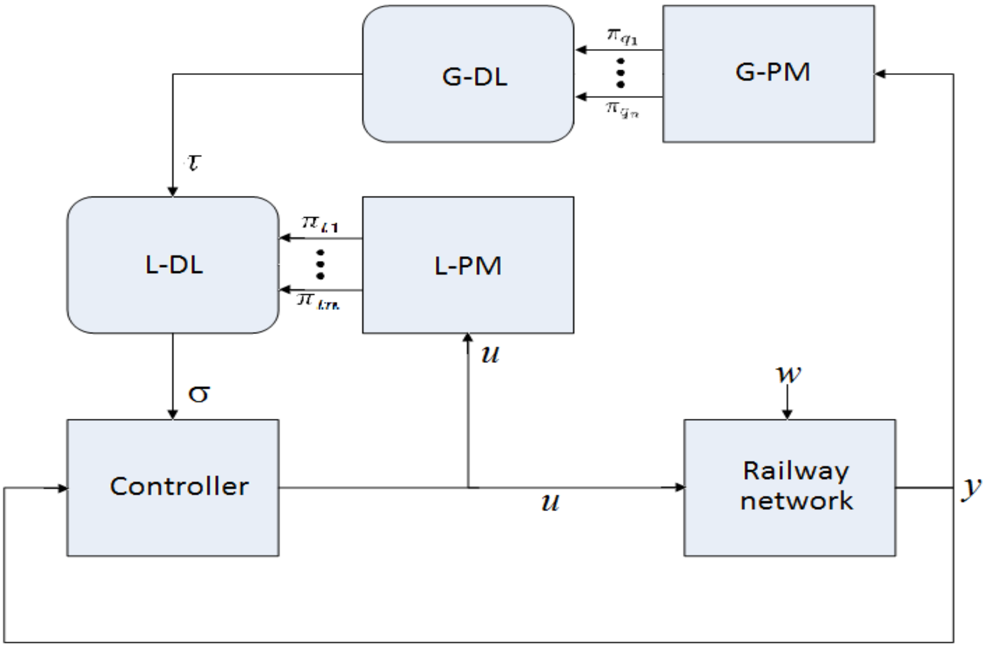


Figure 6-8: Abstract supervision architecture

6.3 Performance monitor

Due to the supervision type used for the rescheduling process, the supervisor should keep the controller while the observed performance is acceptable. When the performance of the current controller becomes unacceptable, the supervisor should switch to the controller that leads to the best outcome. The architecture of the supervision system was presented in the previous section (see Figure 6-5). In this section, two main performance monitors are developed to measure the performance of the input signals.

6.3.1 G-PM

The daily operation of the railway network is a complex and unpredictable system. With a disruption, w, and a controlled result, u, the running operation is simulated in BRaVE and delay propagation is also captured. G-PM can be replaced by the delay propagation visualisation tool, and there are some measures which can be extracted from the G-PM to evaluate the performance of the train operation. The key measures consist of two parts: delay condition and recovery condition.

In more detail, the delay condition indicates the Primary Delay Detection (PDD) and the Total Delay (TD) of the whole control region at time t; the recovery condition is set by the estimated Recovery Time (RT) for the current delay. A railway network running a number of trains $TR = \{tr_1, tr_2, \dots, tr_n\}$with a time window $T = [t_{start}, t_{end}]$, and there are a number of primary delays happened during the operation $PD = [d_1, d_2, \dots, d_n]$. For each primary delay, d_m, the time when it is firstly recorded is t_{d_m}. The sum of lateness $L(t)$ can be calculated by the Equation 3-1, Equation 3-2 and Equation 3-3, the total recovery time RT_{total} is defined as the time between the first primary delay of the system increasing above a small threshold value, and it returning below this threshold (the recovery threshold). All these measurements can be derived from the delay recovery graph by using the delay propagation visualisation tool.

$$PDD_t = \begin{cases} 1, & when\ t = t_{d_m},\ d_m \in PD \\ 0, & otherwise \end{cases}$$

Equation 6-1

$$TD_t = L(t)\,, \qquad t \in T$$

Equation 6-2

$$RT_t = \begin{cases} 0, & t_{start} \le t < t_{d_1} \lor RT_{total} < t \le t_{end} \\ RT_{total} - t, & t_{d_1} \le t \le RT_{total} \end{cases}$$

Equation 6-3

PDD_t indicates the primary delay detection at time t; when $PDD_t = 1$, a primary delay is detected as starting in the network. When $PDD_t = 0$, no primary delay is present at the current time (see Equation 6-1). In Equation 6-2, the real-time total delay measurement of the whole control region at time t is presented as TD_t, which equals the current sum of lateness $L(t)$. In Equation 6-3, RT_t stands for the estimated recovery time of the current delay at time t, and it is calculated by using the total recovery time minus the current time.

For example, a scenario has 7 trains running in the network from 7:00am to 8:00am with two trains delayed at two different stations, and no rescheduling is applied for the delay situation. The network is the same as in Section 5.3.2 and the original conflict free timetable of these 7 trains is the same as the timetable in period 1 in Section 5.3.2. The two delayed trains are train 01 and train 04, which are delayed at Brookmans Park Station and Alexandra Palace respectively.

The delay propagation is visualised as set out below (see Figure 6-9) in G-PM. The green line indicates the total delay by time: $TD(t)$; the red line and the purple line refer to the two delayed trains T1 and T4 respectively. T1 is delayed from 07:05:58 to 07:10:58 for 300 seconds and T4 is delayed from 07:21:43 to 07:23:43 for 120 seconds. The x axis is shown in hours and the y axis is shown in seconds. The observation window T ranges from 07:00:00 to 08:00:00 ($t \in T = [7 , 8]$). In this scenario, a primary delay on T1 is detected at 07:05:58 ($t = 7.099$) and another on T4 is detected at 07:21:43 ($t = 7.362$). The real-time total delay measurement of the whole control region TD_t, which equals the sum of lateness $L(t)$, shown as the green line in Figure 6-9. The recovery threshold is set to 60 seconds, and the time in which the delay has been totally absorbed is 07:35:51 ($t = 7.598$).

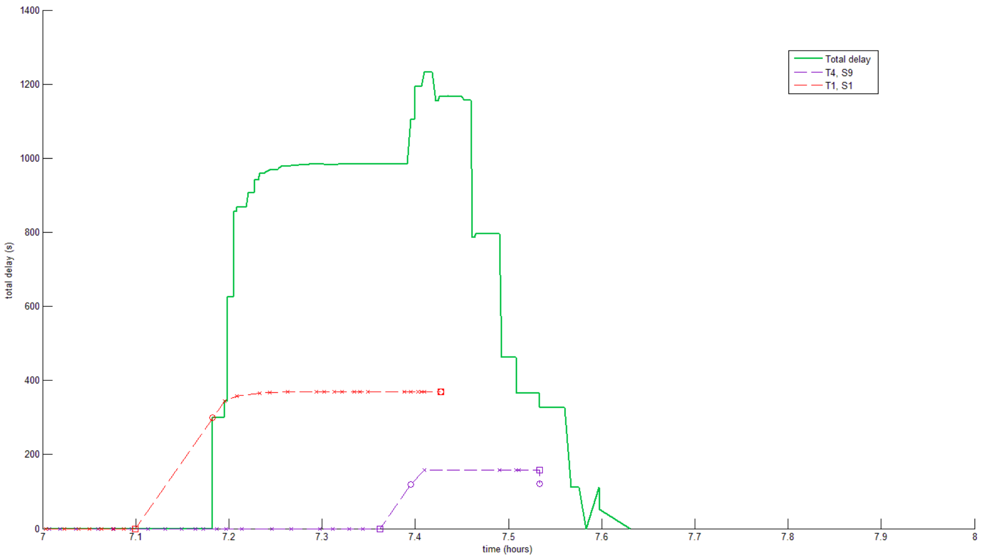


Figure 6-9: Delay propagation in the example scenario

6.3.2 L-PM

The L-PM monitors the junction condition in real-time when the rescheduling results are received. The main measure for the junction condition is defined as the junction detection factor

(JD). Trains pass the junction area with a given order, the time when train tr_i is entering the junction area is defined as t_{tr_i}. The junction detection factor for this junction can be defined as Equation 6-4.

$$JD_t = \begin{cases} 1, & when\ t = t_{tr_i},\ \ tr_i\ \in TR \\ 0, & otherwise \end{cases}$$

Equation 6-4

At time t, if a train has been detected as reaching the edge of the junction area, $JD_t = 1$, or not, $JD_t = 0$; In this delayed scenario, with no rescheduling strategy applied, trains passed the junction based on the timetable or der: $S = [T1,\ T2,\ T6,\ T3,\ T4,\ T7,\ T5]$, and the junction detection times for each train are: 07:16:42 ($t = 7.278$), 07:18:36 ($t = 7.31$), 07:21:08 ($t = 7.352$), 07:22:27 ($t = 7.374$), 07:23:35 ($t = 7.393$), 07:26:16 ($t = 7.438$) and 07:27:26 ($t = 7.457$). Figure 6-10 shows the junction detection times for the scenario described in the previous section; the time difference between the switching signals can be observed.

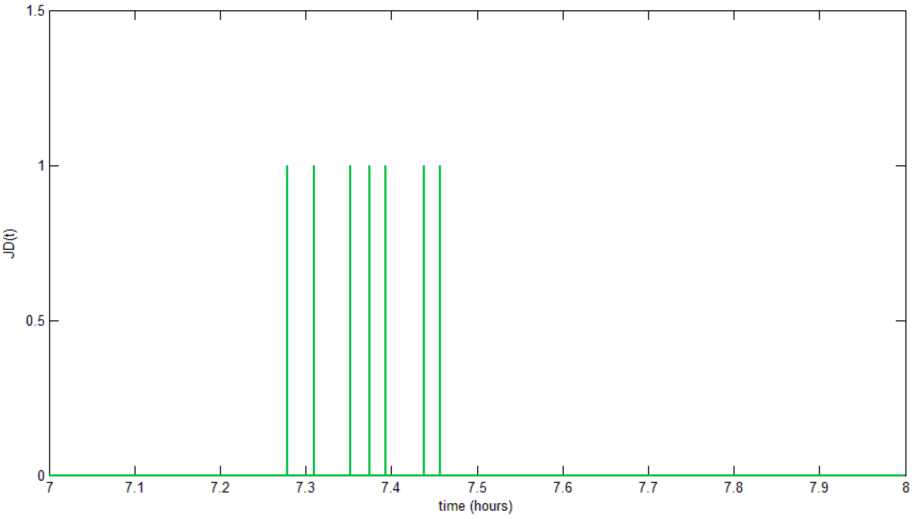


Figure 6-10: Junction detection in the example scenario

6.4 Decision logic

6.4.1 G-DL

The global decision logic (G-DL) for the global control takes PDD_t, TD_t and RT_t as the inputs and the rescheduling calling signal, τ, as the output. τ takes the value "1" or "0". The G-DL is defined with the following rules:

Rule 1: If TD_t is beyond the tolerant limit, the output τ is defined as "1"; (d_{limit} refers to the delay tolerant limit for the network).

$$TD_t \geq d_{limit} \rightarrow \tau(t) = 1$$

Equation 6-5

Rule 2: If TD_t has not reached the tolerant limit, but RT_t has exceeded the acceptable recovery time limit, the output τ is defined as "1"; (r_{limit} refers to the recovery time limit for the network.)

$$\begin{cases} TD_t < d_{limit} \\ RT_t \geq r_{limit} \end{cases} \rightarrow \tau(t) = 1$$

Equation 6-6

Rule 3: If TD_t has not reached the tolerant limit, and RT_t has not exceeded the acceptable time limit, the output τ is defined as "0";

$$\begin{cases} TD_t < d_{limit} \\ RT_t < r_{limit} \end{cases} \rightarrow \tau(t) = 0$$

Equation 6-7

Rule 4: Every time a primary delay has been detected, a new call is made.

$$PDD_t = 1 \rightarrow \tau(t) = 0$$

Equation 6-8

For the example in Section 6.3.1, the delay propagation and measures have been computed in the G-PM and an output $\tau(t)$ is launched after dealing with all measures in the G-DL. The delay tolerant limit $\mathrm{d}_{\mathrm{limit}}$ is set to 300 seconds and the recovery time limit r_{limit} is set to 30 minutes. Figure 6-11 shows the G-DL output according to the decision logic rules stated above.

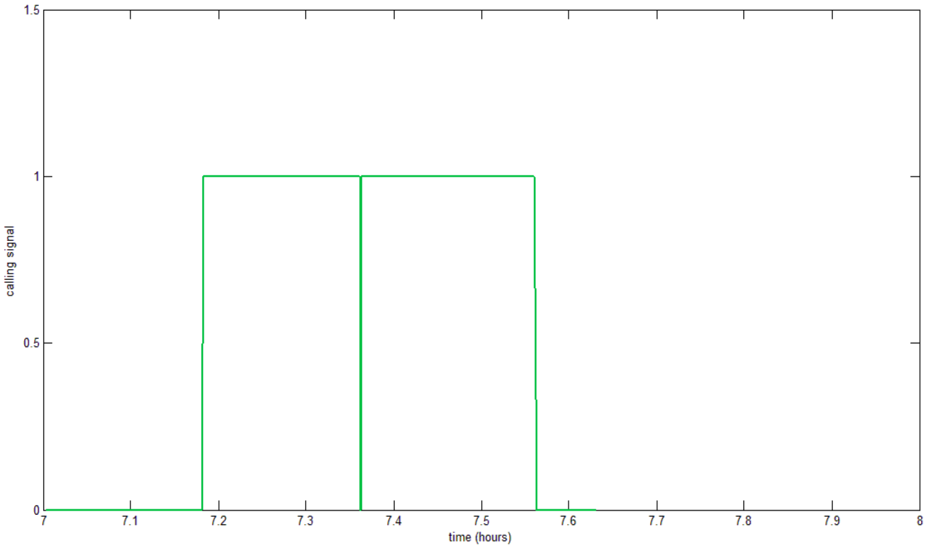


Figure 6-11: G-DL output for the example scenario

6.4.2 L-DL

The inputs for the local decision logic (L-DL) are the calling signal $\tau(t)$ from the G-DL, and JD_t from the L-PM. The local decision module aims to control the local rescheduling activities by giving the switching signal σ. When σ is set to “1”, the running process is interrupted and a new rescheduling command is given to the local area along with the current train movement information. When σ stays at “0”, the DC keeps the current timetable and sends trains to the junction based on the timetable order or the current rescheduled order. A new parameter is introduced into the L-DL, which is the rescheduling requirement (RR). Only when the RR has been satisfied, the order of rescheduling decision may be generated. The value helps to calculate the switching signal σ and it can be defined by analysing the calling signal $\tau(t)$ and the junction detection parameter JD_t. Rule 5 and rule 6 presented below give a demonstration of how to define the value of RR and the corresponding t, and rule 7 combines these three parameters together to obtain the switching signal σ.

Rule 5: When the rescheduling calling signal τ is received as 0, which means the rescheduling status has been initialised, each train, no matter it was rescheduled or not, is available to accept a new rescheduling command.

$$\tau(t) = 0 \rightarrow \ RR_{t\rightarrow\infty} = 1$$

Equation 6-9

Rule 6: After receiving a calling signal $\tau = 1$ from the G-DL, whenever a junction has detected a train reaching the edge of the junction area and this approaching train is available for rescheduling, the rescheduling command is made immediately, i.e., the switching signal σ is defined as 1.

$$\begin{cases} \tau(t) = 1 \\ JD_t = 1 \rightarrow \ \sigma(t) = 1 \\ RR_t = 1 \end{cases}$$

Equation 6-10

Rule 7: After receiving a switching signal $\sigma = 1$, a rescheduling command is made and all trains in the control region at the current time must follow the new schedule. These trains are not available for rescheduling when they reach the junction area since they have been rescheduled already. However, once a train which has not been rescheduled before reaching the junction area, the rescheduling status is initialised. All trains currently in the control zone are available for the next rescheduling command. ($t^{'}$ refers to the junction detection time of the next train which has not been rescheduled before).

$$\sigma(t) = 1 \rightarrow \ RR_{t+1\rightarrow t'-1} = 0$$

Equation 6-11

For the example scenario described in Section 6.3.1, the calling signal $\tau(t)$ (see Figure 6-11) and the junction detection signal JD_t (see Figure 6-10) are presented by the G-DL and the L-PM. According to the decision logic rules stated above, Figure 6-12 shows that the switching signal $\sigma(t)$ calculated by L-DL equals 1 at 07:16:42 ($t = 7.278$) and 07:22:27 ($t = 7.374$).

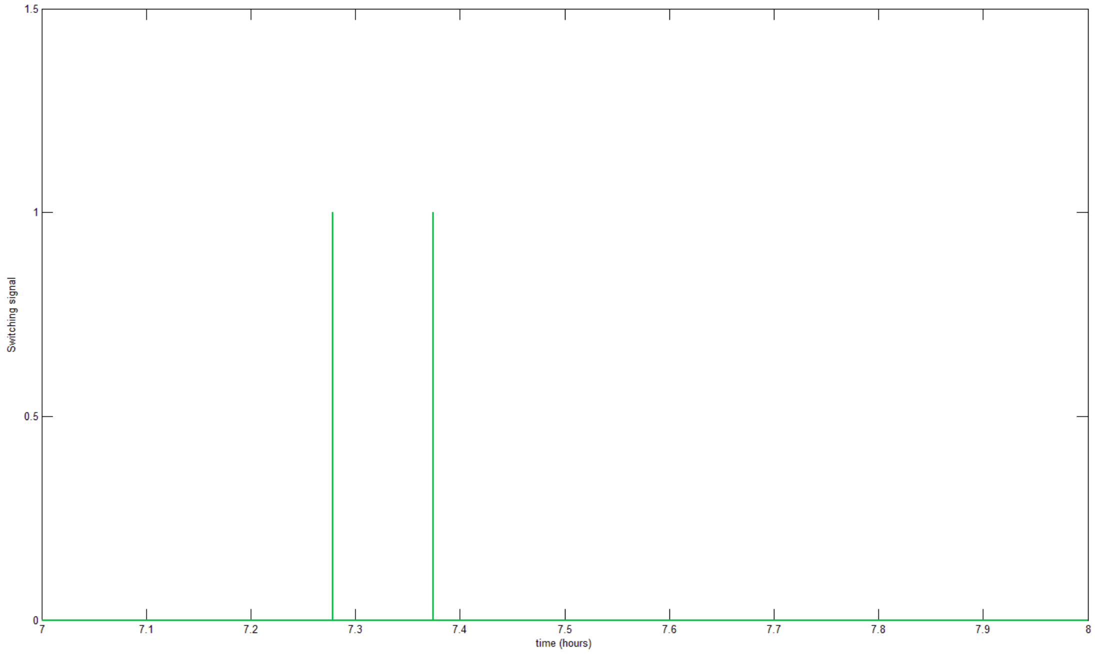


Figure 6-12: L-DL output for the example scenario

6.5 Testing progress

Supervisory control provides a way to combine more than one control strategy together to deal with dynamic situations (uncertain delays, mixed traffic patterns, dynamic traffic flows, etc.). An initial case study is prepared in the next section to realise performance-based supervisory control under a dynamic situation. The track layout is the same as the one in Section 5.1 and the nominal timetable has been slightly modified from that described in Section 5.3. In addition, the rescheduling process becomes more complex due to changes of the control strategy during operation. The testing progress of a simulation model with multiple control strategies applied to a single junction are explained in this section.

To analyse the performance of supervisory control on a single junction, a delayed scenario with dynamic information is tested with different control strategies applied. In addition, this scenario is tested by adopting supervisory control with alternating algorithms. The testing progress is illustrated in Figure 6-13. The upper part of this figure shows the rescheduling strategies applied on the single junction area, as well as the alternating algorithm obtained by applying supervisory control. The lower part of the figure gives the basic data of the railway network, which are infrastructure data, the nominal timetable and rolling stock characteristics. The delay scenarios are created and tested in the simulator and the results are visualised and evaluated in MATLAB.

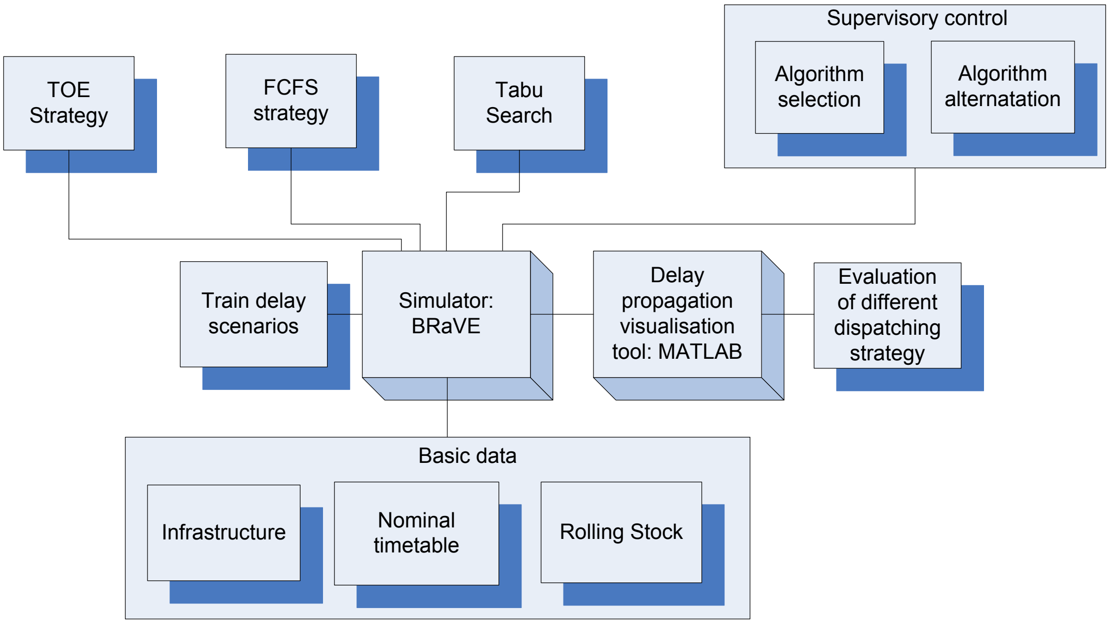


Figure 6-13: Testing process

The supervisory control for rescheduling management includes two parts: control strategy (algorithm) selection and algorithm alternation. The test progress for the supervisory control in the simulation follows Step 1 to Step 6 below:

Step 1: Run the nominal timetable with no advanced rescheduling strategy applied;

Step 2: Run a disturbed timetable with no advanced rescheduling strategy applied;

Step 3: Apply supervisory control to determine the algorithm changing time $t = [t_1, t_2, \dots t_n]$ in the junction area when switching signal $\sigma(t) = 1$;

Step 4: Determine the earliest algorithm changing time $t = t_1$, and select the rescheduling algorithm based on the algorithm recommendation provided by the local area;

Step 5: Re-run the disturbed timetable with rescheduling at $t = t_1$ in junction area.

Step 6: Repeat Step 3 to Step 5, until no more local rescheduling is needed.

6.6 Analysis of supervisory control on a single junction

6.6.1 Scenario development

Infrastructure and the working timetable

The network layout is the same as the one discussed in Section 5.2.1, with all 28 services coming from the same direction between 7:00am and 9:00am. In the first time period, trains A1 to A7 are presented. In the second time period the services are repeated, giving trains B1 to B7. In the third time period, trains C1 to C7 are present, and in the fourth time period, trains D1 to D7. Inspired by the real timetable published by Network Rail, an idealised model with high frequency services (services repeating every 10 minutes) is simulated in BRaVE. These services include two types of rolling stock: a fast train (vehicle type Class 313) and a slow train (vehicle type Class 59). The core junction area is located in Finsbury Park Station which is the same as the scenario described in Section 5.3.

Within the main rescheduling area, all services in the network are arranged to pass the area according to a conflict-free timetable. The timetable in Table 6-1 shows all 7 services in the time period with scheduled times: A1, A2 and A3, will stop at Brookmans Park Station and Alexandra Palace Station, then terminate at Kings Cross Station; A4 and A5 will stop at Hatfield Station and Kings Cross Station; A6 and A7 will stop at Bayford Station, Enfield Chase Station, and Palmers Green Station, and then terminate at Moorgate Station.

Timetable in Period 1							
Service ID/Train Name	**S1/A1**	**S2/A2**	**S7/A3**	**S9/A4**	**S10/A5**	**S8/A6**	**S11/A7**
Vehicle Type	Class 313	Class 313	Class 313	Class 59 loco	Class 59 loco	Class 313	Class 313
Start	(F, W)	(F, W)	(F, W)	(S, W)	(S, W)	(P, H)	(P, H)
Destination	(F, K)	(S, K)	(P, M)	(S, K)	(P, M)	(P, M)	(P, M)
Welwyn Garden City	07:00:00	07:02:00	07:04:00	07:00:00	07:02:00		
Hatfield	-	-	-	07:05:00	07:08:00		
Brookmans Park	07:07:00	07:09:00	07:12:00	-	-		
Hertfort						07:00:00	07:04:00
Bayford						07:05:00	07:09:00
Enfield Chase						07:13:00	07:17:00
Palmers Green						07:17:00	07:21:00
Alexandra Palace	07:17:00	07:19:00	07:22:00	-	-	-	-
Kings Cross	07:24:00	07:27:00		07:31:00			
Moorgate			07:33:00		07:38:00	07:31:00	07:35:00

Table 6-1: Original conflict-free timetable for one period

Scenario design

A delayed timetable is created to verify the supervisory control: Train B1 (the same service pattern as train A1 with an interval of 10 minutes) is delayed at Alexandra Palace Station for 8 minutes from 07:27:00 to 07:35:00.

- For this scenario, the delay on Train B1 influence the trains behind it on the same line (e.g. Train B2, B3, C1, C2, C3…), collisions may happen between delayed trains and scheduled trains at the Finsbury Park Junction. According to the testing progress, four tests are created with four different rescheduling strategies, which are TOE, FCFS, TS and alternating algorithms (AAs) obtained by supervisory control, are applied in this scenario.

To evaluate the results, the delay propagations of different trains are presented in G-PM. Considering the cost function used for the rescheduling is designed to minimise the total delay penalty, the total weighted delay is used to help visualise and compare the results. Delays are weighted according to the train type: the weight for the freight train (Class 59) is 0.2 and that for the passenger train (Class 313) is set to 1.

6.6.2 Results and analysis

Result of TOE:

The result of TOE is shown in Table 6-2. Figure 6-14 shows the weighted delay propagation obtained by TOE, which was to run the scenario with no advanced rescheduling strategy applied (TOE applied). The green line shows the total weighted delay for the whole railway network with 28 trains included. The six most delayed trains are also presented in the graph. Figure 6-

14 shows a very bad performance when TOE is applied because there is no rescheduling strategy used to help recover delay.

Rescheduling time	Junction area	Algorithm	Solution/Order of the trains
07:00:00	Finsbury Park Junction	TOE	Order = [A1, A2, A6, A3, A4, A7, A5, B1, B2, B6, B3, B4, B7, B5, C1, C2, C6, C3, C4, C7, C5, D1, D2, D6, D3, D4, D7, D5]

Table 6-2: Result of TOE

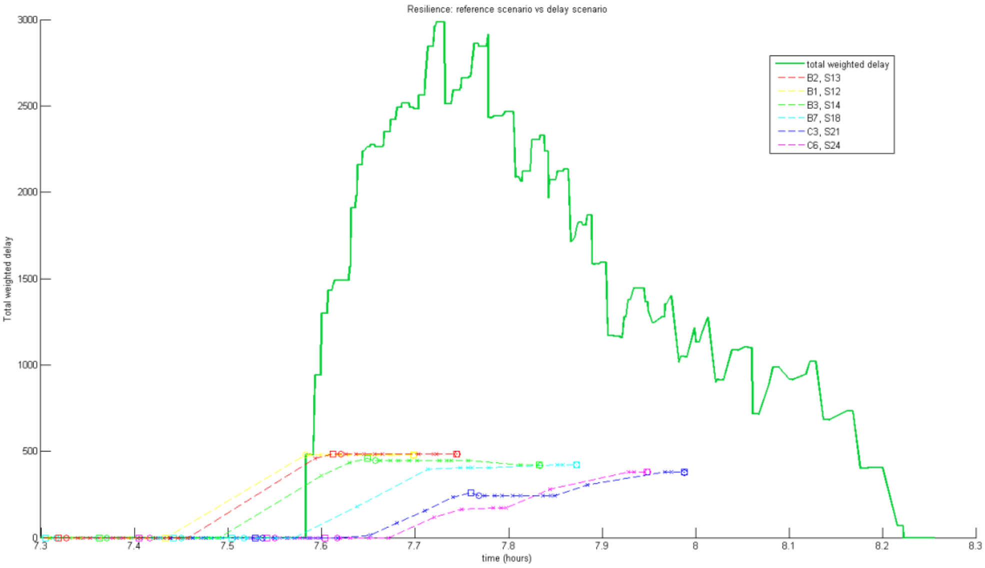


Figure 6-14: Weighted delay propagation when TOE is applied

Result of FCFS:

The result of FCFS is shown in Table 6-3. Figure 6-15 shows the weighted delay propagation obtained by FCFS, which runs the scenario with the FCFS strategy applied for the whole time (from 7:00:00). From the results we can see that the weighted delay is effectively restored by applying FCFS to the junction. The total weighted delay recovers more quickly than for other approaches. However, from time $t = 7.8\,h$ to $t = 7.9\,h$ and $t = 8.0\,h$ to $t = 8.1\,h$, there are significant influxes in the total weighted delay to be observed.

Rescheduling time	Junction area	Algorithm	Solution/Order of the trains
07:00:00	Finsbury Park Junction	FCFS	Order = [A1, A2, A6, A3, A4, A7, A5, B1, B4, B6, B7, B5, B2, B3, C6, C1, C2, C4, C7, C3, C5, D1, D6, D2, D3, D4, D7, D5]

Table 6-3: Result of FCFS

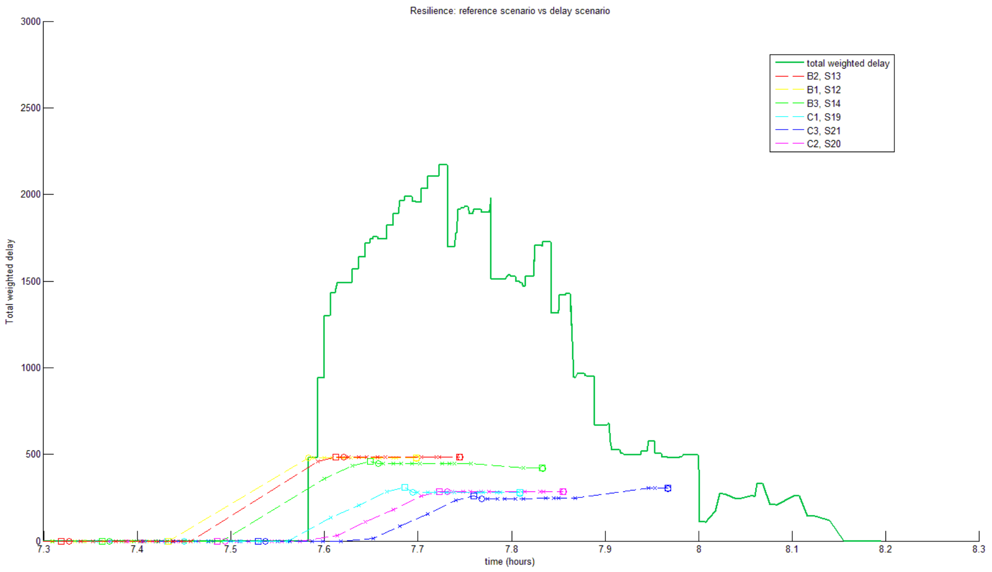


Figure 6-15: Weighted delay propagation when FCFS is applied

Results of TS:

The result of TS is shown in Table 6-4. Figure 6-16 shows the weighted delay propagation obtained by TS, which was to run the scenario with the TS strategy applied for whole time. The total weighted delay is observed to recover smoothly, albeit with a slight increase after $t = 8\ h$. That is because some trains must decelerate and accelerate a lot due to the trains in front of it are running late. The services are slower to recover than with FCFS.

Rescheduling time	Junction area	Algorithm	Solution/Order of the trains

07:00:00	Finsbury Park Junction	TS	Order = [A1, A2, A6, A3, A4, A7, A5, B4, B6, B7, B1, B5, B2, B3, C6, C4, C1, C2, C7, C3, C5, D1, D6, D2, D4, D3, D7, D5]

Table 6-4: Result of TS

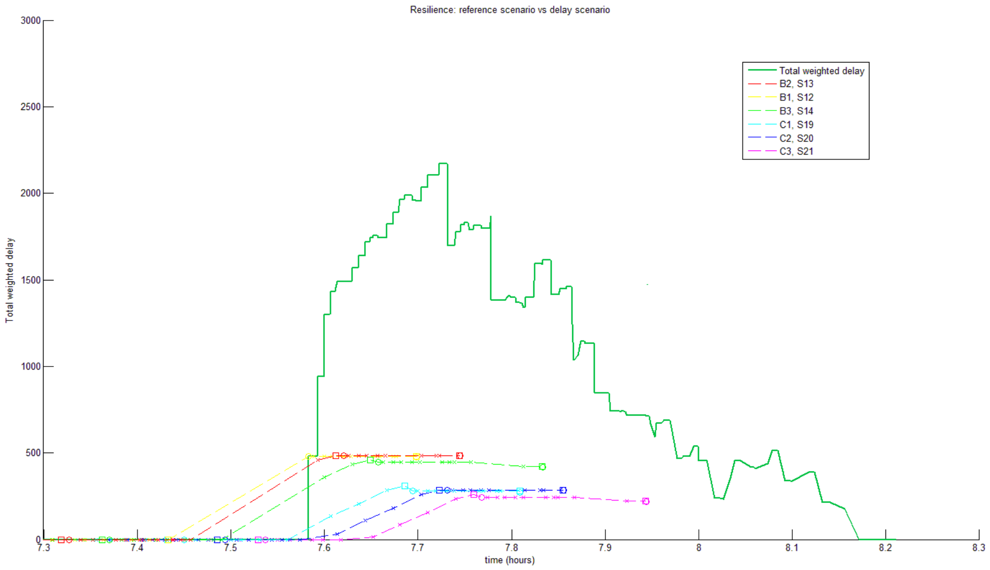


Figure 6-16: Weighted delay propagation when TS is applied

Results of AAs:

Supervisory control is applied in this scenario to achieve the rescheduling by alternating algorithms. In this experiment, d_{limit} is set to 300 seconds and r_{limit} is set to 30 minutes. From the $TD(t)$graph obtained by G-PM, $TD(t)$ is first observed to increase to 480 seconds at $t = 7.58\ h$ (07:35:00). However, in reality, delay starts accumulating by the time it happens. Therefore, when calculating Equation 6-4, $t \geq 7.5\ h$ (07:30:00) is acceptable (the primary delay on B1 starts at 07:27:00, and the secondary delay on train B2 starts at 07:28:00).

Switching signal $\sigma(\mathrm{t})$ is calculated to equal 1 for the first time at $t = 7.53\ h\ (07{:}31{:}49\ )$ from Equation 6-5 to Equation 6-11. According to the algorithm recommendation provided by the local area, TS is chosen for the local rescheduling. Re-running the delayed timetable with TS applied at $07:31:49$ and apply supervisory control to the new traffic flows, the switching signal $\sigma(\mathrm{t})$ is calculated to equal 1 at $t = 7.843\ h$ ($07:50:35$). According to the algorithm recommendation provided by the local area, FCFS is chosen for the local rescheduling. Re-running the delayed timetable with TS applied at $07:31:49$ and FCFS applied at $07:50:35$ and applying supervisory control to the new traffic flows, no more rescheduling is needed. Finally, the alternating algorithm TS-FCFS is obtained by the supervisory control. The result of AAs is shown in Table 6-5.

Figure 6-17 shows the weighted delay propagation obtained by AAs, which was to apply supervisory control in the junction area alternating algorithm between TS and FCFS. In Figure 6-17, the total weighted delay is recovered smoothly and quickly. It takes slightly longer for total recovery than when applying FCFS for the whole time, but no significant increases in weighted delay are observed in the recovery from $t = 7.8\ h$ to $t = 7.9\ h$.

Rescheduling time	**Junction area**	**Algorithm**	**Order of the trains**
07:00:00	Finsbury Park Junction	TOE	Order = [A1, A2, A6, A3, A4, A7, A5]
07:31:49	Finsbury Park Junction	TS	Order = [B4, B6, B7, B1, B5, B2, B3, C6, C1, C4, C2, C7, C3, C5,]
07:50:35	Finsbury Park Junction	FCFS	Order = [D1, D6, D2, D3, D7, D4, D5]

Table 6-5: Result of AAs

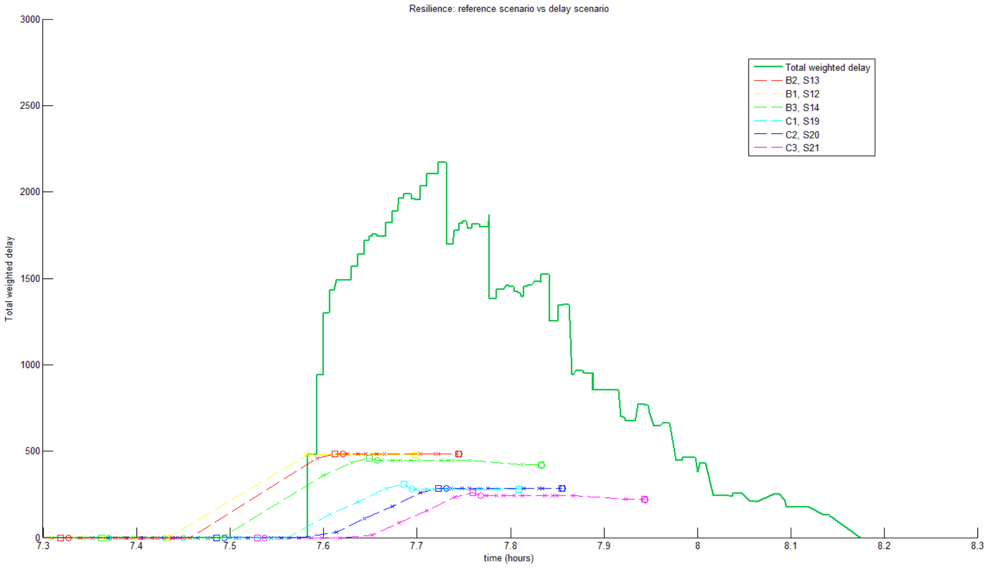


Figure 6-17: Weighted delay propagation when TOE-TS-FCFS is applied

6.6.3 Summary

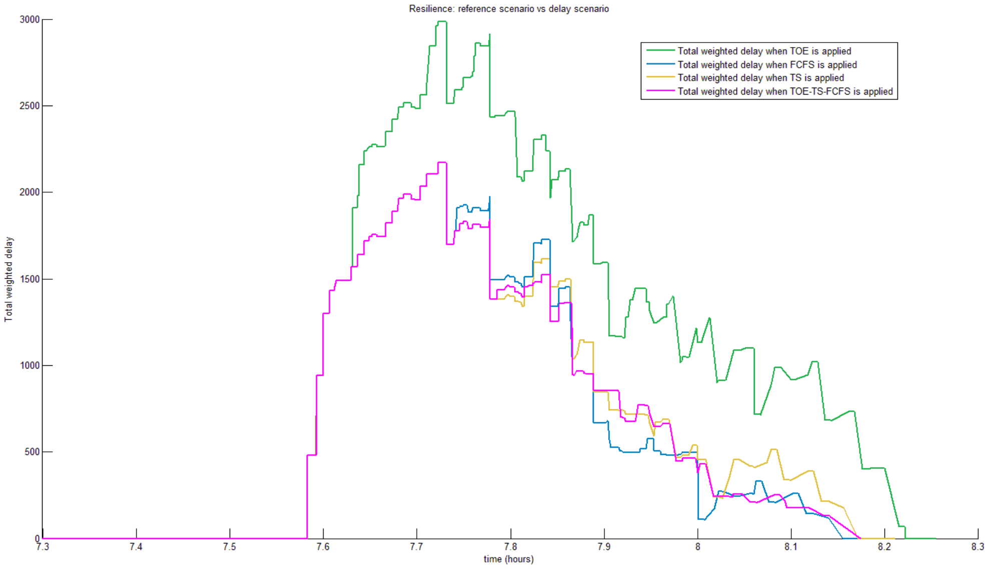


Figure 6-18: A comparison of total weighted delays when different strategies applied

Figure 6-18 gives a direct view on the instantaneous total weighted delays when different strategies applied to the same scenario. The green line (TOE is applied) shows the worst recovery on the delay propagation since no reordering strategy has applied, and the delayed trains (e.g. B2, B3, C2, C3) block the tracks for a long time prevent other trains from passing. The blue curve (FCFS is applied) shows a higher lateness between $t = 7.75$ to $t = 7.85$ because C2 is scheduled before C4. However, the pink curve (when TOE-TS-FCFS is applied) and the yellow curve (TS is applied) have a better recovery in this time period because C2 is scheduled after C4. The blue curve and the yellow curve both suffer a rebound after $t = 8$ due to the lateness on D5 is accumulated because of some frequent accelerate and decelerate movements after D7. On the contrary, the pink curve is stably restored since D5 can run smoothly after D4.

From each of the graphs shown, the results from the cost function (i.e. total delay penalty) and three KPIs of total weighted delay curves – maximum weighted lateness, time to recover and integral of delay – and the proportion of the maximum possible integral of delay value are collected and summarised in Table 6-6.

Rescheduling strategy	**Total delay penalty (£)**	**Max weighted Lateness (s)**	**Time to recover (s)**	**Integral Delay (s^2) (normalised value (%))**	**Integral Delay Proportion (%)**
TOE	535.82	2987	2297	35488 (100)	51.723
FCFS	297.58	2097	2003	21351 (60.16)	50.832
TS	301.72	2097	2061	22065 (62.18)	51.054
AAs	295.3	2097	2009	20732(58.42)	49.211

Table 6-6: Comparison information of different strategies

From the table shown above, TOE takes the biggest total delay penalty, and AAs takes the smallest value. The differences between FCFS, TS and AAs are quite small. FCFS, TS and AAs have the same max weighted lateness, and FCFS takes the shortest time to recover in this scenario. AAs takes a slightly longer time to recover than FCFS in this scenario. However, AAs has the smallest integral delay, the smallest integral delay proportion in this case, which reflects the delay occupation and how severe the delays are relative to the maximum weighted lateness throughout the period of the time window.

6.7 Conclusion

It has been shown that supervisory control has the potential to provide a feasible means to solve problems with dynamic information by switching the control strategies. A general introduction to the supervisory control is given and the different types of supervision were demonstrated in the first part of this chapter.

A performance-based supervisory control on railway rescheduling management has been developed to solve the rescheduling problem by combining different algorithms together with a specified sequence and time of changing. The BRaVE simulator and a specially developed visualisation tool are used to visualise and assess the results.

BRaVE can output log files compatible with the visualisation tools (programmed with MATLAB). Stations and signals can be selected from a menu and set as timing points. In addition to timing points, the arrival and departure times at stations and passing times at selected signals can be recorded for all trains. From this information, details of the propagation of delays in the system can be derived by the visualisation method mentioned in Chapter 3.

The case study in this section evaluates the performance of applying different algorithms alone, as well as the alternating algorithms provided by supervisory control. It verifies the benefits in using more than one algorithm in solving complex, dynamic problems.

CHAPTER 7

STUDY OF MULTI-JUNCTION RESCHEDULING IN A LARGE SCALE RAILWAY NETWORK

Railway dispatching and scheduling are conventionally modelled using classical technologies, such as constraint programming [122] and logic programming [123]. These approaches are good at modelling static situations where all resources are known in advance, and constraints are known (e.g. train timetabling problem (TTP)). However, they lack the ability to handle dynamic situations with partial information. Additionally, as the size of the area to be considered and the number of trains increases, it becomes more difficult to implement effective centralised control strategies that are able to satisfy the problem of real-time traffic management. These issues have led researchers to consider new distributed approaches. Often a distributed artificial intelligence (DAI) system is used to approach this problem with multiple distributed “problem solvers” [124] .

DAI, in general, is a method to solve complex decision making problems. It usually requires the distribution of an intelligent process among independent entities, thus it is able to accomplish large scale computation resources [125]. DAI systems consist of distributed processing nodes (also called agents) at a large scale. Therefore, DAI systems do not require all the relevant information to be considered and coupled at one time; in contrary, these nodes can act independently and partial solutions are generated by communication between nodes, often asynchronously. What’s more, the source may change or be updated during the execution process.

The main benefit of adopting a DAI system in the railway network rescheduling problem is that it allows the local, independent intelligent systems to collaborate and interact with each other to perform traffic management based on a real-time condition and it potentially removes the upper limit of the maximum railway network size. The challenges arising with distributed processing are discussed in the next section and a blackboard architecture for constructing DAI systems are presented.

In this chapter, a DAI model consisting of multiple junctions is constructed to solve train rescheduling problems on a large scale railway network. A detailed description of each component of the model is presented. A case study of multi-junction rescheduling is presented to demonstrate the efficiency of the DAI model in solving train rescheduling problems in a large scale railway network.

7.1 Problem statement

Generally, the real-time rescheduling problem can be defined as follows: given a railway network, the position and speed of each train entering the control region is known by the dispatchers, and decisions covering a set of train routes and passing times at each relevant crossing point in the network must be made when a delay is detected that may result in a conflict with the optimisation objectives. The train rescheduling problem can be categorized into static and dynamic rescheduling. For static rescheduling, the entire information of the whole network and all rolling stock is known before solving the problem. This information includes train arrival times, train lengths, train speed, etc. For dynamic rescheduling, the dispatchers only know the information of the trains at the time they enter the network. The schedule of the newly entered train is based on the information of the trains currently in the network. All the information of later arriving trains is unknown [126].

In previous chapters, it has already been shown how rescheduling strategies at junctions influence delay recoveries by visualising the delay propagation of the system. Junctions, as vital components of the railway infrastructure, need to be intelligently controlled in delayed scenarios. The main purpose of this chapter is to define a junction control system (control rules and module architecture) which can be applied to a large network with dynamic traffic flows.

Compared with the centralised railway junction control, in order to consider all junctions in the network at one time, a distributed railway junction control system has to achieve dynamic real-time rescheduling through one or several separated junctions working cooperatively; these junctions are treated as decision centres (DCs) in a distributed junction control system.

One main problem of utilising distributed processing is how to guarantee that decisions taken locally with incomplete information contribute to the global goal. Various approaches are possible for global control including [125]:

- Do nothing;
- Implement hierarchical control;
- Use optimisation techniques;
- Use a market approach;
- Use a collaborative approach.

Doing nothing means letting each agent make their own decision corresponding to sub-problems and return the results to the calling agent. Hierarchical control is used in a 'blackboard architecture' for controlling the sequencing of each agent. Optimisation techniques such as the genetic algorithm or taboo search are used to find additional solutions to a global problem. The 'market approach' uses a monetary value to balance supply and demand. In a collaborative approach, agents are continually exchanging information, and collaborate with global goals.

7.2 Blackboard architecture of DAI systems

The general models of DAI systems include blackboard architecture, contract net, and actors. The blackboard architecture is proposed as an approach to organize and operate a large AI system and is used in this thesis to providing hierarchical control to each processing node. It consists of three major components (see Figure 7-1) [127]:

- The specialist modules, which are called knowledge sources (KSs), can be considered as human experts in the real world, each one provides specific expertise to the application.
- The blackboard is a shared storage of problems, solutions and information. The blackboard can be considered as a dynamic "warehouse" of all possible solutions to the current problem which are provided by different KSs.
- In the blackboard architecture, the control shell is constructed to control the problem-solving process in the system. These KSs need an operating system to organise them in the most effective manner, and this is provided by the control shell.

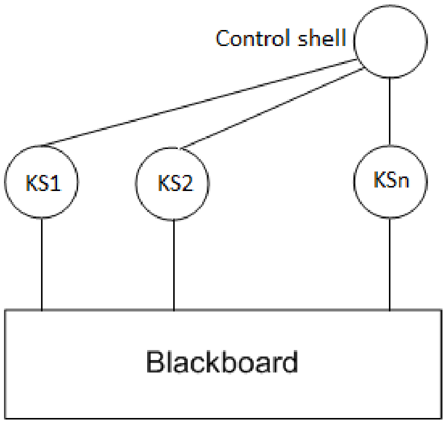


Figure 7-1: Blackboard architecture [127]

The main advantages of the blackboard architecture are: (1) separate, independent knowledge sources; (2) shared memory between each KSs; and (3) the possibilities of parallel processing. The shared memory contains all the data, variables and information used to solve the problem at hand. Gathering knowledge into separate, independent knowledge sources allows for a simple form of parallel processing, which may increase the speed and efficiency of the overall process [128]. By using this model, the structure becomes parallel and asynchronous.

7.3 An architecture for network rescheduling

In this section, a DAI system architecture is developed for railway rescheduling in a large scale operational network. The structure represented in Figure 7-2 is based on a traditional blackboard model with individual decision centres (DCs). One decision centre (DC) is a local control centre which can make decisions based on the local information. It is formed of a sensor module, an intelligent local rescheduling system module, a decision execution system module and a supervisor.

DCs gather information from the whole railway network database through a sensor interface. After an intelligent local rescheduling process, a rescheduled timetable is calculated and returned back to the railway network. These DCs are distributed but they share the same railway network database. Once a new timetable is delivered to the network, the network database is updated so that the neighbouring agents can react to the new condition. A global control centre is built to control different DCs to gather a reliable and feasible global solution.

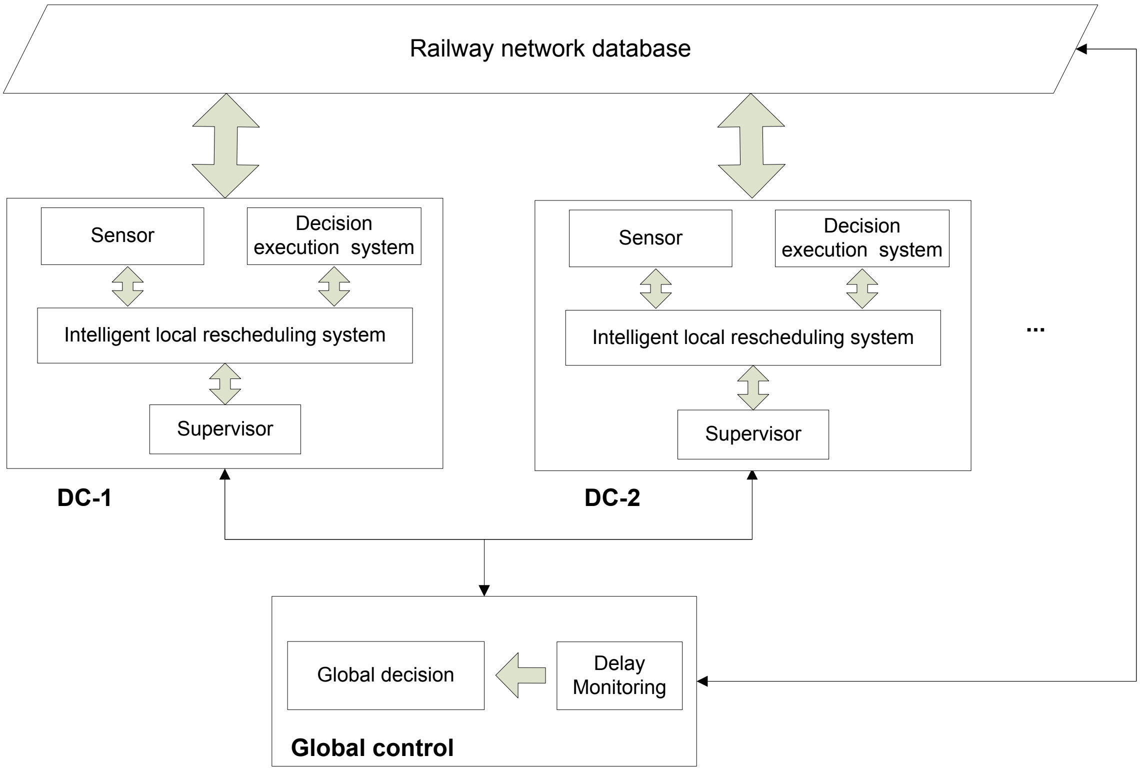

Railway network database
Sensor
Decision execution system
Intelligent local rescheduling system
Supervisor
DC-1
Sensor
Decision execution system
Intelligent local rescheduling system
Supervisor
DC-2
...
Global decision
Delay Monitoring
Global control

Figure 7-2: Architecture of railway network rescheduling

7.3.1 Network database

The railway network database consists of three main elements: users (trains), resources (infrastructure) and management constraints from the control centre [129].

1) **User:** the information about users (trains) includes static characteristics and dynamical information about their state. The static quantities include the train identifier, train class, train running timetable and vehicle characteristics (e.g. max

speed, max acceleration). The dynamic quantities give the information about the train's current position, speed and delay.

2) **Resources:** the network resources are what users need to fulfil their scheduled plan. Here, the track identifier, length and maximum line speed are presented as the resources' static data. States (free/occupied/prepared) and signal states (red/green) are regarded as dynamic quantities.

3) **Management constraints:** the traffic management constraints lead rescheduling results when a delay is detected by the control centre. Some of these constraints are static like minimum headway, minimum dwell time, and train delay penalty, which are given by the control centre; some of them are dynamic, such as the current speed of the train. For example, when a delay has occurred, the driver is allowed to increase the speed of the train to recover the delay along the journey.

7.3.2 Decision centre (DC)

DCs are the sites in the network where the control action on the user's state can be performed. Generally, DCs are the stations, switches and points. In this work, DCs are regarded as core junction areas and rescheduling decisions can be performed on all approaching users in the control zone. The size of a control zone depends on the capability of a DC, and one user is considered by one DC at one time. Thus, the boundary is usually situated at a specific distance (for example 10 km) from the core junction area, or at the edge of neighbouring DCs.

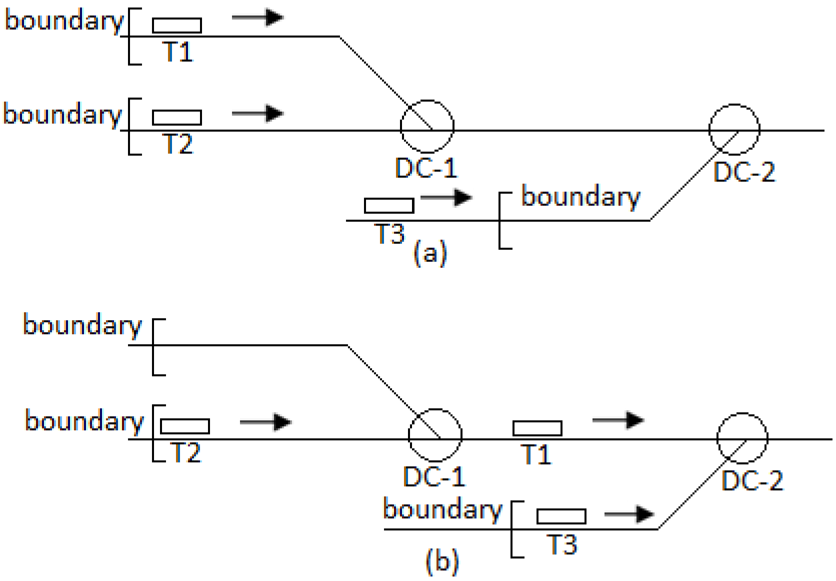


Figure 7-3: Control boundary description

For example, in the upper graph of Figure 7-3, at time $t = t_1$, train T1 and T2 are approaching decision centre DC-1 inside of the boundary of the control zone and train T3 is approaching decision centre DC-2 outside of the boundary of DC-2. DC-1 calculates and provides a rescheduling solution for train T1 and T2, for example, T1 is scheduled to pass the junction area before T2. At this time, DC-2 has no train to schedule. In the lower graph of Figure 7-3, at time $t = t_2$, T1 has exited DC-1 and is approaching DC-2, T3 has entered the control zone of DC-2 and is approaching DC-2. T1 and T3 will be considered by DC-2 and a rescheduled plan will be performed on them. At this time, DC-1 is still executing the previous rescheduling plan.

A DC consists of four modules: sensor, intelligent local rescheduling system, decision execution system and supervisor. DCs do not communicate with each other directly, but a supervisory control is given to each DC to manage coordination and cooperation.

1) Sensor:

The sensor module is in charge of gathering information from the network database to the local DC. To make a local rescheduling decision, all real-time data in the control zone should be transmitted to the DC. The management constraints from the DC are also gathered and stored by the sensor. It is a local database which can filter and store the data in real-time.

2) Intelligent local rescheduling system:

An intelligent local rescheduling system module is the core module of the DC. It requires the following capability for problem solving: i) a dynamic planning ability which can adjust the real-time situation; ii) a fast response to the real-time data when called; iii) an ability to learn what improves its performance as more problem solving experiences are obtained. Proenca and Oliveira have previously proposed an adaptable architecture for railway traffic control in a communication based train control system which consists of two sub-systems: “Control” and “Learning” [130]. The “Control” module is responsible for traffic management and the “Learning” module has the objective of analysing the previous information, identifying cases and giving control rules. To satisfy all requirements of an intelligent local rescheduling system, it contains a learning process with its control process. A flow chart that indicates how a rescheduling system operates is presented in Figure 7-4. During the network operation, a rescheduling command will be sent to the local rescheduling system from the supervisor. When this order is received, the data collected from the sensor will be passed to the rescheduling system, which is dealt with by an advanced algorithm. The learning module requests the control registry to record, evaluate and select an algorithm, which requires the identification of the problem and a recommendation for the most efficient methods for the optimisation. Generally, the objectives for local rescheduling are defined by the local DC.

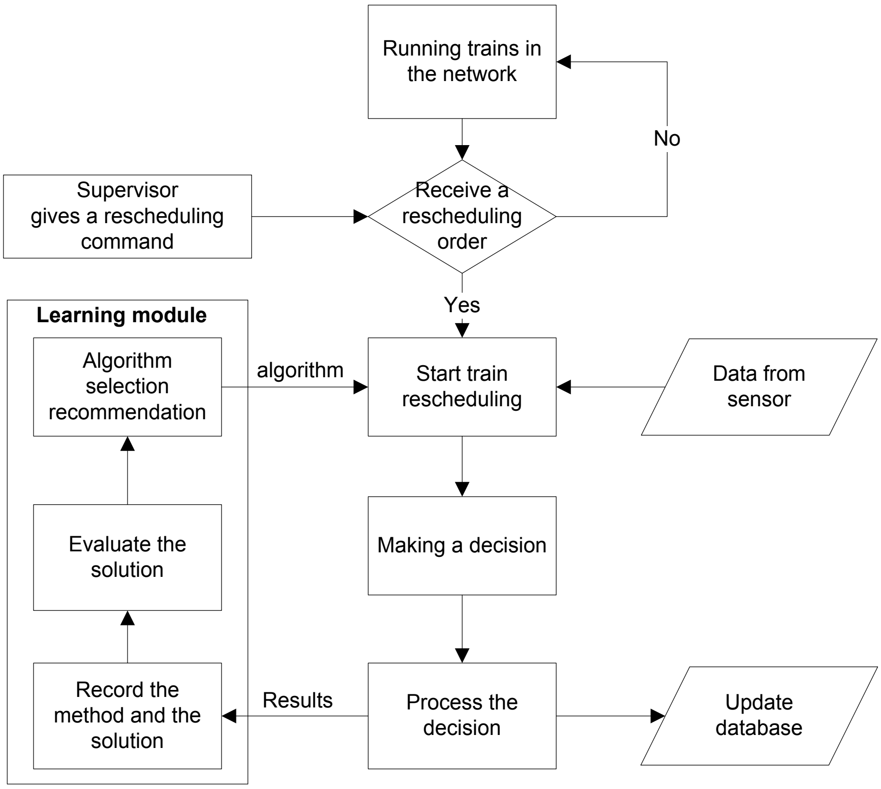

Running trains in the network
No
Supervisor gives a rescheduling command
Receive a rescheduling order
Yes
Learning module
Algorithm selection recommendation
algorithm
Start train rescheduling
Data from sensor
Evaluate the solution
Making a decision
Record the method and the solution
Results
Process the decision
Update database

Figure 7-4: Intelligent Local Rescheduling System

3) Decision execution system:

After the rescheduling progress, a rescheduled plan for all existing trains in the local control zone will be sent to the decision execution system. This plan is regarded as the best current plan and is processed immediately. The local rescheduling system and decision execution system cannot work synchronously; once a new call is received, the decision execution system is interrupted and waits for a new rescheduling plan.

4) Supervisor:

The supervisor is an important part which links the global control to the local control, the supervisor is in charge of giving orders to the local rescheduling centre. The order is simply ‘go’ or ‘no go’ (‘1’ or ‘0’) based on the global information and the local information at the junctions. Performance-based supervisory control is used to provide rescheduling orders to the local rescheduling system. A detailed explanation can be found in Section 6.2.

7.3.3 Global control

Due to the complexity and the scale of the system, the network rescheduling problem has been divided into several individual local rescheduling problems. Here, a global control, commonly referred to as supervisory control, is introduced to coordinate all DCs by controlling different supervisors.

Delay propagation is monitored by the global control centre and adjustments to the control are made based on the observed performance. A detailed explanation was provided in Section 6.2, and also in Section 6.2, a mathematic model of supervisory control is built, and an illustrative case study is presented in Chapter 6 to demonstrate how it operates.

7.4 Case study of multi-junction rescheduling on the southern part of ECML

7.4.1 Introduction to the control regions

This multi-junction case study is an extension of the single-junction case study introduced in Chapter 5 and 6. A southern part of the ECML, which is a major inter-city railway with mixed services, was used as the basis of the model, and shown in Figure 7-5. The entire network has been allocated to three DCs and each DC indicates a junction that the rescheduling strategies are applied on. DC-1 is located near Finsbury Park Station and consists of two adjacent flat junctions. Considering the capability of DC-1, its control zone includes the lines which come from Welwyn Garden City Station and Hertford Station to the core junction area, and the boundaries of the control zone addressed at Welwyn Garden City Station and Hertford Station are approximately 28 km from the core junction area. DC-2 is a bi-direction junction which is located near Belle Isle Station. The control zone considers the trains which come from the north after they exit the junction area of DC-1 (about 2.5 km from the junction of DC-2), the trains that come from King's Cross Station in the south (about 1.2 km from the junction of DC-2), and trains that enter the network from the direction of the Midland Road Junction (about 6.4 km from the junction of DC-2). DC-3 is another flat junction near Finsbury Park Station and it considers the trains that come from the south after they exit from DC-2 (about 2 km from the junction of DC-3), and trains that come from Moorgate Station in the south (about 2 km from the junction of DC-3). See Figure 7-5.

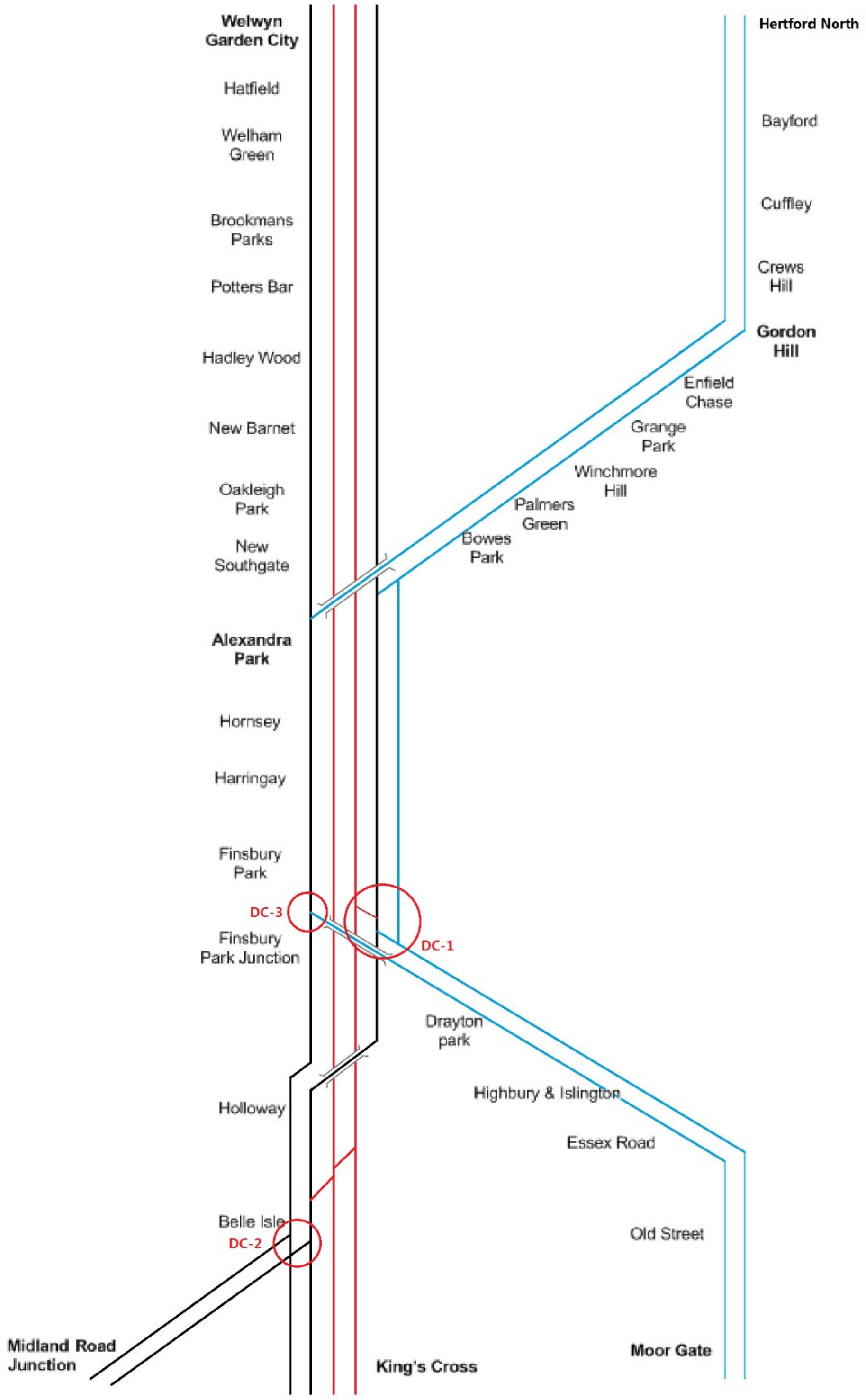


Figure 7-5: Control region and DCs specification

7.4.2 Introduction to the vehicles and nominal timetables

The modelling is carried out with a simplified timetable with 37 services running from 7:00 am to 9:00am based on a timetable published by Network Rail [92] (see Appendix C). These services include four types of vehicles, which are Class 91, Class 317, Class 313 and TL39. The characteristics of each type are presented in Table 7-1, including the maximum speed of the train, the maximum braking and acceleration rates, and the train-specific delay penalties. All this information is used to simulate train running and calculate the cost function in the following sections.

Type	**Class 91**	**Class 317**	**Class 313**	**TL39**
Maximum speed (km/h)	201	160	121	160
Maximum braking rate (ms^{-2})	0.67	0.78	0.78	0.78
Max acceleration (ms^{-2})	0.588	0.588	0.588	1
Total mass (tonnes)	394	264	220	332
Train length (m)	253.9	158.6	118	161.99
Number of seats	400	400	231	231
Penalty ($£s^{-1}$)	0.5	0.2	0.1	0.1

Table 7-1: Vehicles parameters

7.4.3 Introduction to the delay scenarios

Scenario 1: Train E1 is delayed at Potters Bar Station for 930 seconds and at Finsbury Park for 300 seconds.

- For this scenario, one train is delayed before it passes through DC-1, and different rescheduling decisions could cause different delay scenarios in the other DCs.

Scenario 2: Train E1 is delayed at Potters Bar Station for 930 seconds and at Finsbury Park for 300 seconds. Train J2 is delayed at Old Street Station for 510 seconds.

- For this scenario, the delays of Train E1 and Train J2 result in knock-on delays at two DCs; different rescheduling decisions could lead to a different delay recovery situation.

Three tests are created with three different rescheduling strategies, which are TOE, FCFS and alternating algorithms (AAs) obtained by supervisory control, applied to each scenario. To evaluate the results, the delay propagations of different trains are presented in G-PM. Considering the cost function to minimise the total delay penalty, the total weighted delay is used to help visualise and compare the results. Delays are weighted according to the train type: the weight for the passenger trains (vehicle types are Class 313 and TL39) is 1, for the premium passenger trains (vehicle types are Class 317) it is 2, and for high speed trains (vehicle types are Class 91) it is set to 5.

7.4.4 Results and analysis

The results of scenario 1

TOE, FCFS and AAs obtained by supervisory control are applied to the scenario 1. Table 7-2, Table 7-3 and Table 7-4 show the results obtained by different rescheduling approaches, including the times to active rescheduling, places of rescheduling, rescheduling approaches and the planned order in which the trains pass the junction area. Here, the order planned by the algorithm is different from the order has been implemented, and trains is following the planned order until a new reschedule plan is received. Figure 7-6, Figure 7-7 and Figure 7-8 show the total weighted delay recovery graphs obtained by the different approaches. The green line shows the total weighted delay for the whole railway network with 37 trains included. The ten most delayed trains are also presented in the graph.

Rescheduling time	Junction area	Algorithm	Order of the trains
07:00:00	DC-1	TOE	Order = [S11, S16, S12, S15, S13, S24, S14, S27, S21, S28, S22, S25, S37, S26, S39, S23, S40, S35, S36, S38]
07:00:00	DC-2	TOE	Order = [S17, S20, S15, S33, S30, S14, S28, S41, S42, S26, S40, S38]
07:00:00	DC-3	TOE	Order = [S17, S20, S18, S19, S34, S33, S30, S31, S32, S45, S41, S42, S44, S46]

Table 7-2: The rescheduling results of scenario 1 of TOE

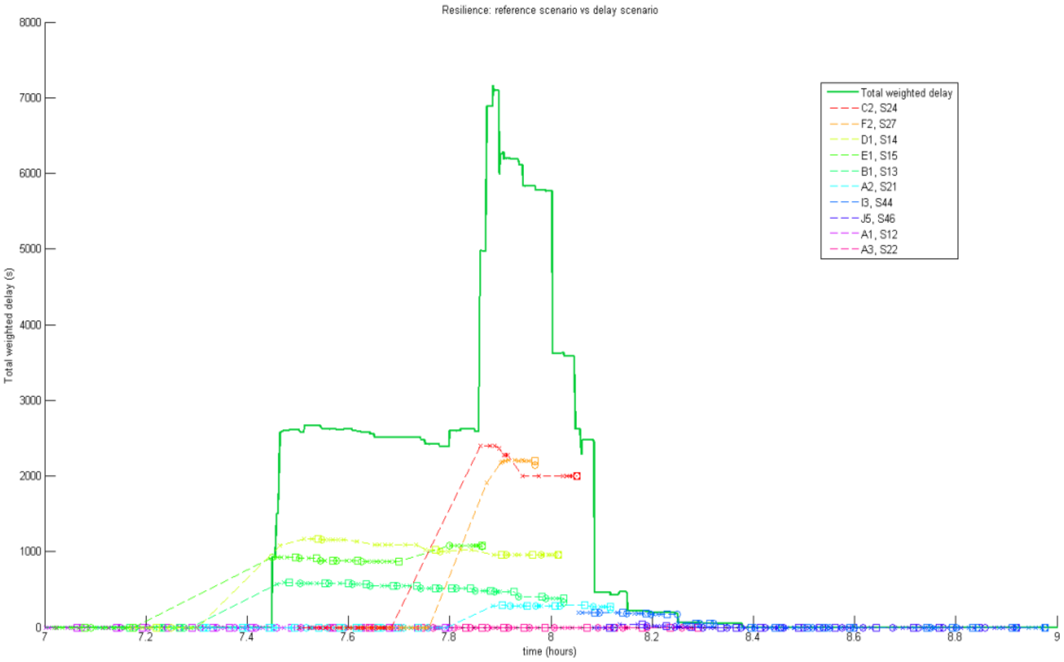


Figure 7-6: the delay propagation graph of scenario 1 of TOE

Rescheduling time	**Junction area**	**Algorithm**	**Order of the trains**
07:00:00	DC-1	FCFS	Order = [S11, S16, S12, S15, S24, S21, S13, S27, S14, S28, S22, S25, S37, S26, S39, S23, S40, S35, S36, S38]
07:00:00	DC-2	FCFS	Order = [S17, S20, S33, S30, S15, S14, S28, S41, S42, S26, S40, S38]
07:00:00	DC-3	FCFS	Order = [S17, S20, S18, S19, S34, S33, S30, S31, S32, S45, S41, S42, S44, S46]

Table 7-3: The rescheduling results of scenario 1 of FCFS

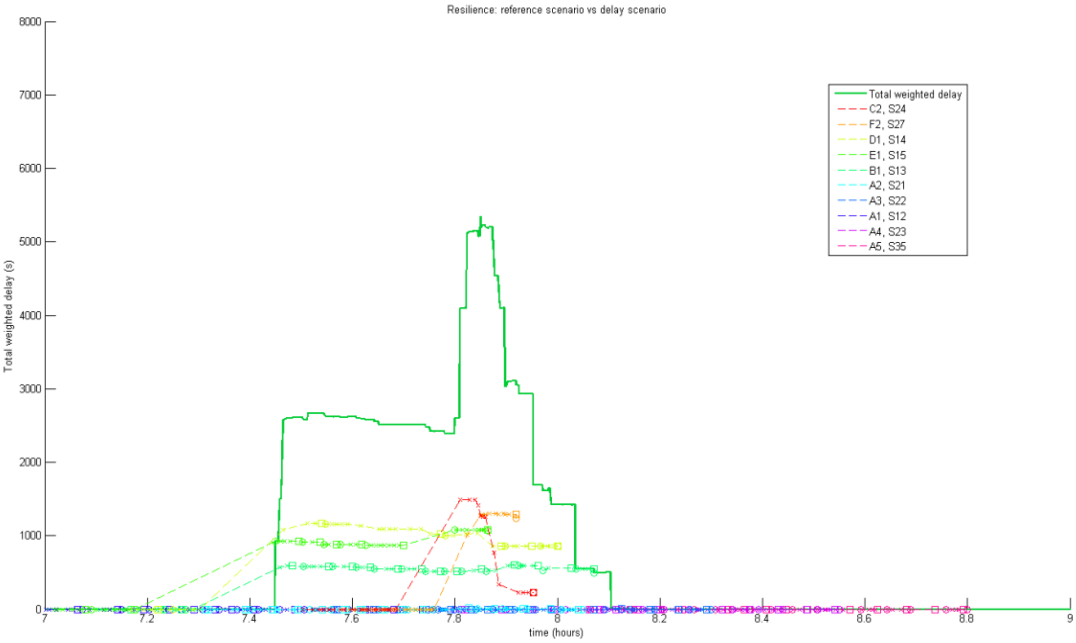


Figure 7-7: the delay propagation graph of scenario 1 of FCFS

Rescheduling time	**Junction area**	**Algorithm**	**Order of the trains**
07:00:00	DC-1	TOE	Order = [S11, S16]
07:00:00	DC-2	TOE	Order = [S17, S20, S33]
07:00:00	DC-3	TOE	Order = [S17, S20, S18, S19, S34]

07:33:00	DC-1	BF	Order = [S12, S24, S15, S13, S21, S14, S22, S28]
07:33:53	DC-2	TOE	Order = [30]
07:35:35	DC-3	TOE	Order = [S33, S30, S31, S32]
07:46:08	DC-1	TS	Order = [S27, S13, S21, S14, S22, S28, S25, S23, S26]
07:50:46	DC-2	TOE	Order = [S15, S14]
07:56:27	DC-1	TOE	Order = [S28, S22, S25, S26, S23]
08:00:56	DC-3	TOE	Order = [S45]
08:01:52	DC-2	TOE	Order = [S28, S41]
08:03:53	DC-2	TOE	Order = [S42]
08:05:35	DC-3	TOE	Order = [S41, S42, S44, S46]
08:11:08	DC-1	TOE	Order = [S37, S26, S39, S23, S40, S35, S36, S38]
08:18:49	DC-2	TOE	Order = [S26, S40, S38]

Table 7-4: The rescheduling results of scenario 1 of AAs

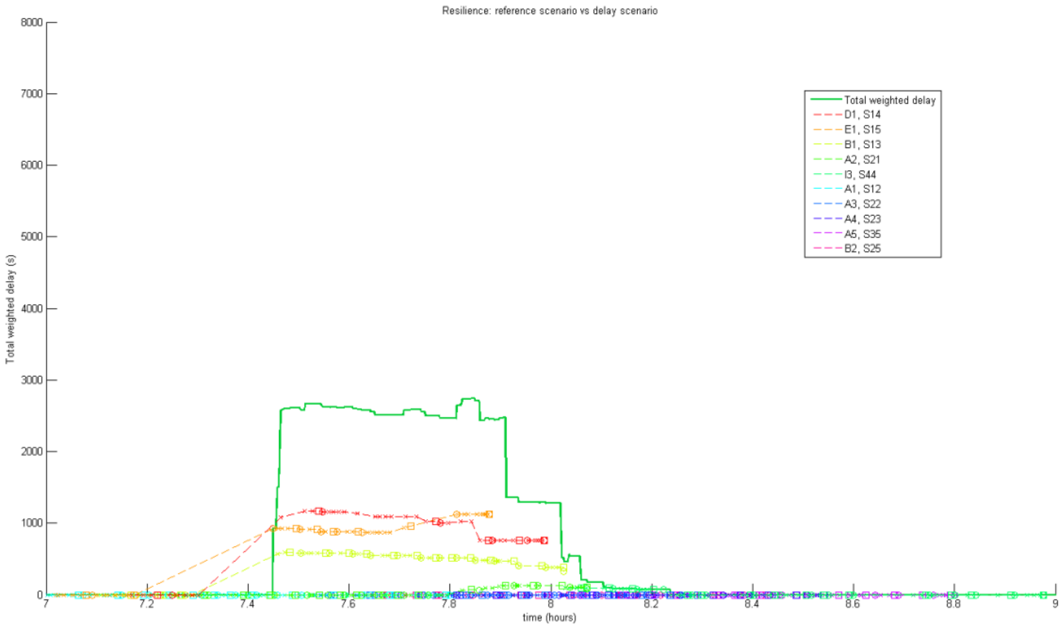


Figure 7-8: the delay propagation graph of scenario 1 of AAs

In Table 7-4, the application of TOE applies at each time only to the order of services have entered the control area, which means, the trains out of the control area are deleted from the original TOE order. Figure 7-6 shows a very bad performance when TOE is applied to each DC because there is no rescheduling strategy used to help recover the delay. From Figure 7-7 we can see that the weighted delay is restored by applying FCFS to the junctions. However, there is a significant influx in the total weighted delay at time $t = 7.8\ h$ because of the fact that the disruption on E1 at Finsbury Park Station delays the trains behind it. Figure 7-8 shows the weighted delay propagation obtained by AAs. The total weighted delay is observed to last for some time and then to recover quickly. It takes longer to recover than when applying FCFS, but no significant increase in weighted delay is observed during the recovery.

The results of Scenario 2:

In this section, TOE, FCFS and AAs obtained by supervisory control are applied to scenario 2. Table 7-5 and Table 7-6 show the results obtained by TOE and FCFS respectively, Table 7-7 shows the results obtained by AAs applied on DC-1, and TOE applied on DC-2 and DC-3. Table 7-8 presents the results by applying AAs on all DCs. Figure 7-9, Figure 7-10, Figure 7-11 and Figure 7-12 show the total weighted delay propagation graphs obtained by different approaches.

Rescheduling time	**Junction area**	**Algorithm**	**Order of the trains**
07:00:00	DC-1	TOE	Order = [S11, S16, S12, S15, S13, S24, S14, S27, S21, S28, S22, S25, S37, S26, S39, S23, S40, S35, S36, S38]
07:00:00	DC-2	TOE	Order = [S17, S20, S15, S33, S30, S14, S28, S41, S42, S26, S40, S38]

07:00:00	DC-3	TOE	Order = [S17, S20, S18, S19, S34, S33, S30, S31, S32, S45, S41, S42, S44, S46]

Table 7-5: The rescheduling results of scenario 2 of TOE

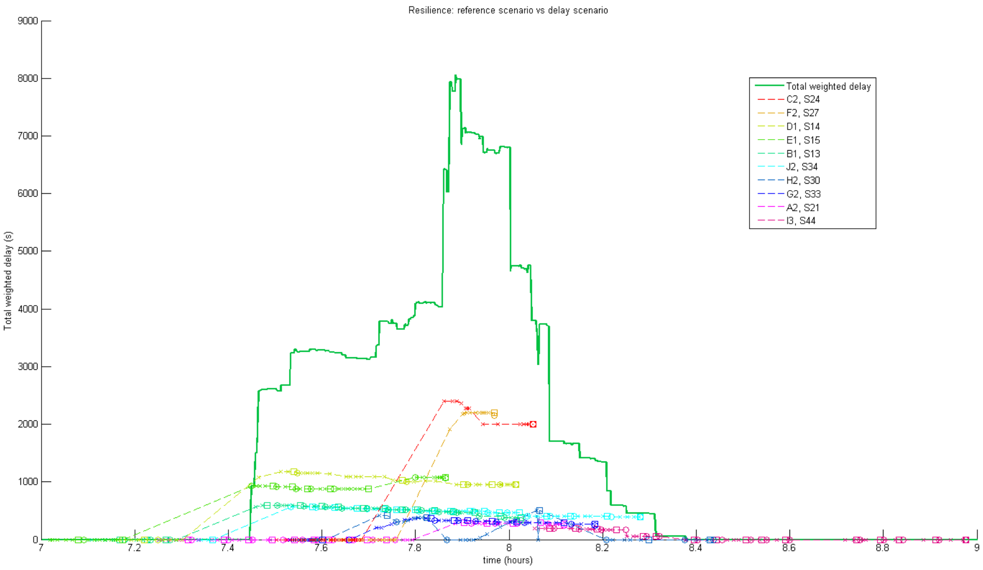


Figure 7-9: the delay propagation graph of scenario 2 of TOE

Rescheduling time	**Junction area**	**Algorithm**	**Order of the trains**
07:00:00	DC-1	FCFS	Order = [S11, S16, S12, S15, S21, S13, S27, S14, S28, S22, S25, S37, S26, S39, S23, S40, S35, S36, S38]
07:00:00	DC-2	FCFS	Order = [S17, S20, S33, S30, S15, S14, S28, S41, S42, S26, S40, S38]
07:00:00	DC-3	FCFS	Order = [S17, S20, S18, S19, S33, S30, S34, S31, S32, S45, S41, S42, S44, S46]

Table 7-6: The rescheduling results of scenario 2 of FCFS

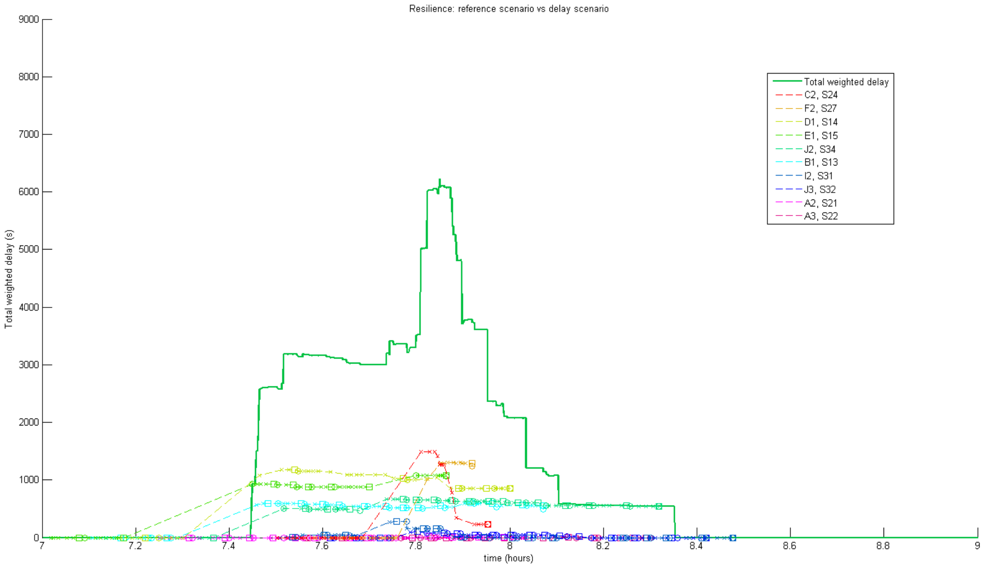


Figure 7-10: the delay propagation graph of scenario 2 of FCFS

Rescheduling time	**Junction area**	**Algorithm**	**Order of the trains**
07:00:00	DC-1	TOE	Order = [S11, S16]
07:00:00	DC-2	TOE	Order = [S17, S20, S15, S33, S30, S14, S28, S41, S42, S26, S40, S38]
07:00:00	DC-3	TOE	Order = [S17, S20, S18, S19, S34, S33, S30, S31, S32, S45, S41, S42, S44, S46]
07:33:00	DC-1	BF	Order = [S12, S24, S15, S13, S21, S14, S22, S28]
07:46:08	DC-1	TS	Order = [S27, S13, S21, S14, S22, S28, S25, S23, S26]
07:56:27	DC-1	TOE	Order = [S28, S22, S25, S26, S23]
08:11:08	DC-1	TOE	Order = [S37, S26, S39, S23, S40, S35, S36, S38]

Table 7-7: The rescheduling results of scenario 2 of AAs applied on DC-1

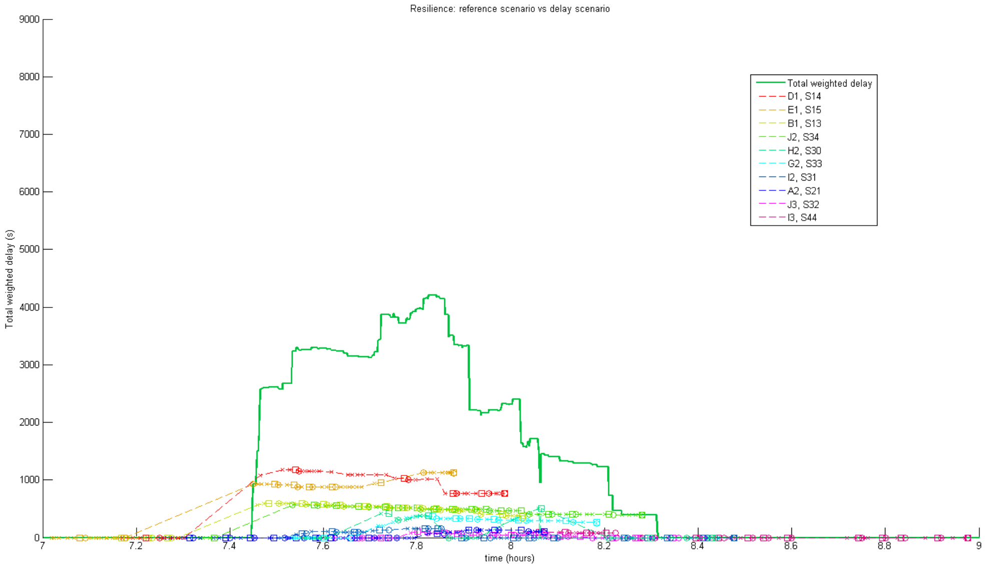


Figure 7-11: The delay propagation graph of scenario 2 of AAs applied on DC-1

Rescheduling time	**Junction area**	**Algorithm**	**Order of the trains**
07:00:00	DC-1	TOE	Order = [S11, S16]
07:00:00	DC-2	TOE	Order = [S17, S20, S33]
07:00:00	DC-3	TOE	Order = [S17, S20, S18, S19, S34]
07:33:00	DC-1	BF	Order = [S12, S24, S15, S13, S21, S14, S22, S28]
07:33:53	DC-2	TOE	Order = [30]
07:35:35	DC-3	BF	Order = [S33, S34, S30, S31, S32]
07:46:08	DC-1	TS	Order = [S27, S13, S21, S14, S22, S28, S25, S23, S26]
07:50:46	DC-2	TOE	Order = [S15, S14]
07:56:27	DC-1	TOE	Order = [S28, S22, S25, S26, S23]
08:00:56	DC-3	TOE	Order = [S45]
08:01:52	DC-2	TOE	Order = [S28, S41]
08:03:53	DC-2	TOE	Order = [S42]
08:05:35	DC-3	FCFS	Order = [S41, S42, S44, S46]

08:11:08	DC-1	TOE	Order = [S37, S26, S39, S23, S40, S35, S36, S38]
08:18:49	DC-2	TOE	Order = [S26, S40, S38]

Table 7-8: The rescheduling results of scenario 2 of AAs applied on DCs

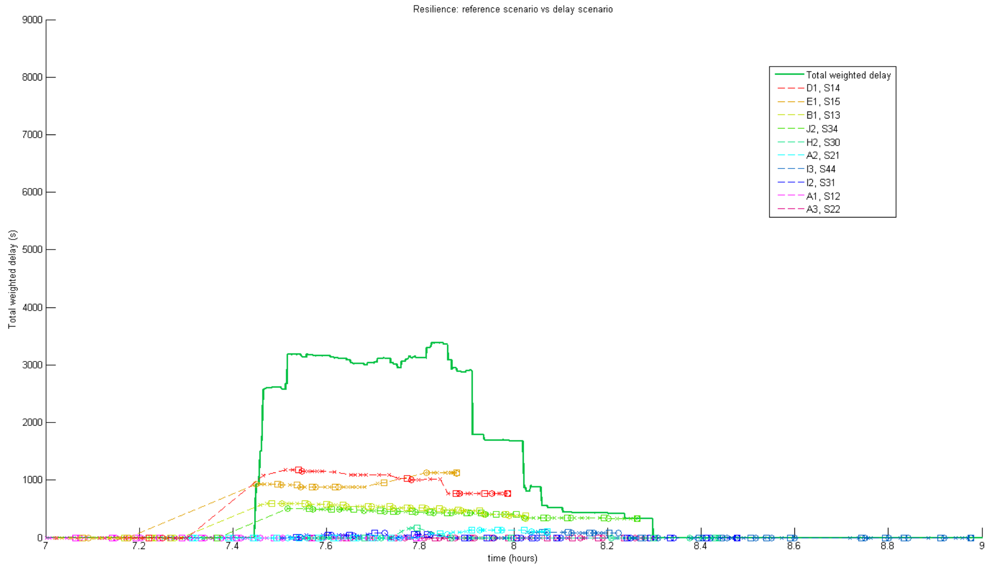


Figure 7-12: The delay propagation graph of scenario 2 of AAs applied all DCs

Figure 7-9 shows a bad performance when TOE is applied to each DC because there is no rescheduling strategy used to help to recover delay. From Figure 7-10 we can observe an improvement in the total weighted delay by applying FCFS to the junctions. As in Figure 7-7, there is an increase in the total weighted delay by applying FCFS because the delay on E1 at Finsbury Park Station delays the trains behind it. Figure 7-11 shows the weighted delay propagation obtained by applying AAs to DC-1; the total weighted delay is significantly restored. However, due to the delay situation, all DCs are influenced. When the delayed scenario is run by applying AAs on all DCs, Figure 7-12 shows a better performance on the total weighted delay propagation.

A summary of the KPIs for each graph in this section is presented in the next section, and a discussion about the performance of supervisory control on a large scale network is given at the end of this chapter.

7.4.5 Summary

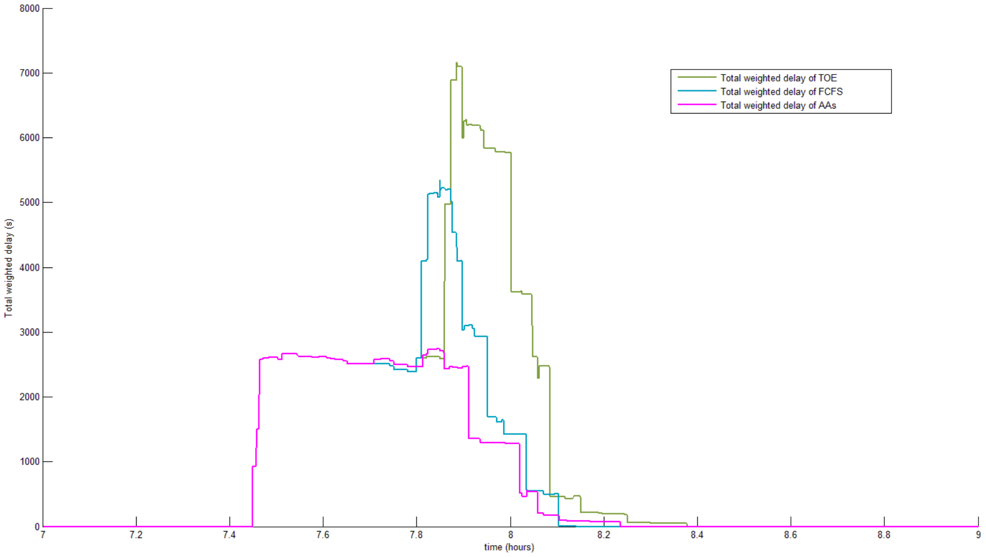


Figure 7-13: A comparison of total weighted delays when different strategies applied in scenario 1

Figure 7-13 gives a direct view on the instantaneous total weighted delays when different strategies applied to the scenario 1. The green line (when TOE is applied) has a large influx from $t\ = 7.8$ to $t = 8$ because trains from other lines on time are waiting at the junction until the delayed train E1/S15 has passed through. The blue line (when FCFS is applied) still has an influx because train E1/S15 has been given the authority to pass the junction before the second delay happened. The pink line (when AAs is applied) shows a good recovery without any significant influx.

For each of control strategy, the result from the cost function (i.e. total delay penalty) and the three KPIs of the delay propagation - maximum lateness, time to recover and integral of delay of each scenario are collected and summarised in Table 7-9 and Table 7-10. It can also be given as a proportion of the maximum possible integral of delay as: Integral of delay proportion = Integral of delay/ (Maximum lateness* Time to recover). This value gives an indication of how severe the delays were relative to the maximum lateness throughout the period of delay.

Rescheduling strategy	**Total delay penalty (£)**	**Maximum weighted Lateness (s)**	**Time to recover (s)**	**Integral Delay (s^2) (normalised value (%))**	**Proportion of Integral Delay (%)**
TOE	674.3	7165	3055	7943800 (100)	36.29
FCFS	371	5345	2352	5917500 (74.49)	47.07
AAs	229.2	2737	2829	4845800 (61)	62.58

Table 7-9: Comparison information of different strategies of scenario 1

Table 7-9 shows that TOE takes the biggest total delay penalty, and AAs takes the smallest value, and AAs also has the smallest value of Maximum total weighted lateness, but FCFS takes the shortest time to recover from the disturbed situation. The integral delay shows that the system suffers less delay during the time of observation when AAs is applied, and the proportion of integral delay indicates there are no significant increases or decreases on the total weighted lateness when AAs is applied.

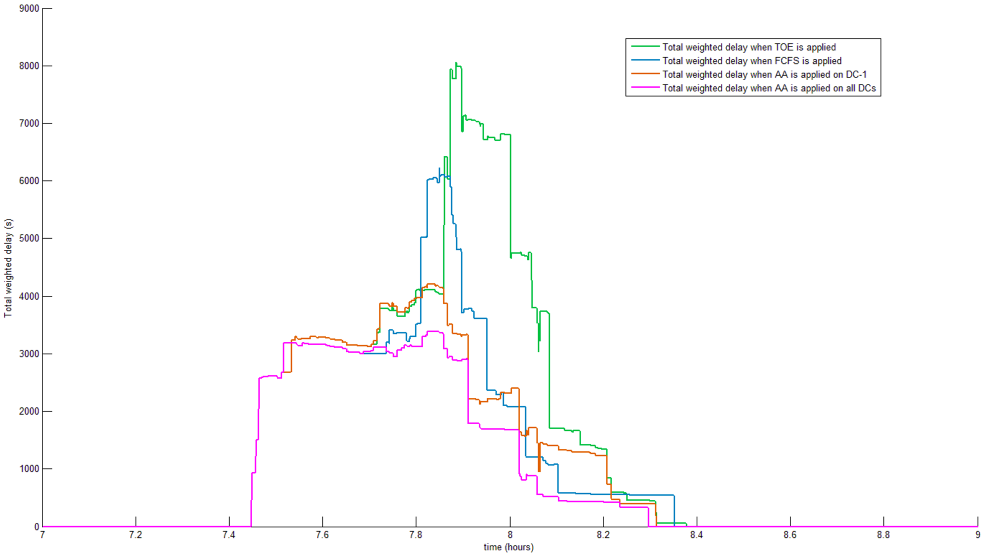


Figure 7-14: A comparison of total weighted delays when different strategies applied in scenario 2

Figure 7-14 gives a direct view on the instantaneous total weighted delays when different strategies applied to the scenario 2. The green line (when TOE is applied) has the worst performance compare with other three due to no rescheduling strategy has been applied, which leads to the trains from other lines on time should wait at the junction until the delayed trains has passed through. The blue line (when FCFS is applied) still has an influx because of the same reason as in scenario 1. The orange line (when AAs is applied on DC-1) shows a better recovery compare with the green line and the blue line, however, the delays on J2/S34 still influence the traffic flows in DC-2 and DC-3. The pink line (when AAs is applied on all DCs) shows a good recovery without any significant influx because all DCs are controlled with rescheduling strategies.

Rescheduling strategy	**Total delay penalty (£)**	**Maximum weighted Lateness (s)**	**Time to recover (s)**	**Integral Delay (s^2) (normalised value (%))**	**Proportion of Integral Delay (%)**
TOE	738.1	8056	3102	10597535 (100)	42.41

FCFS	422.6	6225	3244	7823300 (73.82)	38.74
AAs on DC-1	425.8	4211	3102	7498280(70.75)	57.4
AAs on all DCs	262.8	3390	3045	6133000(57.87)	59.41

Table 7-10: Comparison information of different strategies of scenario 2

Table 7-10 shows that TOE takes the biggest total delay penalty, and AAs on all DCs takes the smallest value. AAs on DC-1 takes the second biggest total delay penalty, and FCFS takes a slight smaller value. Even through FCFS takes smaller total delay penalty compared with AAs on DC-1, the values of three KPIs are bigger than AAs on DC-1. Table 7-10 shows the smallest value of Maximum total weighted lateness can be found when AAs are applied to all DCs, and this also gives the shortest time to recover. A better performance on delay recovery in the system is achieved by implementing AAs to all DCs.

7.5 Conclusions

In this chapter, a Distributed Artificial Intelligence (DAI) system is chosen to solve the railway network rescheduling problem due to the complexity and scale of the network. The concept of DAI is given and a typical model (the blackboard architecture) used to construct DAI are described.

The global rescheduling problem has been divided into several individual local rescheduling problems and a blackboard architecture for railway network rescheduling based on a distributed artificial intelligence (DAI) system is proposed and structured to solve this problem. The whole network has been divided into different control centres (DCs) and each DC indicates a core junction area along the network. These DCs work individually and independently, but the network database and control memories are shared by all of them. The components of this architecture are detailed and illustrated with examples.

Finally, a case study using the ECML is presented to verify the feasibility and efficiency of this control architecture to the multi-junction train rescheduling problem. The advantages are that this method may increase the efficiency of rescheduling in a large scale network because all DCs are processing in parallel and asynchronously, and this method achieves coordination between each DC by controlling their supervisor through the global control shell. The rescheduling results are proven to be feasible and efficient for a multi-junction rescheduling problem with continuous traffic flows and dynamic traffic. The disadvantage is that global optimisation cannot be guaranteed because of the distributed architecture, and boundaries of the DCs should be carefully defined, which will influence the final results.

CHAPTER 8

CONCLUSION AND FURTHER WORK

8.1 Conclusion

The occurrence of disruptions leads to a chain of knock-on delays in the railway network due to the growth of traffic demands and limited resources. In this thesis, rescheduling approaches are studied to minimise the effect of these delays. The purpose of real-time train rescheduling is to find a conflict-free operating schedule for the services in a given area and a given time horizon during operations, providing a solution that is compatible with the actual traffic conditions and infrastructure constraints. The main objective of this thesis is to provide the dispatchers with better decision support by considering the delay propagation of the whole railway network with continuous traffic flows and mixed services after disruptive events.

Firstly, the resilience of a railway network is defined at three levels: stable, robust and recoverable according to the system response to a delay. By analysing the train running information from the simulator, a time-based delay propagation graph is developed to visualise and evaluate the delay recovery performance.

A number of existing railway rescheduling optimisation approaches have been introduced and applied in a series of scenarios for minimising the delays. A common benchmark case study is designed to compare the goodness of different rescheduling algorithms. The results have determined the appropriate applications for each algorithm by comparing them in the same delay scenario. A cost function of total delay penalty (TDP) and the computation time (CT) are used to evaluate the performance of each algorithm.

The characteristics of the applications for each algorithm can be concluded from their comparison, and some suggestions for choosing algorithms to use in the dynamic rescheduling can be formulated. Table 5-5 shows an example of choosing algorithms for different networks and delay conditions after considering the characteristics of each algorithm set out in Chapter 5.

In addition, a performance-based supervisory control on railway rescheduling management has been developed to solve the rescheduling problem by combining different algorithms together with a specified sequence and time of changing. This control method provides a way of changing algorithms to deal with continuous traffic flows and mixed services after disruptive events. Algorithms are selected by the local area based on the current network and delay conditions. A case study is provided to show the efficiency of alternating algorithms to support a complicated scenario.

Finally, to deal with the multi-junction train rescheduling problem, a DAI system architecture of train rescheduling in a large scale network is introduced. This system is constructed based on a blackboard model with individual decision centres (DC) and a global control shell. The DC is formed of a sensor module, a local rescheduling module, a decision execution module and a supervisor. The sensors are in charge of collecting information from the whole railway network database. The local rescheduling systems generate reschedule plans based on the local information, the execution modules take charge of transmitting new production plans and the supervisors are responsible for selecting algorithms and determining the time of application. All DCs work individually and independently, but the network database and control memories are shared by all DCs. A case study using the ECML is presented to verify the feasibility and efficiency of this control architecture to the multi-junction train rescheduling problem. The advantages are that this method may increase the efficiency of rescheduling in a large scale network because all DCs are processing in parallel and asynchronously, and this method achieves coordination between each DC by controlling their supervisor through the global control shell. The rescheduling results are proven to be feasible and efficient to a multi-junction rescheduling problem with continuous traffic flows and dynamic situation. The disadvantage is that global optimisation cannot be guaranteed, and boundaries of the DCs should be carefully defined, which may influence the final results.

8.2 Further Work

Based on the case studies of the optimal single-junction rescheduling and the multi-junction control approach, further tasks are proposed to extend the work.

(1) The reported research focuses on the reordering of trains in the junction area. Some operational management measures such as train rerouting, platforming at the junctions, cancelling a service or reallocating a driver could be considered in the future in order to develop a more comprehensive decision support system.

(2) The supervisory control system selects algorithms based on some rules which have been derived through a comparison of a number of advanced algorithms used in different scenarios in this thesis. The reliability of these rules will impact the decision made by the control centre. One area of future work is to investigate the relationship between the network conditions and the appropriateness of using the algorithms. This will give a clearer understanding of algorithm application, and it will help to select more appropriate algorithms.

(3) Although the supervisory control system was tested on the ECML with three junctions included, the generalisation of this approach to a larger railway network with a more complicated topological structure should be studied and justified. In addition, the rescheduling process, including transmission of network conditions, conflict detection, the prediction of train current state, as well as the delivery of solution plans, can also be evaluated in more detail.

(4) This preliminary research on train rescheduling for railway networks was based on an off-line laboratory test environment with computer simulation experiments. Due to the limitations of simulation modelling, differences exist between the modelling and real railway traffic management. The validation of the algorithms and control architectures proposed in this research to practical railway traffic systems could be necessary in future.

(5) Considering the energy efficiency, a dispatching system should be able to control the speed of the train, reduce unnecessary train stops and smooth the traffic flow of the junction area in the railway network in order to lower carbon emissions and improve the quality of service; further work should take optimal energy consumption into account.

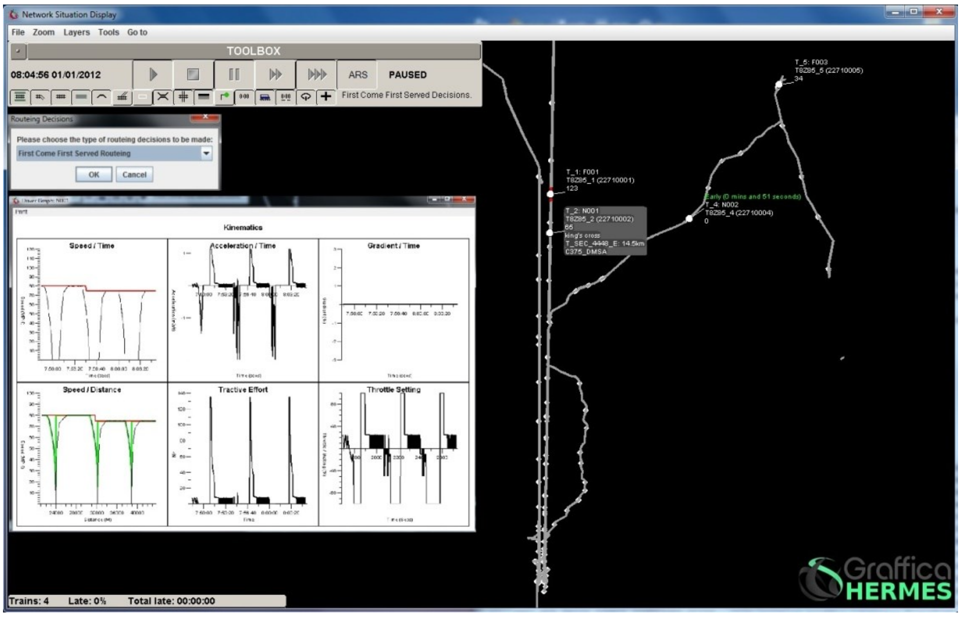
APPENDIX A

HERMES: A MICROSCOPIC SIMULATIOR OF RAILWAY TRAFFIC

The Holistic Environment for Rail Modelling and Experimental Simulation (HERMES) simulator is an object-oriented microscopic railway simulator produced by the British company GRAFFICA that models all aspects of a railway operation [131]. The simulation model consists of six different components respectively representing the infrastructure (signals, points and stations), the rolling stock, the signalling systems, the interlocking, the timetable and the human behaviour (driver/passenger behaviour or dispatcher decisions).

Figure A-0-1 shows the user view of the simulator, including track information, train types, timing window, routing decision window, and train running characteristics. HERMES is highly flexible, allowing the modification of many parameters. For example, the timetable information, train and routing information are all editable in the simulation.

Moreover, it is equipped with an open set of APIs that allow the users to introduce traffic disturbances, infrastructure disruptions and/or customise functions relative to the driver behaviour or the route setting. The simulator is used in this thesis to run trains on a section of a railway network. It may be run as timetabled, and also with a delay that is inserted artificially. HERMES can then calculate the running of trains after the introduction of the delay. The times of arrival and departure at stations and passing times at selected signals can be recorded for all trains. These points are called timing points in this study. From this information, details of the propagation of delays and of the resilience of the system can be derived.



First Come First Served Decisions.

Please choose the type of routeing decisions to be made:

First Come First Served Routeing

Trains: 4 Late: 0% Total late: 00:00:00

Figure A-0-1: User interface of the HERMES simulator.

To summary, HERMES is an interface to the dispatchers to provide data of event types, subscriptions and the actual state of the network during simulation [132]. The key benefits of HERMES include [131]:

- Provides a microscopic platform to provide a high-fidelity simulation of railway operations;
- Simulates the actions and models the behaviour of the driver;
- Reproduces the characteristics of train operation of railway networks;
- Predicts potential conflicts between trains at junctions;
- Provides support for timetable development and validation;
- Accurately negotiates the operational mechanisms behind the aspects of railway signalling;
- Models ATO in ERTMS Levels 1, 2 and 3;
- Provides support for accurate and detailed mapping of rail infrastructure.

APPENDIX B

RESULTS OF THE CASE STUDY IN CHAPTER 5

(a) Results analysis for scenarios 1.1 to 1.4.1

The results include two main indictors which are TDP and CT for each algorithm, and are recorded to address different scenarios, Table B-1 show the results for scenarios 1.1 to 1.4.1.

Approaches	**TDP(pence) & CT(ms)**	**Numbers of trains**			
		7	10	13	14
TOE	TDP	372.0	465.0	569.0	671.0
	CT	1	1	1	1
FCFS	TDP	268.0	308.0	356.0	382.8
	CT	1	1	1	1
FLFS	TDP	372.0	465.0	569.0	671.0
	CT	9	12	18	23
BF	TDP	263.0	298.0	342.0	361.4
	CT	1104	17117	513252	10685345
DP	TDP	263.0	298.0	342.0	361.4
	CT	399	7991	235449	2811818
DTBE	TDP	263.0	298.0	342.0	361.4
	CT	185	1460	17986	119640
ACO	TDP	263.0	298.0	342.0	367.4
	CT	146	272	1049	5794
GA	TDP	263.0	300.14	356.23	364.01
	CT	226	727	4958	20074
SA	TDP	263.0	298.0	342.0	404.164
	CT	243	952	9214	19835

TS	TDP	263.0	298.0	342.0	362.6
	CT	15	35	299	2072.1
LS	TDP	263.0	298.0	342.0	378.602
	CT	53	239	3837	21030

Table B-1: Results of scenarios 1.1 to 1.4.1

According to the total delay penalty results, we can rank different algorithms by the quality of the solutions. The ranks of different algorithms by comparing their TDP in different delayed scenarios are represented below:

- For scenario 1.1 the rank is BF = DP = DTBE = ACO = GA = SA = TS = LS < FCFS < FLFS = TOE.
- For scenario 1.2 the rank is BF = DP = DTBE = ACO = SA = TS = LS < GA < FCFS < FLFS = TOE.
- For scenario 1.3 the rank is BF = DP = DTBE = ACO = SA = TS = LS < FCFS < GA < FLFS = TOE.
- For scenario 1.4.1 the rank is BF = DP = DTBE < TS < GA < ACO < LS < FCFS < SA < FLFS = TOE.

(b) Results analysis for scenarios 1.4.1 to 1.4.6

Table 4.7 shows the results for scenarios 1.4.1 to 1.4.6. All scenarios are 14-train scenarios with the same original timetable, with an increasing delay on train 01. The total delay penalty and computation time for the different algorithms are recorded in Table 4.7 with a delay size from 2 minutes to 30 minutes.

Approaches	**TDP(pence) & CT(ms)**	**Delay on Train 01 (minutes)**					
		2	5	10	15	20	30
TOE	TDP	671.0	2555.2	5783.0	9023.0	12263.0	18743.0

	CT	1	1	1	1	1	1
FCFS	TDP	382.8	1500.8	3400.6	4757.0	6298.0	9898.0
	CT	1	1	1	1	1	1
FLFS	TDP	671.0	1568.4	3252.8	4592.2	6284.0	9884.0
	CT	23	20	17	18	18	20
BF	TDP	361.4	1283.0	2813.8	4592.2	6284.0	9884.0
	CT	10685345	7437426	9265637	8142909	6993564	6987316
DP	TDP	361.4	1283.0	2813.8	4592.2	6284.0	9884.0
	CT	2811818	2650093	2630334	2563068	2808200	2874303
DTBE	TDP	361.4	1283.0	2852.6	4592.2	6284.0	9884.0
	CT	119640	77077	38778	30530	24494	20590
ACO	TDP	367.4	1536.91	3291.56	4738.61	6284	9884
	CT	5794.162	6092.14	6951.22	6900.71	5486.28	5394.49
GA	TDP	364.01	1424.08	2962.35	4650.72	6284	9884
	CT	20073.63	16967.66	16413.58	18023.05	12813.32	12786.87
SA	TDP	404.164	1421.3	3013.96	4670.55	6284	9884
	CT	19835.29	17501.34	15301.22	14524.89	14315.48	15004.79
TS	TDP	362.6	1323.32	3340.6	4744	6284	9884
	CT	2072.28	2042.05	1975.55	1637.94	1515.13	1595.14
LS	TDP	378.602	1312.00	2817.81	4598.90	6284	9884
	CT	21029.96	13497.07	10947.47	10477.87	7072.28	6906.99

Table B-2: Results of scenarios 1.4.1 to 1.4.6

According to the results of the total delay penalty of each column, we can rank the different algorithms as below:

- In scenario 1.4.1, the rank is BF = DP = DTBE < TS < GA < ACO < LS < FCFS < SA < FLFS = TOE.

- In scenario 1.4.2, the rank is BF = DP = DTBE < LS < TS < SA < GA < ACO < FCFS < FLFS < TOE.
- In scenario 1.4.3, the rank is BF = DP < LS < DTBE < GA < SA < FLFS < ACO < TS < FCFS < TOE.
- In scenario 1.4.4, the rank is BF = DP = DTBE = FLFS < LS < GA < SA < ACO < T S < FCFS < TOE.
- In scenario 1.4.5, the rank is BF = DP = DTBE = FLFS = ACO = GA = SA = TS = LS < FCFS < TOE.
- In scenario 1.4.6, the rank is BF = DP = DTBE = FLFS = ACO = GA = SA = TS = LS < FCFS < TOE.

(c) Results analysis for scenarios 2.1 to 2.3

Table 4.8 shows the results for scenarios 2.1 to 2.3. These are all 14-train scenarios with the same original timetable; the speed of train 01 is kept as 80%, 50% and 20% of the original speed due to engine failure. Total delay penalty and computation time for different algorithms are recorded in Table 4.8 with the remaining speeds.

Approaches	**TDP(pence) &CT(ms)**	**Remaining speed /original speed**		
		80%	50%	20%
TOE	TDP	903.4	7656.8	37991
	CT	1	1	1
FCFS	TDP	450.8	4078.0	19397
	CT	1	11	1
FLFS	TDP	903.4	3868.2	19397
	CT	13	15	12
BF	TDP	427.4	3865	19493
	CT	7771248	9976705	10991864
DP	TDP	427.4	3865	19493
	CT	2585475	3419362	3603199

DTBE	TDP	427.4	3914	19493
	CT	112912	53938	37413
ACO	TDP	435.068	4041.11	19493
	CT	4443.8	5982.04	97331.71
GA	TDP	434.94	3884.042	19500.35
	CT	15864.83	16083.06	29954.31
SA	TDP	464.154	3934.854	19493
	CT	15122.09	13181.71	23528.17
TS	TDP	428.84	4006.52	19493
	CT	1458.79	1423.83	2184.49
LS	TDP	437.888	3871.166	19493
	CT	12800.35	8850.8	18069.1

Table B-3: Results of scenarios 2.1 to 2.3

According to the results giving the total delay penalty of each column, we can rank different algorithms by their performance.

- In scenario 2.1, the rank is BF = DP = DTBE < TS < GA < ACO < LS < FCFS < SA < FLFS = TOE.
- In scenario 2.2, the rank is BF = DP < FLFS < LS < GA < DTBE < SA < ACO < TS < FCFS < TOE.
- In scenario 2.3, the rank is FCFS = FLFS < BF = DP = DTBE = ACO =TS = LS = SA <GA <TOE.

(d) Results analysis for scenarios 3.1 to 3.5

Scenarios 3.1 to 3.5 deal with different initial delays which occur on different trains, and results are shown in the Table 4.9 below.

Approaches	**TDP(pence) &CT(ms)**	**Delay on Train 01 + Delay on Train 04 (minutes)**				
		2+2	2+5	2+10	5+2	10+2
TOE	TDP	695.8	850.6	1408.6	2570.8	5783.0

	CT	1	1	1	1	1
FCFS	TDP	401.6	457.4	704.4	1394.6	3283.8
	CT	1	1	1	1	1
FLFS	TDP	695.8	749.6	975.4	1584.0	3304.8
	CT	19	18	16	23	19
BF	TDP	386.2	441.2	689.4	1298.6	2876.8
	CT	10486682	10454257	10794885	10063199	10931261
DP	TDP	386.2	441.2	689.4	1298.6	2876.8
	CT	3208596	3612387	2889143	3357818	2931350
DTBE	TDP	386.2	441.2	689.4	1298.6	3001.4
	CT	155719	132986	111959	82317	38520
ACO	TDP	386.2	441.2	689.4	1440.566	3152.858
	CT	6484.83	6734.3	5009.66	5268.29	4942.29
GA	TDP	395.144	444.19	734.47	1376.48	2917.21
	CT	22512.46	19868.95	17187.18	14574.38	13714.54
SA	TDP	429.604	448.01	705.12	1391.81	2983.912
	CT	23183.97	21148.16	18656.5	16143.99	12891.45
TS	TDP	386.2	441.2	689.4	1344.488	3001.63
	CT	2286.63	1753.67	1653.96	1502.57	1302.91
LS	TDP	398.608	445.4	689.4	1328.714	2877.824
	CT	18188.85	18372.44	16332.18	9443.93	7919.01

Table B-4: Results of scenarios 3.1 to 3.5

According to the results showing the total delay penalty of each column, we can rank the different algorithms by their performance.

- In scenario 3.1, the rank is BF = DP = DTBE =ACO =TS < GA < LS < FCFS < SA < FLFS = TOE.
- In scenario 3.2, the rank is BF = DP = DTBE =ACO =TS < GA < LS < SA< FCFS < FLFS < TOE.
- In scenario 3.3, the rank is BF = DP = DTBE =ACO =TS= LS < FCFS < SA< GA < FLFS < TOE.

- In scenario 3.3, the rank is BF = DP = DTBE < LS < TS < GA < SA < FCFS < ACO< FLFS < TOE.
- In scenario 3.3, the rank is BF = DP< LS < GA < SA < DTBE < TS < ACO < FCFS < FLFS < TOE.

(e) Results analysis for scenarios 4.1 to 4.3

Considering a multiple additive delay situation, scenarios 4.1 to 4.3 are designed with additive delays caused by engine failure which occurs to multiple trains. Table 4.10 shows the results for different algorithms in these three scenarios. The columns are formed as percentage + percentage, which means the remaining speed / original speed on train 01 + remaining speed / original speed on train 04.

Approaches	**TDP(pence) & CT(ms)**	**Remaining speed / Original speed on Train 01 + Remaining speed / Original speed on Train 04**		
		80%+50%	50%+50%	20%+50%
TOE	TDP	3117.8	8754.8	49520.1
	CT	1	1	1
FCFS	TDP	1460.6	4720.6	20497.6
	CT	1	1	1
FLFS	TDP	1937.6	4706.6	20497.6
	CT	9	13	10
BF	TDP	1468.6	4706.6	20497.6
	CT	7816483	11539756	14072195
DP	TDP	1468.6	4706.6	20497.6
	CT	2633280	3828622	3562152
DTBE	TDP	1465.6	4706.6	20497.6
	CT	107735	114516	116832
ACO	TDP	1454.61	4112.43	20497.6
	CT	4342.5	5333.45	4963
GA	TDP	1471.82	4878.196	20497.6
	CT	14830.35	14926.78	13250

SA	TDP	1462.18	4706.6	20497.6
	CT	15758.6	14777.58	13211.2
TS	TDP	1453.6	4706.6	20497.6
	CT	1293.63	1177.8	2000.9
LS	TDP	1457.44	4706.6	20497.6
	CT	12590.99	6054.03	7140

Table B-5: Results of scenarios 4.1to 4.3

According to the results for total delay penalty of each column, the different algorithms can be ranked by their performance.

- In scenario 4.1, the rank is TS < ACO < LS< FCFS < SA < DTBE < BF = DP < GA < FLFS = TOE.
- In scenario 4.2, the rank is ACO < BF = DP = DTBE = SA=TS=LS =FLFS < FCFS<GA <TOE.
- In scenario 4.3, the rank is BF = DP = DTBE =ACO =TS= LS = SA= GA= FCFS =FLFS < TOE.

APPENDIX C

NOMINAL TIMETABLE OF THE CASE STUDY IN CHAPTER 7

The model in Chapter 7 is carried out with a simplified timetable with 37 services running from 7:00 am to 9:00am based on a practical timetable published by Network Rail [92]. The following tables give the information of these 37 services, including the Train numbers, Service IDs/Train IDs, Types of vehicles, Departure time at the stations, Minimum and Maximum dwell time at the stations and Arrival time at the termination.

Timetable 1: Hertford to Moor Gate

	1	2	3	4	5
	S12/A1	S21/A2	S22/A3	S23/A4	S35/A5
	Class 313	Class 313	Class 313	Class 313	Class 313
Hertford	07:00:00	07:15:00	07:30:00	07:45:00	08:00:00
Bayford	07:04:21 30s~30s	07:19:21 30s~30s	07:34:21 30s~30s	07:49:21 30s~30s	08:04:21 30s~30s
Cuffley	07:09:00 30s~30s	07:24:00 30s~30s	07:39:00 30s~30s	07:54:00 30s~30s	08:09:00 30s~30s
Crews Hill	07:12:00 30s~30s	07:27:00 30s~30s	07:42:10 30s~30s	07:57:10 30s~30s	08:12:10 30s~30s
Gordon Hill	07:16:00 60s~60s	07:31:00 60s~60s	07:46:00 60s~60s	08:01:00 60s~60s	08:16:00 60s~60s
Enfield Chase	07:18:30 60s~60s	07:33:30 60s~60s	07:48:30 60s~60s	08:03:30 60s~60s	08:18:30 60s~60s
Grange Park	07:20:30 30s~30s	07:35:30 30s~30s	07:50:20 30s~30s	08:05:20 30s~30s	08:20:20 30s~30s

Winchmore Hill	07:22:30 30s~30s	07:37:30 30s~30s	07:52:30 30s~30s	08:07:30 30s~30s	08:22:30 30s~30s
Palmers Green	07:25:00 30s~30s	07:40:00 30s~30s	07:55:00 30s~30s	08:10:00 30s~30s	08:25:00 30s~30s
Bowes Park	07:27:30 30s~30s	07:42:30 30s~30s	07:57:30 30s~30s	08:12:30 30s~30s	08:27:30 30s~30s
Alexandra Palace	07:30:30 60s~60s	07:45:30 60s~60s	08:00:23 60s~60s	08:15:23 60s~60s	08:30:23 60s~60s
Hornsey	…	…	…	…	…
Harringay	…	…	…	…	…
Finsbury park	07:35:30 60s~75s	07:50:30 60s~75s	08:05:32 60s~75s	08:20:32 60s~75s	08:35:32 60s~75s
Drayton Park	07:38:30 60s~60s	07:53:30 60s~60s	08:08:33 60s~60s	08:23:33 60s~60s	08:38:33 60s~60s
Highbury & Islington	…	…	…	…	…
Essex Road	07:41:22 30s~30s	07:56:22 30s~30s	08:11:22 30s~30s	08:26:22 30s~30s	08:41:22 30s~30s
Old Street	07:45:30 60s~60s	08:00:30 60s~60s	08:15:30 60s~60s	08:30:30 60s~60s	08:45:30 60s~60s
Moor Gate (Arrive Time)	07:48:00	08:03:00	08:18:00	08:33:00	08:48:00

Timetable 2: Welwyn Garden City to Moorgate

	6	**7**	**8**	**9**	**10**	**11**
	S11/C1	**S24/C2**	**S37/C3**	**S13/B1**	**S25/B2**	**S36/B3**
	Class 91	Class 91	Class 91	Class 313	Class 313	Class 313
Welwyn Garden City	07:00:00	07:30:00	08:00:00	07:06:00	07:36:00	08:06:00
Hatfield	…	…	…	07:10:36 30s~30s	07:40:36 30s~30s	08:10:36 30s~30s

Welham Green	…	…	…	07:14:00 30s~30s	07:44:00 30s~30s	08:14:00 30s~30s
Brookmans Parks	…	…	…	07:16:30 30s~30s	07:46:30 30s~30s	08:16:30 30s~30s
Potters Bar	…	…	…	07:20:30 75s~90s	07:50:30 75s~90s	08:20:30 75s~90s
Hadley Wood	…	…	…	07:24:00 30s~30s	07:54:00 30s~30s	08:24:00 30s~30s
New Barnet	…	…	…	07:27:00 30s~60s	07:57:00 30s~60s	08:27:00 30s~60s
Oakleigh Park	…	…	…	07:29:30 30s~60s	07:59:30 30s~60s	08:29:30 30s~60s
New Southgate	…	…	…	07:32:45 30s~30s	08:02:45 30s~30s	08:32:45 30s~30s
Alexandra Palace	…	…	…	07:36:00 30s~30s	08:06:00 30s~30s	08:36:00 30s~30s
Hornsey	…	…	…	07:38:00 30s~30s	08:08:00 30s~30s	08:38:00 30s~30s
Harringay	…	…	…	07:40:15 30s~30s	08:10:15 30s~30s	08:40:15 30s~30s
Finsbury park	…	…	…	07:43:00 30s~60s	08:13:00 30s~60s	08:43:00 30s~60s
Drayton Park	…	…	…	07:45:30 30s~60s	08:15:30 30s~60s	08:45:30 30s~60s
Highbury & Islington	…	…	…	…	…	…
Essex Road	…	…	…	07:49:30 30s~30s	08:19:30 30s~30s	08:49:30 30s~30s
Old Street	…	…	…	07:53:00 30s~60s	08:23:00 30s~60s	08:53:00 30s~60s
Moor Gate	07:20:00	07:50:00	08:20:00	07:56:01	08:26:01	08:56:01

Timetable 3: Welwyn Garden City to Kings Cross

	12	13	14	15	16	17
	S16/F1	**S27/F2**	**S39/F3**	**S14/D1**	**S26/D2**	**S38/D3**
	Class 91	Class 91	Class 91	Class 317	Class 317	Class 317
Welwyn Garden City	07:05:00	07:35:00	08:05:00	07:10:00	07:40:00	08:10:00
Hatfield	…	…	…	07:15:00 30s~60s	07:45:00 30s~60s	08:15:00 30s~60s
Welham Green	…	…	…	…	…	…
Brookmans Parks	…	…	…	…	…	…
Potters Bar	…	…	…	07:23:15 30s~60s	07:53:15 30s~60s	08:23:15 30s~60s
Hadley Wood	…	…	…	…	…	…
New Barnet	…	…	…	…	…	…
Oakleigh Park	…	…	…	…	…	…
New Southgate	…	…	…	…	…	…
Alexandra Palace	…	…	…	07:38:30 30s~45s	08:08:30 30s~45s	08:38:30 30s~45s
Hornsey	…	…	…	…	…	…
Harringay	…	…	…	…	…	…
Finsbury park	…	…	…	07:45:10 30s~60s	08:15:10 30s~60s	08:45:10 30s~60s
Holloway	…	…	…	…	…	…
Belle Isle	…	…	…	07:49:30 60s~60s	08:19:30 60s~60s	08:49:30 60s~60s
Kings Cross	07:21:00	07:51:00	08:21:00	07:52:30	08:22:30	08:52:30

Timetable 4: Welwyn Garden City to Midland Road Junction

	18	**19**	**20**
	S15/E1	**S28/E2**	**S40/E3**
	TL 39	TL 39	TL 39
Welwyn Garden City	07:01:00	07:31:00	08:01:00
Hatfield	07:05:30 30s~30s	07:35:30 30s~30s	08:05:30 30s~30s
Welham Green	…	…	…
Brookmans Parks	…	…	…
Potters Bar	07:11:30 30s~60s	07:41:30 30s~60s	08:11:30 30s~60s
Hadley Wood	07:15:00 30s~30s	07:45:00 30s~30s	08:15:00 30s~30s
New Barnet	07:18:00 30s~60s	07:48:00 30s~60s	08:18:00 30s~60s
Oakleigh Park	07:20:00 30s~30s	07:50:00 30s~30s	08:20:00 30s~30s
New Southgate	07:23:00 30s~30s	07:53:00 30s~30s	08:23:00 30s~30s
Alexandra Palace	…	…	…
Hornsey	…	…	…
Harringay	…	…	…
Finsbury park	07:30:00 30s~60s	08:00:00 30s~60s	08:30:00 30s~60s
Holloway	…	…	…
Belle Isle	…	…	…
Midland Road Junction	07:34:00	08:04:00	08:34:00

Timetable 5: Kings Cross to Welwyn Garden City

	21	**22**	**23**	**24**	**25**	**26**
	S942/K1	**S29/K2**	**S43/K3**	**S20/H1**	**S30/H2**	**S42/H3**
	Class 91	Class 91	Class 91	Class 317	Class 317	Class 317
Kings Cross	07:00:00	07:30:00	08:00:00	07:03:00	07:33:00	08:03:00
Belle Isle	…	…	…	…	…	…
Holloway	…	…	…	…	…	…
Finsbury park	…	…	…	07:13:00 75s~90s	07:43:00 75s~90s	08:13:00 75s~90s
Harringay	…	…	…	…	…	…
Hornsey	…	…	…	…	…	…
Alexandra Palace	…	…	…	07:22:00 75s~90s	07:52:00 75s~90s	08:22:00 75s~90s
New Southgate	…	…	…	…	…	…
Oakleigh Park	…	…	…	…	…	…
New Barnet	…	…	…	…	…	…
Hadley Wood	…	…	…	…	…	…
Potters Bar	…	…	…	07:42:30 30s~60s	08:12:30 30s~60s	08:42:30 30s~60s
Brookmans Parks	…	…	…	…	…	…
Welham Green	...	...	...	…	…	…
Hatfield	…	…	…	07:52:30 30s~60s	08:22:30 30s~60s	08:52:30 30s~60s
Welwyn Garden City	07:18:00	07:48:00	08:18:00	07:56:00	08:26:00	08:56:00

Timetable 6: Moor Gate to Welwyn Garden City

	27	**28**	**29**
	S18/I1	**S31/I2**	**S44/I3**
	Class 313	Class 313	Class 313
Moor Gate	07:00:00	07:30:00	08:00:00
Old Street	07:02:30 30s~30s	07:32:30 30s~30s	08:02:30 30s~30s
Essex Road	07:06:00 30s~30s	07:36:00 30s~30s	08:06:00 30s~30s
Highbury & Islington	…	…	…
Drayton Park	07:09:00 30s~60s	07:39:00 30s~60s	08:09:00 30s~60s
Finsbury park	07:12:00 75s~90s	07:42:00 75s~90s	08:12:00 75s~90s
Harringay	07:15:00 30s~30s	07:45:00 30s~30s	08:15:00 30s~30s
Hornsey	07:16:45 30s~30s	07:46:47 30s~30s	08:16:47 30s~30s
Alexandra Palace	07:19:00 30s~60s	07:49:00 30s~60s	08:19:00 30s~60s
New Southgate	07:27:20 30s~30s	07:57:20 30s~30s	08:27:20 30s~30s
Oakleigh Park	07:31:00 30s~30s	08:01:00 30s~30s	08:31:00 30s~30s
New Barnet	07:33:00 30s~30s	08:03:00 30s~30s	08:33:00 30s~30s
Hadley Wood	07:36:00 30s~30s	08:06:00 30s~30s	08:36:00 30s~30s
Potters Bar	07:40:00 30s~60s	08:10:00 30s~60s	08:40:00 30s~60s
Brookmans Parks	07:43:00 30s~30s	08:13:00 30s~30s	08:43:00 30s~30s
Welham Green	07:45:30 30s~30s	08:15:30 30s~30s	08:45:30 30s~30s

Hatfield	07:49:30 30s~60s	08:19:30 30s~60s	08:49:30 30s~60s
Welwyn Garden City	07:53:00	08:23:01	08:53:01

Timetable 7: Moor Gate to Hertford

	30	**31**	**32**	**33**	**34**
	S19/J1	**S34/J2**	**S32/J3**	**S45/J4**	**S46/J5**
	Class 313	Class 313	Class 313	Class 313	Class 313
Moor Gate	07:05:00	07:20:00	07:35:00	07:50:00	08:05:00
Old Street	07:07:30 30s~30s	07:22:30 30s~30s	07:37:30 30s~30s	07:52:30 30s~30s	08:07:30 30s~30s
Essex Road	07:11:00 30s~30s	07:26:00 30s~30s	07:41:00 30s~30s	07:56:00 30s~30s	08:11:00 30s~30s
Highbury & Islington	…	…	…	…	…
Drayton Park	07:14:30 60s~60s	07:29:30 60s~60s	07:44:30 60s~60s	07:59:30 60s~60s	08:14:30 60s~60s
Finsbury park	07:18:00 75s~90s	07:33:00 75s~90s	07:48:00 75s~90s	08:03:00 75s~90s	08:18:00 75s~90s
Harringay	07:20:50 30s~30s	07:35:50 30s~30s	07:50:50 30s~30s	08:05:50 30s~30s	08:20:50 30s~30s
Hornsey	07:22:00 30s~30s	07:37:00 30s~30s	07:52:00 30s~30s	08:07:00 30s~30s	08:22:00 30s~30s
Alexandra Palace	07:25:10 30s~30s	07:40:10 30s~30s	07:55:10 30s~30s	08:10:10 30s~30s	08:25:10 30s~30s
Bowes Park	07:27:30 30s~30s	07:42:30 30s~30s	07:57:30 30s~30s	08:12:30 30s~30s	08:27:30 30s~30s
Palmers Green	07:29:40 30s~30s	07:44:40 30s~30s	07:59:40 30s~30s	08:14:40 30s~30s	08:29:40 30s~30s
Winchmore Hill	07:32:20 30s~30s	07:47:20 30s~30s	08:02:20 30s~30s	08:17:20 30s~30s	08:32:20 30s~30s
Grange Park	07:34:30	07:49:30	08:04:30	08:19:30	08:34:30

	30s~30s	30s~30s	30s~30s	30s~30s	30s~30s
Enfield Chase	07:37:00 60s~60s	07:52:00 60s~60s	08:07:00 60s~60s	08:22:00 60s~60s	08:37:00 60s~60s
Gordon Hill	07:40:30 60s~60s	07:55:30 60s~60s	08:10:30 60s~60s	08:25:30 60s~60s	08:40:30 60s~60s
Crews Hill	07:42:30 30s~30s	07:57:30 30s~30s	08:12:30 30s~30s	08:27:30 30s~30s	08:42:30 30s~30s
Cuffley	07:45:30 30s~30s	08:00:30 30s~30s	08:15:30 30s~30s	08:30:30 30s~30s	08:45:30 30s~30s
Bayford	07:51:30 30s~30s	08:06:30 30s~30s	08:21:30 30s~30s	08:36:30 30s~30s	08:51:30 30s~30s
Hertford	07:55:00	08:10:01	08:25:01	08:40:01	08:55:01

Timetable 8: Midland Road Junction to Welwyn Garden City

	35	**36**	**37**
	S17/G1	**S33/G2**	**S41/G3**
	TL 39	TL 39	TL 39
Midland Road Junction	07:02:00	07:32:00	08:02:00
Belle Isle	…	…	…
Holloway	…	…	…
Finsbury park	07:09:30 60s~210s	07:39:30 60s~210s	08:09:30 60s~210s
Harringay	…	…	…
Hornsey	…	…	…
Alexandra Palace	…	…	…
New Southgate	07:15:00 30s~30s	07:45:00 30s~30s	08:15:00 30s~30s
Oakleigh Park	07:18:00 30s~30s	07:48:00 30s~30s	08:18:00 30s~30s
New Barnet	07:20:30 60s~60s	07:50:30 60s~60s	08:20:30 60s~60s

Hadley Wood	07:23:10 30s~30s	07:53:10 30s~30s	08:23:10 30s~30s
Potters Bar	07:27:00 60s~60s	07:57:00 60s~60s	08:27:00 60s~60s
Brookmans Parks	…	…	…
Welham Green	…	…	…
Hatfield	07:33:30 60s~60s	08:03:30 60s~60s	08:33:30 60s~60s
Welwyn Garden City	07:37:00	08:07:00	08:37:00

APPENDIX D

PUBLICATIONS

[1] L. Dai, G. Nicholson, L. Chen*, et al.* (2013) "Visual Methods to Analyse Railway System Resilience", presented at IAROR, Copenhagen.

[2] M. Lu, G. Nicholson, F. Schmid, L. Dai, *et al.* (2013)"A Framework for the Evaluation of the Performance of Railway Networks", International Journal of Railway Technology, 2(2), 79-96.

[3] L. Dai, G. Nicholson, D. Kirkwood*, et al.* (2016) "Dynamic train rescheduling using alternating algorithm", accepted by ICIRT, Birmingham.

REFERENCE

[1] Department for Transport. (2013). "Transport Statistics Great Britain: 2013", *National Statistics*.

[2] R. Cohn and J. Russell. (2012). "Transport in the United Kingdom". VSD.

[3] S. Oskar, L. Felix, and G. Thomas. (2003). "Increasing Performance of the Rail Network in the Heart of Europe: A Program for the Swiss Federal Railways", proceedings of *International Symposium on Speed-up and Service Technology for Railways and Maglev Systems: STECH*, Tokyo, pp. 104-107.

[4] Network Rail Ltd. (2012). "Annual Report and Accounts: 2012. "

[5] Network Rail Ltd. (2013). "Network Rail Monitor, " *Office of Rail Regulation: ORR*.

[6] F. Corman, A. D'Ariano, M. Pranzo*, et al.* (2010). "Effectiveness of Dynamic Reordering and Rerouting of Trains in a Complicated and Densely Occupied Station Area", *Transportation Planning and Technology,* vol. 34, pp. 341-362.

[7] Y. H. Chou. (2009). "Distributed Approach for Rescheduling Railway Traffic in the Event of Distribution.", PhD Thesis, School of Electrinic, Electrical and Computer Engineering, University of Birmingham, Birmnigham. pp. 8-9.

[8] T. K. Ho and T. H. Yeung. (2001). "Railway junction traffic control by heuristic methods", *Electric Power Applications,* vol. 148, pp. 77-84.

[9] Y. V. Bocharnikov, A. M. Tobias, C. Roberts*, et al.* (2007). "Optimal Driving Strategy for Traction Energy Saving on Dc Suburban Railways", *IET Electric Power Applications,* vol. 1, pp. 675-682.

[10] P. Weston, C. Goodman, R. TAKAGI*, et al.* (2006) "Minimising Train Delay in a Mixed-Traffic Railway Network with Consideration of Passenger Delay", presented at *WCRR*, Montreal, Canada.

[11] F. Corman, A. D'Arianoc, D. Pacciarellic*, et al.* (2012). "Optimal Inter-Area Coordination of Train Rescheduling Decisions", *Transportation Research Part E: Logistics and Transportation Review,* vol. 48, pp. 71-88.

[12] V. Cacchiani, D. Huisman, M. Kidd*, et al.* (2014). "An Overview of Recovery Models and Algorithms for Real-Time Railway Rescedhuling", *Transportation Research Part B: Methodological,* vol. 63, pp. 15-37, May 2014.

[13] T. Siefer and A. Radtke. (2005). "Evaluation of Delay Propagation, " University of Hannover (IVE), Hannover, Germany.

[14] F. Barber, L. Ingolotiti, A. Lova*, et al.* (2009). "Meta-Heuristic and Constraint-Based Approaches for Single-Line Railway Timetabling", *Robust and Online Large-Scale optimization,* pp. 145-181.

[15] J. Demitz, C. Hübschen, and C. Albrecht. (2010). "Timetable Stability - Using Simulation to Ensure Quality in a Regular Interval Timetable", *WIT Transactions on State-of-the-art in Science and Engineering,* vol. 40, p. 14.

[16] L. P. Veelenturf, M. P. Kidd, V. Cacchiani*, et al.* (2014). "A Railway Timetable Rescheduling Approach for Handling Large Scale Disruptions. "

[17] D. Pacciarelli. (2013). "Introduction and Classification of Existing Rescheduling Literature", proceedings of *IAROR*, Copenhagen.

[18] F. Fischer, B. Grimm, and T. Klug. (2012). "Movement Planner Algorithm Design for Dispatching on Muli-Track Territories, " Chemnitz University of Technology, Department of MathematicsAugust 15, 2012.

[19] C. Yan and L. Yang. (2012). "Mixed-Integer Programming Based Approaches for the Movement Planner Problem: Model, Heuristics and Decomposition, " Tsinghua University, China.

[20] J. Törnquist. (2006). "Railway Traffic Disturbance Management", Department of Software Engineering School of Engineering, Blekinge Institute of Technology, SWEDEN

[21] T. Dollevoet, D. Huisman, M. Schmidt*, et al.* (2013). "Delay Management in Railway Operations", proceedings of *Innovative Approaches in Real-Time Railway Operations*.

[22] A. Schobel. (2002). "Integer Programming Models for the Delay Management Problem", ed: AMORE Research Seminar

[23] T. Dollevoet and D. Huisman. (2011). "Fast Heuristics for Delay Management with Passenger Rerouting, " Econometric Institute Report EI2011-35, Utrecht, The Neherlands.

[24] L. Gely, G. Dessagne, and C. Lerin. (2006). "Modelling Train Rescheduling with Optimization and Operational Research Techniques: Results and Apllications at Sncf, " SNCF innovation and Research department, Paris, France.

[25] M. Schachtebeck and A. Schöbel. (2010). "To Wait or Not to Wait—and Who Goes First? Delay Management with Priority Decisions", *Transportation Science,* vol. 44, pp. 307-321.

[26] R. M. Lusby, J. Larsen, M. Ehrgott*, et al.* (2013). "A Set Packing Inspired Method for Real-Time Junction Train Routing", *Computers and Operations Research,* vol. 40, pp. 713-724.

[27] M. Carey and A. Kwiecinski. (1994). "Stochastic Approximation to the Effectd of Headways on Knock-on Delays of Trains", *Transportation Research Part B: Methodological,* vol. 28, pp. 251-267.

[28] I. A. Hansen. (2004). "Increase of Capacity through Optimised Timetabling", *WIT Transactions on The Built Environment,* vol. 74, p. 10.

[29] Y. H. Chou, P. Weston, and C. Roberts. (2007). "Dynamic Distributed Control for Real-Time Rescheduling of Railway Networks", ed: International Seminar on Railway Operations Modelling and Analysis.

[30] J. Jacobs. (2008). "Chapter 11. Rescehduling", in *Railway Timetabling & Traffic: Analysis. Modelling. Simulation*.

[31] A. Mascis and D. Pacciareli. (2002). "Job-Shop Scheduling with Blocking and No-Wait Constriants", *European Journal of Operational Research,* vol. 143, pp. 498-517.

[32] M. Mazzarello and E. Ottaviani. (2007). "A Traffic Management System for Realtime Traffic Optimisation in Railways", *Transportation Research Part B: Methodological,* vol. 41, pp. 246-274.

[33] A. D'Ariano, D. Pacciarelli, and M. Pranzo. (2007). "A Branch and Bound Algorithm for Scheduling Trains on a Railway Network", *European Journal of Operational Research,* vol. 183, pp. 643-657.

[34] A. D'Ariano, F. Corman, D. Pacciarelli*, et al.* (2008). "Reorering the Local Rerouting Strategies to Manage Train Traffic in Real Time", *Transportation Science,* vol. 42, pp. 405-419.

[35] A. D'Ariano. (2009). "Innovative Decision Support System for Railway Traffic Control", *IEEE Intelligent Transportation Systems Magazine*. 8-16.

[36] F. Corman, A. D'Arianoc, D. Pacciarellic*, et al.* (2012). "Bi-Objective Conflict Detection and Resolution in Railway Traffic Management", *Transportation Research Part C: Emerging Technologies,* vol. 20, pp. 79-94.

[37] T. Dollevoet, F. Corman, A. D'Ariano*, et al.* (2013). "An Iterative Optimization Framework for Delay Management and Train Scheduling", *Flexible Services and Manufacturing Journal,* vol. 26, pp. 490–515.

[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest*, et al.* (2001). "Introduction to Algorithms, Second Edition". Cambridge , Massachusetts London, England McGraw-Hill Book Company

[39] A. Lew and H. Mauch. (2007). "Introduction to Dynamic Programming", *Studies in Computational Intelligence (SCI),* vol. 38, pp. 3-43.

[40] S. C. Ho and D. Haugland. (2011). "Local Serach Heruristic for the Probabilistic Dial-a-Ride Problem", *OR Spectrum: Quantitative Approaches in Management,* vol. 33, pp. 961-988.

[41] T. Albrecht and S. Oettich. (2002). "A New Integrated Approach to Dynamicschedule Synchronization and Energy-Saving Train Control", in *Computers in Railways Viii* vol. 13, pp. 847-856.

[42] S. Shan and P. S. Sastry. (1999). "New Algorithms for Learning and Pruning Oblique Decision Trees.", *IEEETransactions on Systems Man and Cybernetics Part CApplications and Reviews,* vol. 29, pp. 494-505.

[43] P. Weston, C. Goodman, R. Takagi*, et al.* (2006) "Minimising Train Delay in a Mixed-Traffic Railway Network with Consideration of Passenger Delay", presented at *7th World Congress on Railway Research (WCRR)*, Montréal, Canada.

[44] A. Ismail, M. Quafafou, G. Nachouki*, et al.* (2010). "A Decision Tree for Queries Routing in Hierarchical Peer-to-Peer Network", proceedings of *Informatics and Systems (INFOS), 2010 The 7th International Conference* Cairo, pp. 1-8.

[45] L. Gély, G. Dessagne, and C. Lerin. (2006). "Modelling Train Re-Scheduling with Optimization and Operational Research Techniques: Results and Applications at Sncf", proceedings of *7th World Congress on Railway Research (WCRR)*, Montréal, Canada.

[46] T. J. J. van den Boom and B. D. Schutter. (2006). "Dynamic Railway Network Management Using Switching Max-Plus-Linear Models", proceedings of *the11th IFAC Symposium on Control in Transportation Systems*, Delft, The Netherlands, pp. 343–348.

[47] L. Chen. (2012). "Real Time Traffic Management in Junction Areas and Bottleneck Sections on Mainline Railways", PhD Thesis, School of Civil

Engineering College of Engineering and Physical Sciences, The University of Birmingham, Birmingham, UK

[48] M. Boccia, C. Mannino, and I. Vasilyev. (2013). "The Dispatching Problem on Multitrack Territories: Heurstic Approaches Based on Mixed Integer Linear Programming", *Networks,* vol. 62, pp. 315-326, 1 October 2013.

[49] M. Boccia, C. Mannino, and I. Vasilyev. (2012). "Solving the Dispatching Problem on Multi-Track Territories by Mixed Integer Linear Programming", proceedings of *the RAS Competition 2012, INFORMS Meeting*.

[50] Y. Cheng. (1996). "A Network-Based Train Traffic Simulation with Changeable Rescheduling Strategies ", proceedings of *Second International Symposium on Parallel Architectures, Algorithms, and Networks*, Singapore, pp. 214-220.

[51] Q. Lu, M. Dessouky, and R. C. Leachman. (2004). "Modeling Train Movements through Complex Rail Networks", *ACM Transactions on Modeling and Computer Simulation,* vol. 14, pp. 48-75.

[52] J. Jacobs. (2004). "Reducing Delays by Means of Computer-Aided 'on-the-Spot' Rescheduling", in *Computers in Railway Vi* vol. 15, pp. 603-612.

[53] J. Jacobs. (2004) "Reducing Delays by Means of ’on the Spot’ Reschdeduling", presented at *Computers in Railways IX*, Dresden, Germany

[54] A. D'Ariano. (2008). "Improving Real-Time Train Dispatching: Models, Algorithms and Applications", Dissertation, Civil Engineering and Geosciences, University of Delft, The Netherland

[55] V. V. Breusegem, G. Campion, and G. Bastin. (1991). "Traffic Modeling and State Feedback Control for Metro Lines ", *IEEE TRANSACTIONS ON AUTOMATIC CONTROL,* vol. 36, pp. 770-784.

[56] P. Kecman, F. Corman, A. D'Ariano*, et al.* (2013). "Rescheduling Models for Railway Traffic Management in Large-Scale Networks", *Public Transport,* vol. 5, pp. 95-123.

[57] T. K. Ho, J. F. Norton, and C. J. Goodman. (1997). "Optimal Traffic Control at Railway Junctions", proceedings of *Electric Power Applications*, pp. 140-148.

[58] M. J. Dorfman and J. Medanic. (2004). "Scheduling Trains on a Railway Network Using a Discrete Event Model of Railway Traffic", *Transportation Research Part B: Methodological,* vol. 38, pp. 81-98.

[59] R. M. P. Goverde. (2010). "A Delay Propagation Algorithm for Large-Scale Railway Traffic Networks", *Transportation Research Part C: Emerging Technologies,* vol. 18, pp. 269-287.

[60] R. M. P. Goverde. (2007). "Railway Timetable Stability Analysis Using Max-Plus System Theory", *Transportation Research Part B: Methodological,* vol. 41, pp. 179-201.

[61] B. D. Schutter and T. van den Boom. (2001). "Model Predictive Control for Railway Networks", proceedings of *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM‘01)*, Como, pp. 105-110.

[62] B. D. Schutter, T. van den Boom, and A. Hegyi. (2002). "A Model Predictive Control Approach for Recovery from Delays in Railway Systems, " Delft University of Technology, Delft, The Netherlands.

[63] J. Törnquist and J. A. Persson. (2007). "N-Track Railway Traffic Rescheduling During Disturbances", *Transportation Research Part B: Methodological,* vol. 41, pp. 342-362.

[64] J. Törnquist. (2012). "Design of an Effective Algorithm for Fast Responde to the Rescheduling of Railway Traffic During Disturbance", *Transportation Research Part C: Emerging Technologies,* vol. 20, pp. 62-78.

[65] R. Acuna-Agost, P. Michelon, D. Feillet*, et al.* (2010). "A Mip-Based Local Search Method for the Railway Rescheduling Problem", *Networks,* vol. 57, pp. 69-86.

[66] Y. H. Min, M.-J. Park, S.-P. Hong*, et al.* (2011). "An Appraisal of a Column-Generation-Based Algorithm for Centralized Train-Conflict Resolution on a Metropolitan Railway Network", *Transportation Research Part B: Methodological,* vol. 45, pp. 409-429.

[67] C. Hirai and N. Tomii. (2007). "Train Rescheduling Algorithm Using Pattern Description Language and Integrated Algorithm Framework", proceedings of *The Eleventh IASTED International Conference on Artificial Intelligence and Soft Computing*, Anaheim, CA, USA, pp. 250-255.

[68] M. Missikoff. (1997) "An Object-Oriented Approach to an Information and Decision Support System for Railway Traffic Control", presented at *First International Conference on Knowledge-Based Intelligent Electronic Systems*.

[69] J. Barta, A. E. Rizzoli, M. Salani*, et al.* (2012). "Statistical Modelling of Delays in a Rail Freight Transportation Network", proceedings of *the 2012 Winter Simulation Conference*, pp. 1-12.

[70] A. R. Albrecht, D. M. Panton, and D. H. Lee. (2013). "Rescheduling Rail Networks with Maintenance Disruptions Using Problem Space Search", *Computers & Operations Research,* vol. 40, pp. 703-712.

[71] S. Dündar and İ. Şahin. (2013). "Train Re-Scheduling with Genetic Algorithms and Artificial Neural Networks for Single-Track Railways", *Transportation Research Part C: Emerging Technologies,* vol. 27, pp. 1-15.

[72] B. Fan. (2012). "Railway Traffic Rescheduling Approaches to Minimise Delays in Disturbed Conditions", PhD Thesis, Electrical & Electronic and Computer Engineering, The University of Birmingham,

[73] F. Moberg and S. H. Simonsen. (2011). *What Is Resilience? An Introduction to Social-Ecological Research*. Available: www.stockholmresilience.su.se

[74] ON-TIME (2012). "Ont-Wp01-D-Uob-025-01 - D1.1 - Agreed Principles, Definitions and Requirements".

[75] F. Barber and M. A. Salido. (2014). "Robustness, Stability, Recoverability and Reliability in Dynamic Constraint Satisfaction Problems", *Knowledge and Information Systems,* vol. 44, pp. 719-734.

[76] R. M. P. Goverde. (2008). "Chapter7, Timetable Stability Analysis", in *Railway Timetable & Traffic*ed.: Eurailpress.

[77] M. Carey and S. Carville. (2004). "Esting Schedule Performance and Reliability for Train Stations", *International transaction in operational research,* vol. 2, pp. 382-394.

[78] T. Yoko and T. Norio. (2005) "Robustness Indices for Train Rescheduling", presented at *1st International Seminar on Railway Operations Modelling and Analysis*, Delft, The Netherlands.

[79] M. A. Salido, F. Barber, and L. Ingolotti. (2008) "Robustness in Railway Transportation Scheduling", presented at *the 7th World Congress on Intelligent Control and Automation, WCICA 2008*, Chongqing.

[80] C. Liebchen, M. Lubbecke, R. Mohring*, et al.* (2009). "The Concept of Recoverable Robustness Linear Programming Recovery and Railway Application", *Robust and Online Large-Scale optimization,* vol. 5868, pp. 1-27.

[81] S. Cicerone and G. D. Angelo. (2009). "Recoverable Robustness in Shunting and Timetabling", in *Robust and Online Large-Scale Optimization*ed.: Springer Berlin Heidelberg.

[82] L. Dai, G. Nicholson, L. Chen*, et al.* (2013) "Visual Methods to Analyse Railway System Resilience", presented at *IAROR*, Copenhagen

[83] W. Weigand. (1981). "Verspatungubertragungen in Fernverkehrnetzen", *Eisenbahntechnische Rundschau ETR,* vol. 30, pp. 915-920.

[84] E. Muhlhans. (1990). "Berechnung Der Verspatungsentwicklung Bei Zugfahrten", *Eisenbahntechnische Rundschau ETR,* vol. 39, pp. 465-468.

[85] M. Carey and A. Kwiecinski. (1994). "Swapping the Order of Scheduled Services to Minimise Expected Costs of Delays", *Transportation Research Part B: Methodological,* vol. 28, pp. 409-428.

[86] A. Higgins and E. Kozan. (1998). "Modelling Train Delays on Urban Netwroks", *Transportation Science,* vol. 32, pp. 346-357.

[87] J. Yuan. (2006). "Stochastic Modelling of Train Delays and Delay Propagation in Stations", PhD Thesis, Netherlands TRAIL Research School, Delft University of Technology, The Netherlands

[88] E. R. Petersen and A. J. Taylor. (1982). "A Structured Model for Rail Line Simulation and Optimization", *Transportation Science,* vol. 16, pp. 192-206.

[89] M. M. Dessouky and R. C. Leachman. (1995). "A Simulation Modeling Methodology for Analyzing Large Complex Rail Networks", *Simulation,* vol. 65, pp. 131-142.

[90] H. Krueger. (1999) "Parametric Modeling in Rail Capacity Planning", presented at *the Winter Simulation Conference*, Phoenix, AZ.

[91] G. L. Nicholson, D. Kirkwood, C. Roberts*, et al.* (2015). "Benchmarking and Evaluation of Railway Operations Performance", *Journals of Rail Transport Planning & Management,* vol. 5, pp. 274-293.

[92] First Capital Connect. (11 December 2011 to 8 December 2012). "Train Times. "

[93] NetworkRail. (2010). "Route Plans 2010 - Route Plan G East Coast & North East."

[94] D. Kirkwood. (2015). "Simulation, Test and Performance Evaluation of Railway Control Strategies and Algorithms", School of Electronic, Electrical and Computer Engineering, University of Birmingham, Birmingham

[95] Y. V. Bocharnikov, A. M. Tobias, C. Roberts*, et al.* (2007). "Optimal Driving Strategy for Traction Energy Saving on Dc Suburban Railways", *IET Electronic Power Applications,* vol. 1, pp. 675–682.

[96] I. A. Hansen and J. Pachl. (2008). "Railway Timetable & Traffic", in *Analysis, Modelling, Simulation*. Hamburg, Germany. Eurailpress.

[97] F. Laube and H. Schaffer. (2006) "Specific Working Results About the Rescheduling Process and Phases for the Puls 90 Project", presented at *Internal working paper*, SBB Bern.

[98] M. Lüthi, A. Nash, and U. Weidmann. (2008) "Optimizing Traffic Flow in Heavily Used Railway Networks: Influence Factors and Potential Strategies", presented at *8th Swiss Transport Research Conference (STRC 2008)*, Ascona, Switzerland.

[99] A. Gendenitsch, S. Jakl, Y. Y. Chong*, et al.* (2004). "A Rule-Based Algortihm for Commom Pilot Channel and Antenna Tilt Optimization in Umts Fdd Networks", *ETRI Journal,* vol. 26, pp. 437-442.

[100] J. Daintith. (2004). "A Dictionary of Computing", E. Wright, Ed., 3 ed: Oxford University Press.

[101] N. Webster. (1913). "Webster's Revised Unabridged Dictionary". G. & C. Merriam Co.

[102] J. E. Mitchell. (1999). "Branch-and-Cut Algorithms for Combinatorial Optimization Problems", in *Handbook of Applied Optimization, Oxford University Press, 2000.* April 19, 1999 ed. Troy, NY, USA.

[103] J. Madar and J. Abonyi. (2005). "Genetic and Other Evolutionary Algorithms", in *Process Control and Optimization*ed. vol. 2, B. G. Liptak, Ed.: CRC Press Taylor & Francis Group.

[104] A. D'Ariano, D. Pacciarelli, and M. Pranzo. (2006). "A Branch and Bound Algorithm for Scheduling Trains in a Railway Network", *European Journal of Operational Research,* vol. 183 pp. 643-657.

[105] A. Levitin. (2007). "Chapter 3: Brute Force", in *Introduction to the Design & Anallysis of Algorithms*ed., n. edition, Ed.: Pearson Addison-Wesley.

[106] T. H. Cormen, C. E. Leiserson, R. L. Rivest*, et al.* (2009). "Chapter 15: Dynamic Programming", in *Introduction to Algorithms*. 3 ed. Cambridge, Massachusetts London, England: The Massachusetts Institute of Technology

[107] M. Dorigo. (1992). "Optimization, Learning and Natural Algorithms (in Italian)", PhD Thesis, Université Libre de Bruxelles p. 140.

[108] M. Dorigo and G. Caro. (1999). "Ant Colony Optimization: A New Meta-Heuristic", *IEEE*.

[109] M. Mitchell. (1998). "An Introduction to Genetic Algorithms". Massachusetts Institute of Technology.

[110] V. Maniezzo, T. Stützle, and S. Voß. (2010). "Matheuristics: Hybridizing Metaheuristics and Mathematical Programming", in *Annals of Information Systems*. Springer US.

[111] F. Glover. (1989). "Tabu Search—Part I", *INFORMS Journal on Computing,* vol. 1, pp. 190 - 206.

[112] F. Glover. (1990). "Tabu Search—Part Ii", *INFORMS Journal on Computing,* vol. 2, pp. 4-32, February 1, 1990.

[113] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. (1983). "Optimization by Simulated Annealing", *Science, New Series,* vol. 220, pp. 671-680.

[114] G. Woodbury. (2001). "An Introduction to Statistics", in *Available Titles Cengagenow*. 1 ed. Cengage Learning.

[115] S. A. Broverman. (2001). "Actex Study Manual, Course 1 Examination of the Society of Actuaries, Exam 1 of the Casualty Actuarial Society". 1 ed. vol. 1. Actex Publications.

[116] G. Jacobs. (2006). Ed.^Eds., *Book 2 - Eastern* (Railway Track Diagrams. Trackmaps, p.^pp. Pages.

[117] D. H. a. G. N. David Barney. (2001) "Calculating Train Braking Distance", presented at *6th Australian Workshop on Safety Critical Systems and Software (SCS'01)*, Brisbane.

[118] P. Tesi. (2009). "Switching Supervisory Control: Adaptive and Input-Constrained Systems", PHD Thesis, Dottorato di Ricerca in Ingegneria Informatica e dell' Automazione, Universita Degli Studi di Firenze,

[119] J. a. P. Hespanha. (2002) "Tutorial on Supervisory Control", presented at *Logic and Switching for the 40th Conferenc on Decision and Control*, Orlando, Florida.

[120] M. Fu. (1996) "Minimum Switching Control for Adaptive Tracking", presented at *35th Conference on Decision and Control*.

[121] M. G. Safonov and T. C. Tsao. (1994). "The Unfalsified Control Concept and Learning ", proceedings of *the 33rdConference on Decision and Control* Lake Buena Vista, FL pp. 2819-2824.

[122] E. S. d. Oliveira. (2001). "Solving Single-Track Railway Scheduling Problem Using Constraint Programming", PhD Thesis, School of Computing, The University of Leeds, Leeds

[123] L. Kang and G. M. White. (1992). "A Logic Approach to the Resolution of Constraints in Timetabling", *European Journal of Operational Research,* vol. 61, pp. 306-317, 25 September 1992.

[124] J. d. Yang, M. N. Huhns, and L. M. Stephens. (1985). "An Atchitecture for Control and Communications in Distributed Artificial Intelligence System", *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS,* vol. 15.

[125] Weiming Shen, Douglas H. Norries, and J.-P. A. Barthes. (2001). "Multi-Agent System of Concurrent Intelligent Design and Manufacturing". London. Taylor & Francis.

[126] M. M. Dessouky and S. Mu. (2011). "Dynamic Scheduling of Trains in Densely Populated Congested Areas, " Daniel J. Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, California.

[127] R. Reddy, L. Erman, R. Fennel*, et al.* (1976). "The Hearsay Speech Ubderstanding System: An Example of the Recognition Process", *IEEE Transactions on Computers,* vol. C, pp. 427-431.

[128] W. Shen, D. H. Norrie, and J.-P. A. Barthes. (2001). "Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing". London. Taylor & Francis, p. pp34.

[129] G. Vernazza and R. Zunino. (1990). "A Distributed Intelligence Methodogy for Railway Traffic Control", *IEEE Transactions on Vehicular technology,* vol. 39, August 1990.

[130] H. proenca and E. Oliveira. (2004). "Marcs Multi-Agent Railway Control System", in *Advances in Artificial Intelligence – Iberamia 2004*ed.: Springer Berlin Heidelberg, pp. 12-21.

[131] GRAFFICA Ltd. "Hermes Simulator". Available: http://graffica.co.uk/rail-traffic-management/simulation/

[132] E. Quaglietta, P. Pellegrini, R. M. P. Goverde*, et al.* (2015). "The on-Time Real-Time Railway Traffic Management Framework: A Proof-of-Concept Using a Scalable Standardised Data Communication Architecture", *Transportation Research Part C: Emerging Technologies,* vol. 63, pp. 23-50, 29 December 2015.