
# RAILWAY TRAFFIC RESCHEDULING APPROACHES TO MINIMISE DELAYS IN DISTURBED CONDITIONS

by

**BO FAN**

A thesis submitted to

The University of Birmingham

for the degree of

**DOCTOR OF PHILOSOPHY**

School of Electronic, Electrical and Computer Engineering

College of Engineering and Physical Sciences

The University of Birmingham, UK

August 2012

# Abstract

The advent of modern railway signalling and train control technology allows the implementation of advanced real-time railway management. Optimisation algorithms can be used to: minimise the cost of delays; find solutions that recover disturbed scenarios back to the operating timetable; improve railway traffic fluidity on high capacity lines; and improve headway regulation. A number of researchers throughout the world have previously considered the problem of minimising the cost of train delays and have used various optimisation algorithms for differing scenarios. However, little work has been carried out to evaluate and compare the different approaches.

Firstly, the author of this thesis compares and contrasts a number of optimisation approaches that have been used previously and applies them to a series of common scenarios. The approaches considered are: brute force, first-come-first-served, Tabu search, simulated annealing, genetic algorithms, ant colony optimisation, dynamic programming and decision tree based elimination. It has been found that simple disturbances (i.e., one train delayed) can be managed efficiently using straightforward approaches, such as first-come-first-served. For more complex scenarios, advanced methods are found to be more appropriate. For the scenarios considered in this comparison, ant colony optimisation performed well: the delay cost is decreased by 30% compared with first-come-first-served.

Secondly, in order to improve the currently available algorithm so that it can find more reliably optimal or close to optimal results within a practical computation time,

a new hybrid algorithm, has been developed based on ant colony optimisation. In order to evaluate the new approach, 100 randomly generated delay scenarios were tested, and a comparison is made between the results of the new algorithm and first-come-first-served, brute force and standard ant colony optimisation. It is shown that the hybrid algorithm has an improved performance in terms of optimality and computation speed.

Finally, a new multi-stage rescheduling approach for finding an optimal solution over multiple junctions is proposed. A case study is considered, and it is shown that the proposed approach performs well.

# Acknowledgements

I am extremely grateful to my supervisor, Prof. Clive Roberts, for his continuous support, guidance, and patience. I have greatly benefited from his profound knowledge, plentiful experience, and insightful discussions. I much appreciate his encouragement and he provided me with many helpful suggestions and important advice during my research and academic writing.

I would also like to extend my deep gratitude to Dr Paul Weston for his helpful suggestions, kind support and inspiration for this thesis and publications as well as the challenging research that lies behind it.

Thanks also to the members of the Birmingham Centre for Railway Research and Education for supporting and helping daily research work. Let me also say thank you to Katherine Slater for proof reading my thesis.

Finally, I would like to thank my parents and my wife. Without their love, support and encouragement, my study in the UK would not have been possible.

# Table of Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| ACO | Ant Colony Optimisation |
| ATP | Automatic Train Protection |
| BF | Brute Force |
| DP | Dynamic Programming |
| DTBACO | Decision Tree Based Ant Colony Optimisation |
| DTBE | Decision Tree Based Elimination |
| FCFS | First-Come-First-Served |
| GA | Genetic Algorithms |
| GSM-R | Global System for Mobile communication - Railway |
| JSP | Job Shop Problem |
| NQP | N-Queens Problem |
| OMJR | Optimal Multi-Junction Rescheduling |
| OSJR | Optimal Single-Junction Rescheduling |
| P2P | peer-to-peer |
| QAP | Quadratic Assignment Problem |
| RB | Rule Based |
| RTC | Railway Traffic Controller |
| SA | Simulated Annealing |
| SST | Stage-to-Stage Transformations |
| TS | Tabu Search or Taboo Search |
| TSP | Travelling Salesman Problem |

# Chapter 1: Introduction

## 1.1 Background

Railway timetabling and rescheduling play a central role in day-to-day railway operations. Trains on a railway network are scheduled and controlled according to a timetable. Timetables are designed to be conflict free, that is, they should not contain any situations where a train is restricted in its scheduled movement by another train. However, in practice not all trains run according to the timetable due to delays such as excessive dwell times at stations, infrastructure and/or train faults, and the late arrival of crew. When trains do not operate according to the timetable, even by only a few seconds, there is an increased likelihood that they will cause conflicts with other trains, resulting in those trains also being delayed. Railway operators therefore attempt to run trains to timetable, or, failing this, they try to minimise the cost of delays.

Generally, in today's railways, most train control is carried out by human operators – signallers. They are able to control the operation of signals to reschedule and prioritise traffic flows. In simple scenarios, signallers are able to manage the flow of traffic effectively. However, it has been shown that, as situations become more complex, there is an increased likelihood of signallers making sub-optimal traffic management decisions. This is particularly likely in significantly disturbed situations (Balfe *et al.*, 2007).

In the UK, 49% of trains arrive at their destination 'on time' i.e., on or before their scheduled timetable time (Best and Hyland, 2012). This means that 51% of trains are late. Signallers need support in deciding upon the most appropriate train rescheduling (train speed and sequences) to minimise the potential for further delays.

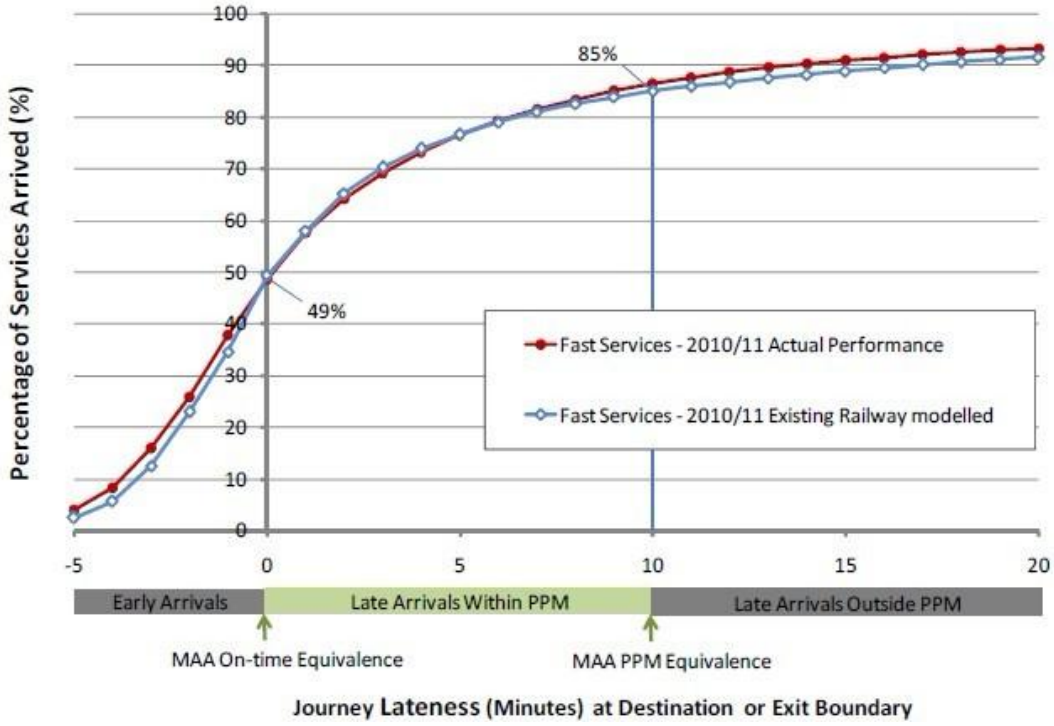


Figure 1-1: Statistics of train lateness in 2010/11by Network Rail (Best and Hyland, 2012)

In recent years, railway operators have sought to find technology based solutions that can help signallers make improved decisions. Such decision support or automatic control systems have been deployed on many networks (Stolk, 1998). In general they rely on straightforward heuristic algorithms. These simple algorithms are able to provide useful solutions in many cases, but, as situations become more complex, they do not perform well (Albrecht, 2009). Therefore, many researchers throughout the world have considered the use of more advanced algorithms as part of real-time optimisation systems for railway traffic management.

## 1.2 Introduction to Railway Traffic Management

Railway traffic management involves three basic elements: the railway timetable, the signalling system and the train control system. The principles of these elements are introduced in this section.

### 1.2.1 Brief Introduction to Railway Timetabling

A railway timetable should perform five main functions (Hansen and Pachl, 2008):

1. Plan the train paths and the use of infrastructure;
2. Give information to passengers and facilitate the sale of tickets;
3. Avoid train traffic conflicts and increase the traffic capacity;
4. Provide arrangements for traffic control, locomotive and rolling stock usage and crew scheduling;
5. Provide sufficient time for energy efficient driving by the driver or for automatic train operation.

Planning the train paths and the use of infrastructure must consider the track itself, junctions and platforms in stations. A timetable ought to provide conflict-free movement at all junctions which the trains need to pass. Timetables should be designed to maximise the traffic capacity without conflicts. Rolling stock, train crew and signallers also need to be scheduled effectively.

### 1.2.2 Railway Signalling System

The railway signalling system is an important element of the railway which ensures the safety of trains. Trains travel along track to reach their destination, with their movement controlled by the signalling system. The railway track is often split into a

series of sections, known as block sections, which normally require there to be only one train in a block section to ensure that trains remain a safe distance apart. The signalling system protects trains from colliding by using block sections. Otherwise, more than one train could pass through a block section at the same time. The signalling system also controls switches (or points), where a single track splits into two tracks, or two tracks merge into a single track.

### 1.2.3 Railway Train Control System

Train control systems generally contain one or both of the following function subsystems:

1. An automatic train control system (as well as automatic train protection and automatic train operation), which uses intermittent communication via Eurobalise or continuous communication via Global System for Mobile communication - Railway (GSM-R) to locate trains and provide advice on driving speed. This technology can be used to replace fixed block sections and line side signalling (Theeg and Vlasenko, 2009). Using the automatic train control system, a 'moving block' strategy is used which is able to provide dynamic control to the railway system and thus increase capacity.

2. A train rescheduling or automatic route setting system that either controls (regulates) the speed of trains through junctions to minimise conflicts (Mazzarello and Ottaviani, 2007) or helps signallers decide on an appropriate sequence of trains through junctions (Kuhn, 1998).

As discussed previously, the train timetable is designed to be conflict-free, with the initial sequence of trains through junctions known. However, when trains do not operate according to the timetable, conflicts can arise. In this scenario a signaller must provide a new train time plan and possibly changes to the order of trains through some junctions. Once a train's journey is disturbed, it may not be able to regain the lost time. Signallers, therefore, are not only responsible for conflict-free rescheduling, but also try their best to minimise the delay loss from timetable disturbances.

It is impossible for signallers to give an optimal rescheduling solution to reach the objectives by using their expertise alone, since an optimal solution requires a large amount of computation. A train rescheduling system (also called decision support system) can be used to assist with the rescheduling task, as shown in Figure 1-2. The details of a disturbance detected by the train control system must be sent to the rescheduling system so that an optimal rescheduling solution can be found. This must be returned within a limited time to enable a new control strategy (sequence, dispatch times, timing points) to be put in place.

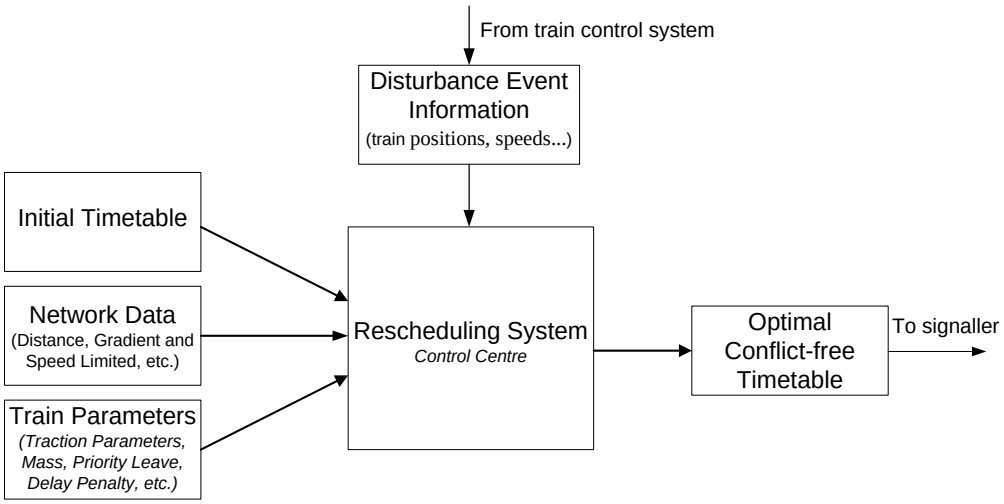


Figure 1-2: Train rescheduling system (Fan)

Train control is implemented in one or more control centres, an example of which is shown in Figure 1-3. In the UK, Network Rail has a number of control centres which are responsible for control regions (Gibson *et al.*, 2002). By contrast, Sweden has only one control centre which is responsible for the whole country's network (Lawson *et al.*, 2001).

Figure 1-3: Example of a railway control centre

## 1.3 The Purpose of Train Control and Rescheduling Systems

In order to find an optimal solution, railway traffic control has to have a rescheduling 'objective' for a particular network, for example, minimising overall delay, emphasising train connections on an airport express line, or minimising journey times on metros. There are six common objectives of timetable rescheduling that have been considered in previous research:

- To reduce the sum of all delays of all the considered trains (Ho and Yeung, 2001);
- Minimise the number of delayed passengers / the delay minutes per passenger (Weston *et al*., 2006);
- To minimise energy consumption (Bocharnikov *et al*., 2007);
- Emphasise train and other connections for passengers who need to change train or mode of transport (Albrecht, 2004);
- To decrease the total delay cost, where the delays of different train types have different weights (Chou *et al.*, 2007);
- To decrease the delay cost of the most delayed train (Corman *et al.*, 2012).

In order for any of the rescheduling approaches to attempt to find an optimal solution in the control centre, there must be some way of modelling the set of possible solutions to the train rescheduling optimisation problem, such that the most appropriate ones may be selected. To be able to select the best solution, a function that measures the quality of possible solutions, the 'cost function', must be formulated. The model format must therefore contain the information which informs the signaller

of the sequence in which the approaching trains must be rescheduled through the junction area.

Consider how a junction conflict is actually resolved: when a queue forms at a junction trains are rescheduled from the junction inputs until there are no longer any trains waiting at the inputs to the junction. It can be inferred that for a given starting set of trains at a junction, there is a finite number of train sequences.

## 1.4 Motivation and Research Objectives

The motivation for the research is to provide signallers with better decision support than is currently available in operational control systems. In this thesis, existing algorithms for optimal train rescheduling are reviewed and a new algorithm is developed. In which rescheduling approaches discussed in this thesis, only deal with timetable disturbances and does not include rerouting of trains.

In certain circumstances, if only one train is delayed, and only a small local area is considered, train rescheduling can be a straightforward problem. However, as the number of trains and the geographic area under consideration increase, the problem becomes more complex. Furthermore, each of the trains may be of a different type (high speed, commuter or freight). This means that each train will have different accelerating and braking rates and a different top speed. The problem of deciding upon an optimal solution then becomes even more complex, even for a simple scenario.

There are many different algorithms that can be used for optimisation; each has its own strengths and weaknesses. For real-time train control, in practice, there is a trade-

off between computation time and the identification of an optimal solution, i.e., the best possible solution. It is not straightforward to choose the most suitable algorithm to reduce the delay. The early chapters of this thesis therefore aim to compare and discuss a number of potential algorithms suitable for real-time railway control, in terms of required computation time and solution optimality. Eight optimisation algorithms are selected for testing on four railway rescheduling benchmark scenarios.

Based on a comparison and discussion of the algorithms' implementation, the strengths and weaknesses of these algorithms are concluded with respect to the railway rescheduling problem. The approach with the best performance is selected for further study. In order to have a better resolution in a shorter computing time, further research and analysis is carried out on the selected approach.

Finally, a new approach to optimal multi-junction rescheduling for a control region is developed and tested, and then refined.

For all of the rescheduling approaches discussed in this thesis, it is very important to estimate the running time as accurately as possible. Therefore, a multi train simulator is designed and built is this thesis. Since a significant amount of work was conducted on this simulator, the concepts and functions are introduced in Section 2.5.3 and the techniques and the equations are illustrated in detail in Appendix A. In addition, the rescheduling solutions are further examined by means of the Railway Traffic Controller (RTC) software which is the American standard software to plan and control railway traffic.

## 1.5 Thesis structure

- Chapter 1 introduces the background, research motivation, research objective, railway timetabling, signalling, traffic control and the structure of this thesis.
- Chapter 2 classifies and reviews related literature of railway real-time rescheduling and introduces benchmark scenarios for testing these reviewed algorithms and the cost function of this research.
- Chapter 3 details the use of fundamental and graphical algorithms in benchmark scenarios in a single junction case study.
- Chapter 4 shows the application of evolutionary algorithms for the same benchmark scenarios in a single junction case study, and then the performance and results are discussed.
- Chapter 5 introduces a new approach to single junction rescheduling using the same case study.
- Chapter 6 describes the design and application of a multi junction rescheduling approach.
- Chapter 7 presents conclusions and comments on further work.

# Chapter 2: Review of Algorithms for Railway Rescheduling and the Development of a Benchmark Scenario for Algorithm Comparison

This chapter reviews the application of optimisation algorithms to railway rescheduling and related areas. Researchers have applied a variety of optimisation algorithms to different case studies to evaluate their approaches. In Chapters 3 and 4, therefore, a benchmark scenario for the single junction case is tested and compared.

## 2.1 High Level Partition of Rescheduling Optimisation Algorithms

In this thesis, the process of optimisation is considered to involve finding the best practical solution to a problem given a number of constraints, e.g. the maximum speed of the trains, the required distances between trains, etc.. The definition of 'best' is defined by a cost function, which will be discussed later in Section 2.5.4.

In railway rescheduling a variety of optimisation algorithms have been applied, namely: Brute Force (BF), First-Come-First-Served (FCFS), Rule Based Method (RB), Dynamic Programming (DP), Decision Tree Based Elimination (DTBE), Tabu

Search (TS), Simulated Annealing (SA), Genetic Algorithms (GA) and Ant Colony Optimisation Algorithm (ACO). Table 2-1 shows a high level partitioning of the optimisation algorithms for railway rescheduling considered in this thesis. The algorithms can be divided into broadly three sets: fundamental methods, graph-based methods and evolutionary methods.

| **Fundamental Algorithms** | **Graph-based Algorithms** | **Evolutionary Algorithms** |
|---|---|---|
| BF | DP | SA |
| FCFS | DTBE | GA |
| RB | ACO | TS |

Table 2-1: High level partition of the optimisation algorithms

A summary of research work in a variety of optimisation application areas is shown in Table 2-2. The table is split into two main areas:

(1) Railway Rescheduling Optimisation, which has two categories: (a) references where the objective has been to minimise the delay; and (b) references where the objective has been to optimise some other operational parameter, for example, minimising the traction energy consumption or maximising traffic throughput.

(2) General Optimisation, which also has two categories: (a) classical problems, which includes references to problems commonly considered in optimisation literature. Since railway timetable rescheduling can be understood as a mathematical combinatorial problem, it is useful to undertake a review of and draw lessons from this type of work. Examples are: Travelling Salesman Problem (TSP) (Padberg and Rinaldi, 1987), Quadratic Assignment Problems (QAP) (Gambardella *et al.*, 1999),

Job Shop Problem (JSP) (Mascis and Pacciarelli, 2002a) and N-Queens Problem (NQP) (Sosic and Gu, 1994). (b) Relevant problems that are similar in formation to the railway rescheduling problem are also considered in this thesis.

| **Applications \ Algorithms** | **Railway Rescheduling Optimisation** | | **General Optimisation** | |
|---|---|---|---|---|
| | **(a) Objective to minimise delay** | **(b) General operational objectives** | **(a) Classical problems** | **(b) Relevant problems** |
| **BF** | (Cheng, 1998a) | | (Ciancarini and Favini, 2010) | (Kodeekha, 2007) |
| **RB** | | (Fay, 2000) | | |
| **FCFS** | | (D'Ariano, 2008) | | (Schmid and Blieberger, 1992) |
| **DP** | (Ho *et al.*, 1997) | | (Lew and Mauch, 2007) | (Albrecht and Oettich, 2002) |
| **DTBE** | (Weston *et al.*, 2006) | (D'Ariano *et al.*, 2007a) | (Lopez and Tunon, 2005) | (Goossens *et al.*, 2004) |
| **TS** | (Ho and Yeung, 2001) | (Corman *et al.*, 2009) | (Pham and Karaboga, 2000) | (Qi *et al.*, 2008) |
| **SA** | (Tornquist and Persson, 2005) | | (Zheng *et al.*, 2006) | (Tornquist and Persson, 2005) |
| **GA** | (Ho and Yeung, 2000) | (Wegele *et al.*, 2008) | (Rothlauf, 2006) | (Wanner *et al.*, 2007) |
| **ACO** | (Fan *et al.*, 2011) | (Zidi and Maouche, 2006) | (Dorigo and Stützle, 2004) | (Albrecht, 2009) |

Table 2-2: The key applications of optimisation algorithms

## 2.2 Fundamental Algorithms

Fundamental algorithms are straightforward to implement. Three such algorithms are considered here: BF, RB and FCFS.

Brute force (sometimes known as exhaustive search or enumeration) evaluates every valid sequence of trains. BF will therefore always find the optimal solution. For simple problems this is an effective approach, however, as the number of potential solutions increases, so does the computational burden, hence increasing the time required to find the solution.

An application of BF is discussed by Cheng (1998a), who introduces a simulation-based train traffic rescheduling approach to resolve timetable conflicts. The conflict-free solution is found using the BF method after all possible solutions have been simulated. Kodeekha (2007) has used the BF method to solve the JSP, which is similar in nature to that of railway rescheduling.

For solving very complex optimisation problems within a limited time, such as these occupy in railway rescheduling, Rule Baed (RB) or knowledge-based approaches are adopted in order to find a solution quickly. Such RB approaches are similar in nature to the decisions carried out by a human controller in a control centre (Peng and Ying, 2008). Normally, the rules are constructed from previous experience, business case driven decisions or an abstraction of a mathematical model. Commonly used rules include: retaining the train sequence of the timetable and allowing the most recent train to proceed through a junction first. This latter approach is used by the Railway Traffic Controller (RTC) which is used by rail freight marshalling in North America (Dingler, 2008).

Cheng (1996) presents a RB method that uses a train's predicted time to destination to determine which train should pass through a junction first. This approach provides

consideration of a train's whole journey, rather than just the train's time to pass through a junction. Chiang (1998) further developed this RB method by developing a more precise train simulation model. Fay (2000) proposes a fuzzy RB system where trains are rescheduled at junctions by a decision support system based at the control centre.

FCFS is a simple 'rule-of-thumb' approach based on an *a priori* understanding of the railway system. A simple heuristic rule is applied that states that the first train to approach the junction area should be allowed to pass first. FCFS can be considered as one of the RB methods, but since it is a method used common in railway operations, it is introduced separately here. FCFS is always computationally fast, but it will generally not find the optimal solution. FCFS is particularly ineffective at finding solutions in complex scenarios where multiple trains are experiencing minor delays (D'Ariano *et al.*, 2007a). Researchers usually compare the results of their methods with FCFS to show how much of an improvement their studies could make, for example (D'Ariano, 2008) and (Chen *et al.*, 2010).

Fundamental algorithms provide a straightforward technique for solving very complex problems in railway rescheduling. BF will always provide the optimal solution but after significant computation time. Other fundamental algorithms cannot guarantee the optimality of the solution, although they have shorter computation times.

## 2.3 Graph-based Algorithms

Graph-based algorithms attempt to organise the solution space to provide a structured way of searching for solutions. Three main transformations are used: Stage-to-Stage Transformations (SST), Decision Trees (DT) and Alternative Graphs.

### 2.3.1 Stage-to-Stage Transformations

A SST is created to model a shortest path problem which is used to find the shortest distance between two localities, which means the lowest cost in railway rescheduling problems. DP is one of the most common shortest path algorithms and it uses SST (Lew and Mauch, 2007). As shown in Figure 2-1, the railway junction problem can be structured as a series of 'Stages'. Starting at Stage 0, each subsequent stage shows the possible next states (where one state is one sequence, or partial sequence, of trains). The links are each given a value to enumerate the cost of moving from one stage to the next. The optimal solution can be found by finding the lowest cost (i.e., shortest path) of moving from the first to the last stage (Cormen *et al.*, 2001). SST generates all valid solutions, and therefore, the optimal solution can always be found. DP is computationally more efficient than the BF algorithm, as solutions are decomposed into a series of interconnected states that can be evaluated sequentially. The optimal solution is found by undertaking a critical path analysis; details of this technique can be found in Lockyer and Gordon (1991).

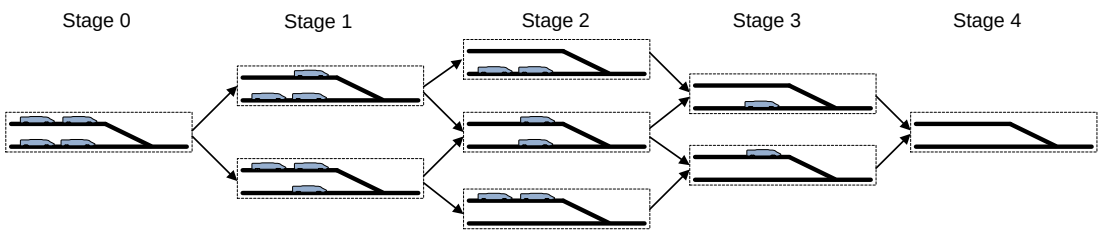


Figure 2-1*:* Example of Stage-to-Stage transformation model

Ho and Haugland (2011) introduce a traffic controller which is based on SST and DP. This traffic controller uses a cost function based on total weighted train delay to determine an optimal train sequence.

DP can also be used in dynamic schedule synchronisation for connecting the railway with other public transport (Albrecht and Oettich, 2002). The travel time between each station is structured as a multi-stage process. The cost function is considered to be the passenger waiting time between services.

### 2.3.2 Decision Trees

Decision trees are commonly used in operations research to structure the solution space. As shown in Figure 2-2, the root of the inverted tree is the state where no trains have passed the junction. The first level of branches represents all possible first trains through the junction. Subsequent branches represent successive trains. The leaves of the tree show all valid solutions. A leaf node represents an intermediate or final state, when some or all of the trains have passed the junction.

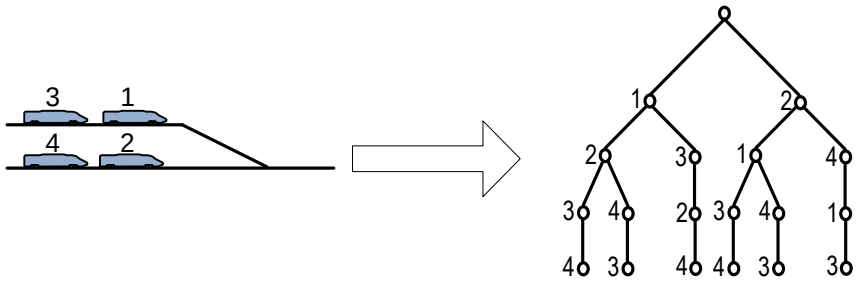


Figure 2-2: Example of a decision tree

Depending on the optimisation algorithm, more or less of the tree will be generated. For example, a decision tree is generally used to identify valid solutions for the Brute Force algorithm. In this case the whole tree is generated and every node on the tree is evaluated. Conversely, Shah and Sastry (1999) propose a pruning technique for decision trees. The nodes are evaluated by a function and if they do not meet a predetermined threshold the branch is not grown further. Such an approach requires less computation time, but increases the risk of not finding the optimal solution.

DT is widely used in similar routing problems, such as internet routing problems. Ismail *et al.* (2010) propose a DT to improve the peer-to-peer (P2P) search performance, i.e., which resources on the internet should be connected to one another. The destination of routing queries is modelled as a DT; the solution is found from the DT model within a practical response time.

In a railway rescheduling study, DT was used to minimise the train delays on a mixed traffic railway network (Weston *et al.*, 2006). In this study, the measure of passenger delay minutes is used as the cost function.

The branch and bound algorithm, which is one of the developed DT methods, has also been considered for train rescheduling by D'Ariano *et al.* (2007a). The authors apply a branch and bound algorithm to reschedule a bottleneck area of the Dutch rail network, in a simple two train problem with an alternative graph formulation. The results give proven optimal or near optimal solutions within short time limits. A related algorithm, branch and cut, is normally used for searching for optimal or near-optimal solutions by bounding or cutting some of the possible branches (Jünger, 2001). In the railway research domain of strategic phase in the planning process of a railway operator, Goossens *et al.* (2004) propose to use branch and cut to solve railway line planning problems in order to reduce the total operating cost.

The application of ACO in this thesis is also based on decision trees. ACO emulates ants searching their territory for food. Solutions can be evaluated by determining the "pheromone level" of a path, where high pheromone levels are found on paths that have been passed by many ants. ACO can also be classified as an evolutionary algorithm.

### 2.3.3 Alternative Graph Model

The alternative graph model is designed to ensure that task (or job) orders meet certain constraints. This is commonly referred to as the job shop problem (Mascis and Pacciarelli, 2002a), which is similar to the railway timetable rescheduling problem. Mascis and Pacciarelli (2002b) have developed this model to solve railway traffic management problems. Later, D'Ariano (2008) developed the model further by using a number of different evolutionary algorithms.

In a railway timetable rescheduling application, the alternative graph is based on the original track map, as shown in Figure 2-3. The station to station journey can be divided into many block sections and the running times to pass different sections are fixed. In each section, trains have two options: stay in the current section or go to the next section. For a conflict-free journey, the journey shown in Figure 2-3 would be 1-2-3 and the journey time would be the sum of the fixed running time of these three sections. However, if there is a conflict in section 3, the journey would be adjusted by repeating one of the sections such as: 1-1-2-3 or 1-2-2-3 *etc*.

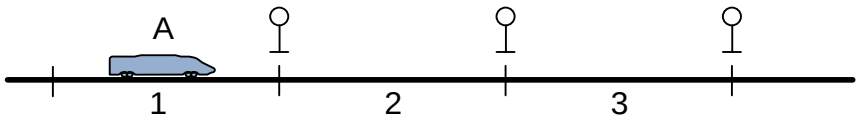


Figure 2-3: Alternative graph model with one train

For example, as shown in Figure 2-4, there is a conflict between train A and train B if they run as normal, A1-A2-C1-C2 and B1-B2-C1-C2.

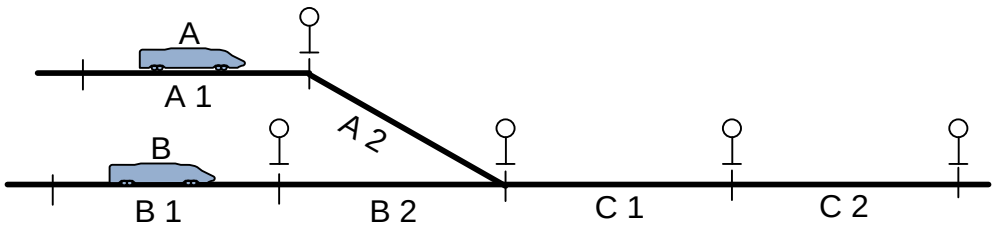


Figure 2-4: Alternative graph model with two trains

An adjustment policy (algorithm) must be applied here to decide which train may pass into block sections C1 and C2 first. However, all studies which are based on this alternative graph model simply provide a valid solution, which is unlikely to be the optimal solution with minimal delays. The reason for this is that the alternative graph

model is focused on the block sections but not the journey time. Furthermore, it is difficult to simulate a large number of trains in a network using the alternative graph model. Therefore, the alternative graph model is not considered further in this study.

## 2.4 Evolutionary Algorithms

Traditional algorithms can be inefficient in complex scenarios because they evaluate all possible solutions, resulting in significant computational times. Evolutionary algorithms have been created to improve computation times. The operation of evolutionary algorithms is based on behaviours inspired by nature. They find solutions through iterative improvement. Generally, an optimal solution will be found eventually; however, in practice, solutions must be found within a limited time, so final solutions are often sub-optimal.

There are many different types of evolutionary algorithm. Tabu Search (TS), Simulated Annealing (SA) and Genetic Algorithms (GA) are considered in this thesis. TS is a variation of the local search method (Reeves, 1993) that includes a heuristic rule that changes the search area when a local minimum is found, making it particularly good for combinatorial problems. SA initially evaluates solutions in a wide search area; as candidate solutions are identified, the search area is refined. GAs mimic biological evolution; each solution (gene) is changed to find better solutions through a set of predetermined functions (cross-over, mutation and selection). Although ACO could also be treated as an evolutionary algorithm, in this thesis it has been used as a tree based method.

Some of the evolutionary algorithm application studies are very relevant to this research. For example, Qi *et al.* (2008) describes an application of a TS algorithm for solving a vehicle routing problem.

D'Ariano *et al.* (2007b) present the idea of using TS coupled with an alternative graph model to solve conflicts by adjusting train speeds. A central control centre communicates with trains that will potentially conflict and suggests revised speed profiles for them to avoid conflict. The rescheduling regime uses TS to find optimal speeds for trains based on the standard time for a train to pass a signalling block. However, this train speed adjustment approach can only solve simple problems since it is based on the alternative graph model. In addition, Corman *et al.* (2009) further developed the speed adjustment approach, proposing a new method called green wave, which lets trains wait at stations, rather than wait at junctions. This approach is analysed using a busy Dutch railway network, and is shown to improve capacity. Furthermore, Corman *et al.* (2010) used TS in their real-time railway traffic controller on parts of the Dutch railway network. The results concluded that an optimal solution was found in a small number of instances, however, a close to optimal solution was found in the majority of cases.

Brown *et al.* (1992) applied both SA and GA to determine, off-line, train routes and train paths over a freight rail network, and concluded that SA is better than GA in the same case study, when computation time is not considered. Isaai and Singh (2001) used TS and SA in a railway timetabling simulation study for passenger trains of Iran’s railway system. The authors show the success of the generated schedules which

outperforms the manual timetabling method. Tornquist and Persson (2005) also tested two algorithms, SA and TS, by running the two algorithms in parallel to determine solutions for railway rescheduling. The best solution was selected from the two algorithms; it was found that TS outperformed SA most of the time.

Applications in real-time railway rescheduling of GA are introduced by Ho and Yeung (2000). In their studies, all trains are assumed to be of the same type. The conflict is resolved at the junction by assigning a right-of-way sequence, with the objective of minimising the total delay and minimising the search time. The algorithm is improved further by modifying the initially selected solutions, as well as the application of TS and SA, to achieve a better solution in a shorter search time (Ho and Yeung, 2001).

In most of the real-time railway rescheduling case studies, evolutionary algorithms cannot guarantee to find the optimal solution. However, using these algorithms is the only way to achieve a near optimal solution within a reasonable time.

## 2.5 Benchmark Scenario to the Single Junction Case Study

As discussed in the previous section, there are many different optimisation algorithms that can be used for optimal train rescheduling; each has its own strengths and weaknesses. For real-time train control, in practice, there is a trade-off between computation time and the identification of an optimal solution. Based on the current literature, it is difficult to choose the best optimisation algorithm to reduce the delay. In order to allow comparison of the reviewed optimisation algorithms, a benchmark scenario is introduced here.

### 2.5.1 Junction and Benchmark

A layout based on the North Stafford and Stenson Junctions on the Derby to Birmingham line in the UK is considered in the following three chapters. Figure 2-5 shows the layout with 12 approaching trains, numbered 1 to 12. The letters (shown in brackets) are of the form (origin, destination). It is assumed that the 'junction area' is initially clear.

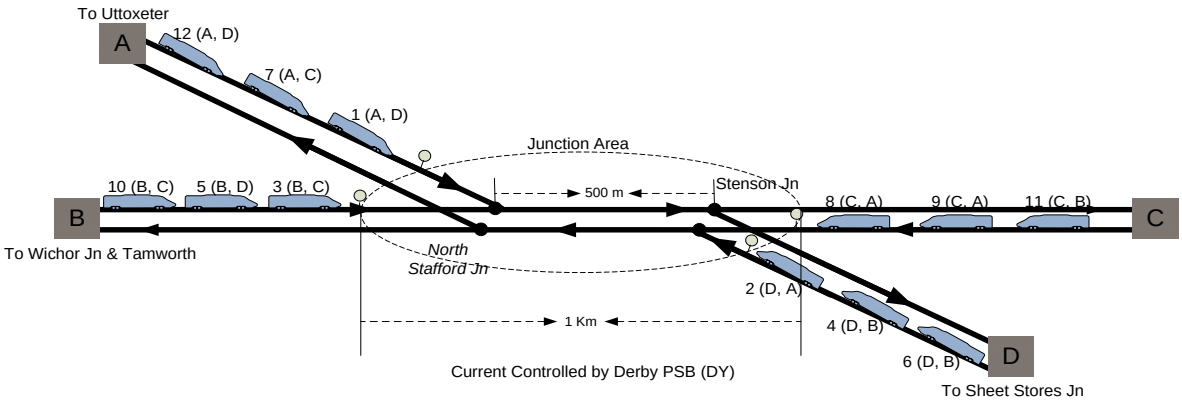


Figure 2-5: Example layout - North Stafford and Stenson junctions

The junction area speed limit is 48 km/h, so trains should slow down to the speed limit before running into the junction area and, for leaving the junction area, they must accelerate to the maximum allowed speed, which depends on the line speed limits and the designed speed limit of the vehicle. The conflict-free timetabled arrival times and train-specific delay penalties are shown in Table 2-3. Four scenarios are considered:

Scenario (1): Train 1 is delayed by 3 minutes - for this scenario a single train, the first train to pass through the junction area, is delayed;

Scenario (2): Train 1 is delayed by 5 minutes – for this scenario the order of Train 1 and Train 7 is reversed;

Scenario (3): Train 1 is delayed by 5 minutes, resulting in Train 7 being delayed by 3 minutes – for this scenario the delay of Train 1 results in a knock-on delay to Train 7;

Scenario (4): Train 1 is delayed by 4 minutes, Train 2 is delayed by 4 minutes and Train 3 is delayed by 4 minutes – for this scenario multiple trains in both directions are delayed and knock-on delays occur.

| **Train number** | **Train type** | **Delay penalty (£/min)** | **Initial distance from junction area (km)** | **Destination** | **Scheduled arrival time at destination station** |
|---|---|---|---|---|---|
| 1 | Class 150 | 20 | 3 | D | 12:14 |
| 2 | Class 220 | 40 | 3 | A | 12:13 |
| 3 | Freight | 10 | 4 | C | 12:27 |
| 4 | Class 220 | 40 | 5 | B | 12:27 |
| 5 | Freight | 10 | 5 | D | 12:31 |
| 6 | Class 150 | 20 | 7 | B | 12:17 |
| 7 | Freight | 10 | 9 | C | 12:19 |
| 8 | Class 150 | 20 | 9 | A | 12:17 |
| 9 | Class 220 | 40 | 11 | A | 12:25 |
| 10 | Class 220 | 40 | 10 | C | 12:28 |
| 11 | Freight | 10 | 13 | B | 12:29 |
| 12 | Class 150 | 20 | 13 | D | 12:31 |

Table 2-3: The conflict-free timetable and train delay penalties

In order to compare the solutions and computation times of the different algorithms, each of the four scenarios will be solved by each of the algorithms. The algorithms have been implemented using Matlab 2011a on a Dell Optiplex 755 computer (Intel Core 2 Duo CPU E8400 @ 3.00 GHZ, 1.96 GB of RAM). It is acknowledged that it would be possible to decrease computation times if the algorithms were implemented

using a lower level programming language. For each of the algorithms, the results for each scenario are shown in Chapters 3 and 4. The optimal solutions are marked with an asterisk.

### 2.5.2 Introduction to the Trains in this Case Study

There are three types of train in this case study, namely commuter train of Class 150, intercity train of Class 220 and freight train of the F2-mixed.

The Class 150, with the family name 'Sprinter', and are diesel multiple units, built by British Rail Engineering Limited from 1984-87. Class 150 trains are currently operated in Northern England, South-West England and the Midlands by Northern Rail, First Great Western and London Midland.

The Class 220, which has the family name, 'Voyager', is a diesel-electric, high-speed multiple-unit train, built by Bombardier Transportation between 2000 and 2001. The Class 220 is currently operated on UK Cross Country Routes by CrossCountry and Virgin Trains.

F2-mixed is a freight train, which is operated all over Europe. This is a diesel train which can deliver a 1041 tonne payload; it is normally used to deliver mail, coal and fuel oil.

The maximum running speed, Davis parameters (Rochard and Schmid, 2000), total masses, total train lengths, traction forces and braking rates of these three trains are shown in Table 2-4 and these parameters are used in the train running time estimation.

The values given in the last row of this table, 'Traction Force/Mass', gives a general idea of the acceleration rate when the train speed is 0.

| Parameter | Class 150 (Hillmansen *et al.*, 2008) | Class 220 (Hillmansen *et al.*, 2009) | F2-mixed (Lukaszewicz, 2007) |
|---|---|---|---|
| Davis Parameters:<br>A<br>B<br>C | <br>2.09 kN<br>0.00983 kN/ms$^{-1}$<br>0.00651 kN/m$^2$s$^{-2}$ | <br>3.4537 kN<br>0.0767 kN/ms$^{-1}$<br>0.0043 kN/m$^2$s$^{-2}$ | <br>11.5 kN<br>0.258 kN/ms$^{-1}$<br>0.037 kN/m$^2$s$^{-2}$ |
| Total Mass | 76.4 tonnes | 213.19 tonnes | 1041 tonnes |
| Total Train Length | 125 m | 195 m | 355 m |
| Power at Rails | 374 kW | 1568 kW | 2036 kW |
| Maximum Speed | 120 km/h | 200 km/h | 110 km/h |
| Maximum traction force | 40.5 kN | 136 kN | 187 kN |
| Maximum braking rate | 0.49 ms$^{-2}$ | 0.25 ms$^{-2}$ | 0.19 ms$^{-2}$ |
| Number of seats | 124 | 188 | none |
| Number of coaches | 2 | 4 | 24 (wagons) |
| Dwell Time | 30 seconds | 120 seconds | 10 minutes |
| Traction Force / Mass | 0.53 | 0.64 | 0.18 |

Table 2-4: Parameters of trains in the benchmark case studies

### 2.5.3 Introduction to the Simulator and Train Running Time Estimation

In the existing railway rescheduling approaches to traffic control introduced above, the majority of researchers use fixed running times for railway real-time rescheduling studies in order to reduce the overall computation time. In these studies, the journey time for trains to pass through a junction and the running time from a junction area to the final station are fixed. The final total running time is achieved by the summation of the junction passage time, the fixed running time plus any waiting time. However, the results from such calculations are only approximations of real operations and the results are commonly not realistic.

For all of the rescheduling methods in this thesis, it is very important to estimate the running time as accurately as possible, when the sequence of the trains to pass the

junction area is under consideration, This is because the correctness of estimation strongly affects the veracity of the rescheduling. The estimation should also calculate the running time in a very short time, as the estimation will often run multiple times, as required by some of the search methods.

Therefore, a multi train simulator is designed here. The principle, details and corresponding equations of this simulator are illustrated in detail in Appendix A. The train running time estimation in this simulator consists of four main elements:

1. Right of way signalling time calculation: Before estimating the train journey times one by one, according to a known order past a junction area in the simulation, it is essential to introduce the time superposition method first. This time superposition method is used before the sequence is fixed and when more than one train could pass the junction area at the same time. By using this method, the right of way signal time for each train can be given by the time the first of the front trains leave the junction area. The results of the superposition method will show which train is the front train.
2. Track information: The real track information data (e.g., the length and speed limit of each track section) is used in this simulator, to ensure the train travelling time calculation close to the real operational time as possible.
3. Vehicle information and traction system: In real operation, trains travel according to their own traction characteristics, in which the acceleration rate, braking rate, resistance *etc.* are variables. Therefore, homogeneous strip modelling is used here. In homogeneous strip modelling, each moving section

is divided into many small sections, and it is assumed that the variables (e.g. acceleration rate, braking rate, resistance and etc.) are constant in each small section. The running time of each train is simulated section by section to achieve an accurate estimation of their travelling times.

4. Train protection: In this simulator, the legal positions of the trains are always being considered. It is designed as a block section based simulation. The safe distances (legal headway) are calculated not only in the junction area, but also for the whole journey until trains arrive at their destinations.

### 2.5.4 Cost Functions

A number of possible solutions can be found to solve a rescheduling problem, giving different arrival times for the trains at their destination stations. The control centre is responsible for identifying an appropriate solution. The identification of an appropriate solution could include the consideration of: delay minutes, delayed passenger numbers, energy consumption, transport connections and financial delay penalties.

The rescheduling objective is measured by cost in the United Kingdom. There is a compensation policy for rail network delays, whereby Network Rail pays a fine to train companies for network control delays, which is calculated from delay minutes. Consequently, the cost function to be minimised in this thesis is the delay penalty of all the considered trains, which is:

$$J(\theta) = \sum_{i=1}^{n} DT_i(\theta)\, DP_i \tag{2-1}$$

where $n$ is the number of trains to be considered, $DT_i$ is the delay time for the $i$th train in minutes at its destination, $DP_i$ is the delay penalty per minute for the $i$th train, $\theta$ is the sequence of trains through the junction and $J(\theta)$ is the total delay cost of each solution. The optimal sequence, $\widehat{\boldsymbol{\theta}}$, i.e. the sequence with the lowest delay cost, is given by:

$$\widehat{\boldsymbol{\theta}} = \arg\min J(\theta) \tag{2-2}$$

## 2.6 Conclusions

This chapter has introduced the algorithms commonly used in research for train rescheduling. The algorithm types range from rule based (RB) type approaches that are computationally efficient, but do not usually find the optimal solution, to graph based algorithms, where the search time is reduced for the same solution, to evolutionary algorithms, which cannot guarantee to find the true optimum, but do give a quick solution which is practical for railway operations.

Before computer control was applied to railway rescheduling, trains were rescheduled by human operators. Fundamental algorithms were the first method used to solve the rescheduling problem. RB rescheduling not only offers a solution in a short time, but it is also easy to implement by either manual or automatic systems.

Shortest-path optimisation algorithms such as BF and DP have been considered. However, since these algorithms are full search algorithms, the computing time is dependent on the complexity of the problem. Due to their computation time, they are

unsatisfactory for real-time rescheduling in complex junctions or for situations where a large number of trains are involved in the delay.

With the intention of improving the application of the full search algorithms, graphical methods, such as DT and SST, were studied since they have the potential to improve the application of the full search algorithms. These graphical method algorithms can offer a different understanding of the railway rescheduling problem which could be used in conjunction with other algorithms. For example, DP can only be carried out with SST, from which a shortest path can be found. Through the use of a DT, BF can be constrained to only consider feasible solutions. By using these graph based algorithms, the search time can be reduced. Nevertheless, the improvement is not significant; the graph-based search algorithms are still unsatisfactory for practical real-time rescheduling.

Evolutionary algorithms can give a good solution within a reasonable time, which meets the requirements of real-time train rescheduling. Solutions are obtained through iterative processing that utilises memory analysis. The memory analysis gives important feedback to the algorithm itself, which can lead to finding better solutions and help to avoid evaluating high cost solutions. Evolutionary algorithms cannot guarantee to find the true optimum, but they give a quick solution to meet the demands of railway operations.

The next chapter will present a numerical benchmark of fundamental and graphical algorithms.

# Chapter 3: Solutions of the Benchmark Problem Using Fundamental and Graphical Algorithms

In this chapter, four of the fundamental and graphical optimisation algorithms will be considered, namely: Brute Force (BF), First-Come-First-Served (FCFS), Dynamic Programming (DP), and Decision Tree Based Elimination (DTBE). All of these approaches will be applied to minimising the cost of disturbances.

## 3.1 Brute Force

### 3.1.1 Introduction to Brute Force Algorithm

BF, known as exhaustive enumeration, search consists of enumerating all possible candidate solutions for the problem and evaluating whether each candidate satisfies the problem statement. BF can be used in the domain of routing problems, reconfiguration problems and combinational problems. The most commonly used application area for BF is in cryptography attack; all possible keys are tried one by one in an encryption system until the correct key is found (Paar and Pelzl, 2010). In train rescheduling studies, BF searching generates and evaluates all possible sequences to find the minimal delay cost sequence.

### 3.1.2 Study of Brute Force Algorithm in Railway Rescheduling

BF can be described in two main steps in train rescheduling applications:

1. Enumerating all possible sequences;
2. Evaluating all sequences

There are a number of ways to enumerate. However, many infeasible sequences are generated if the full permutation method is used. For example, train 3 physically cannot pass the junction before train 1 in the example junction shown in Figure 3-1. In practical problems the overwhelming majority of the sequences are infeasible.

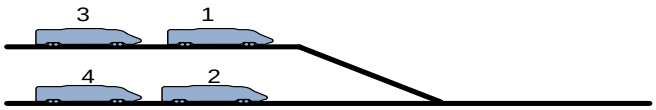


Figure 3-1: Simple example of conflict at junction

#### *3.1.2.1 Feasible and Infeasible Solutions*

The number of different sequences can be calculated by factorial study. According to the full permutation principle, if there are *n* trains at the junction, the number of total sequences, which includes all feasible and infeasible sequences, can be worked out by:

$$total\ number\ of\ orderings = n\,! \tag{3-1}$$

Based on Equation 3-1, the feasible sequences can be calculated by combination theory (Alexander, 2003). Suppose if there are *n* trains on *m* different tracks at the junction, the number of trains on the same track is $n_1, n_2, \cdots \cdots, n_m$, with $\sum_{i=m}^{1} n_i\,! = n$. The number of different conflict-free solutions is:

$$number\ of\ feasible\ orderings = \frac{n!}{n_1!\, n_2! \cdots \cdots n_m!} \quad (3\text{-}2)$$

For the example junction shown in Figure 3-1, there are 4 trains at the junction, and there are 2 trains on two different tracks. The number of feasible sequences can be found by Equation 3-2:

$$\frac{4!}{2!2!} = 6 \quad (3\text{-}3)$$

Therefore, there are 6 feasible sequences for these 4 trains to pass the junction area.

For another more complex example, such as the main case study, introduced in the previous chapter (Figure 2-5), there are 12 trains at the junction, with 3 trains each from four different directions. The number of feasible sequences can be found by Equation 3-2:

$$\frac{12!}{3!3!3!3!} = 369600 \quad (3\text{-}4)$$

There are 369,600 feasible sequences for these 12 trains to pass the junction area, but the number of total sequences is 479,001,600 which can be found by Equation 3-1. This means only 0.077 % of the total sequences are feasible.

#### *3.1.2.2 The Application of BF in Train Rescheduling*

By using a decision tree, the enumeration method can be refined so that it only generates feasible solutions (Anany, 2003). The orders in which the trains arrive at the junction from each direction constrain the solutions. In this example, as there are four directions from which trains can approach the junction, four queues, $Q_A$ to $Q_D$, are formed, as shown in Figure 3-2.

| | 1 | 2 | 3 |
|---|---|---|---|
| Q A | 1 | 7 | 12 |
| Q B | 3 | 5 | 10 |
| Q C | 8 | 9 | 11 |
| Q D | 2 | 4 | 6 |

Figure 3-2: Necessary sequence of trains for the benchmark problem (Fan)

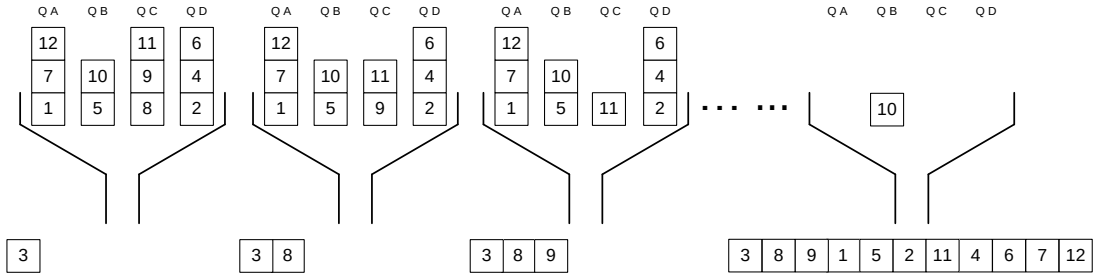


Figure 3-3: Example solution using the enumeration method (Fan)

To decide which train should pass the junction first, solutions are formed that commence with the first element of any queue. Elements must always be picked in the order that they appear in their queue. Respecting these two rules, the elements may then be selected in any order; see example in Figure 3-3.

Individual solutions are considered complete when all queues are empty (as shown in Figure 3-3). A complete set of solutions is formed when all permutations have been exhausted. The cost function is used to identify the solution with the lowest delay cost.

### 3.1.3 Results and Summary

Using BF, all the possible solutions are enumerated and their costs are computed. The solution with the least cost can then be found. If the lowest cost applies to more than

one solution, any of these solutions can be chosen. The BF method will always find the optimal solution.

The result of the BF method when used to solve scenarios 1 to 4 is shown in Table 3-1. Optimal solutions are marked with an asterisk.

| | Computation Time (s) | Total Delay Cost (£) | Sequence and Delay Cost |
|---|---|---|---|
| **Scenario 1** | 1692 | 90 * | Order: 2, 4, 3, 1, 5, 6, 7, 8, 9, 10, 11, 12<br>Delay: 1 (£60), 5 (£10), 8 (£20) |
| **Scenario 2** | 1692 | 230 * | Order: 2, 3, 4, 5, 6, 7, 8, 1, 9, 10, 11, 12<br>Delay: 1 (£120), 9(£40), 10(40), 11(£10), 12 (£20) |
| **Scenario 3** | 1692 | 290 * | Order: 2, 3, 4, 5, 1, 6, 8, 9, 10, 11, 7, 12<br>Delay: 1(£80), 6(£20), 8(£20), 9(£40), 10(£40), 11(£10), 7(£40), 12(£40) |
| **Scenario 4** | 1692 | 370 * | Order: 8, 1, 2, 9, 3, 4, 5, 6, 7, 10, 12, 11<br>Delay: 1(£60), 2(£120), 3(£10),4(£40),5(£10), 6(£20), 7(£10), 10(£40), 12(£20), 11(£40) |

Table 3-1: Results of BF for each scenario

The BF method can take a considerable time to find the optimal solution, due to the number of solutions that must be enumerated. Even in this simple case, 369,600 solutions are evaluated for every feasible solution; the computation times are therefore always the same (approximately 28 minutes). Such processing times are not practical for real-time railway operations.

Brute Force is the most straightforward approach to find the optimal solution. It cannot be implemented in real-time rescheduling due to the considerable computing time. However, brute force may be the best approach when a super-computer is applied in the future.

## 3.2 First-Come-First-Served

### 3.2.1 Introduction to the First-Come-First-Served Algorithm

The FCFS algorithm is the simplest rescheduling algorithm; trains are rescheduled according to their arrival time. The FCFS algorithm is commonly used in many areas to achieve fairness, such as queuing in our daily life and communications engineering (Loher, 1998). In communications, the packet which arrives first should also be the packet first transmitted. However, FCFS is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait (Brucker, 2007).

### 3.2.2 Study of First-Come-First-Served in Railway Rescheduling

FCFS is also the simplest algorithm in railway rescheduling applications. Trains are allowed through the junction in the order in which they arrive (Schmid and Blieberger, 1992). In the case studies considered in this thesis, the trains will pass the junction in the order in which they reach the edge of the junction region.

### 3.2.3 Results and Summary

| | Computation Time (s) | Total Delay Cost (£) | Sequence and Delay cost |
|---|---|---|---|
| **Scenario 1** | 0.005 | 150 | Order: 2, 3, 4, 5, 1, 6, 7, 8, 9, 10, 11, 12,<br>Delay: 1(£80), 9(£40), 11(£10), 12(£20) |
| **Scenario 2** | 0.005 | 340 | Order: 2, 3, 4, 5, 6, 7, 1, 8, 9, 10, 11, 12<br>Delay: 1(£120), 8(£40), 9(£80), 10(£80), 11(£20) |
| **Scenario 3** | 0.005 | 360 | Order: 2, 3, 4, 5, 6, 8, 1, 9, 10, 7, 11, 12<br>Delay:1(£100), 9(£80), 10(£80), 7(£40), 11(£20), 12(£40) |
| **Scenario 4** | 0.005 | 570 | Order: 8, 9, 11, 1, 2, 3, 4, 5, 6, 7, 10, 12<br>Delay: 1(£80), 2(£160), 3(£40), 4(£80), 5(£20), 6(£40), 7(£10) , 10(£80), 12(£40) |

Table 3-2: Results of FCFS for each scenario

The results of the FCFS method applied to the four scenarios are shown in Table 3-2. No optimal solutions are found. Owing to the nature of the method, in every case only one solution is considered; the computation times are therefore always the same.

FCFS is commonly used in railway control centres around the world. FCFS is simple to understand, gives a quick response time and can either be simply carried out by a signaller or by an automatic control system. Therefore, FCFS should primarily be used as an initial solution to initiate other methods or as a fallback solution if insufficient time is available to use other methods. Furthermore, FCFS should be used as a general 'rule-of-thumb' for signallers.

## 3.3 Dynamic Programming

### 3.3.1 Introduction to Dynamic programming

DP is one of the most common shortest path algorithms. It was first proposed and later developed by Richard Bellman in the 1950s (Dreyfus, 2002). DP is normally

understood as a divide-and-conquer method which solves problems by combining the solutions to subproblems (Cormen *et al.*, 2001). The dividing is based on the definition and structure of the problem into multi-stages of time or space, like the shortest path problem shown in Figure 3-4. The shortest path between A and G can be found by dividing it into six stages; there are many different states in each stage. The subproblems are solved by either going forwards stage to stage (A to B, B to C, --- ---, F to G) or backwards (G to F, F to E, --- ---, B to A).

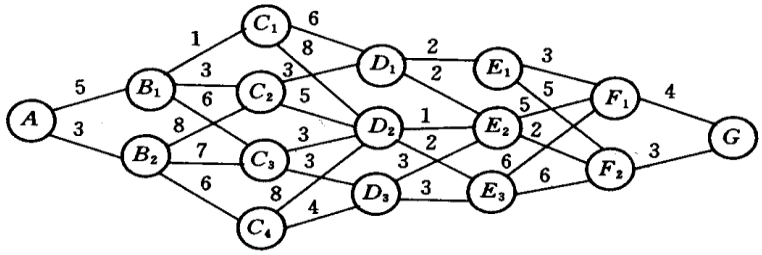


Figure 3-4: Example of a shortest path problem (Lew and Mauch, 2007)

The optimal solutions of the subproblems can be found by evaluating the path length to reach each state of the stage. The longer path(s) are no longer saved. For example, to reach state $C_2$, there are two paths: A→$B_1$→$C_2$ (length = 8) and A→$B_2$→$C_2$ (length = 11). The optimal solution to reach state $C_2$ is A→$B_1$→$C_2$ (length = 8).

### 3.3.2 Study of Dynamic programming in Railway Rescheduling

The first step of DP is to restructure the problem as a graph. The first stage of the graph represents the initial conditions of the problem, e.g., in Figure 3-5 four trains are waiting to pass the junction. The next stage shows all the possible states with three trains remaining to pass the junction, i.e., one train having passed. The graph is continued until all trains have passed the junction.

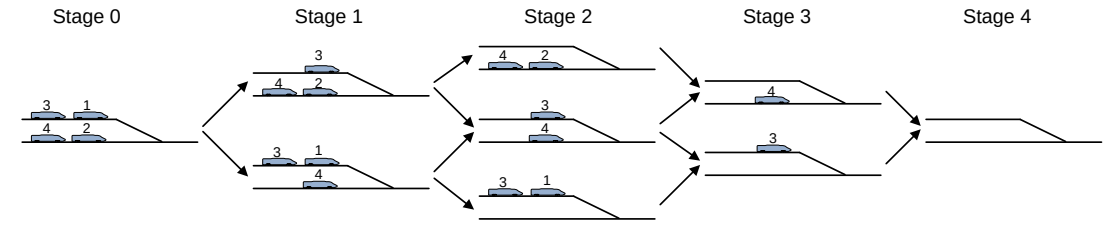


Figure 3-5: An example of the shortest path problem (Fan)

DP can either work forwards or backwards, but it is more common to see DP working backwards. However, in railway rescheduling studies, backwards processing cannot be carried out as the running time of the final trains to pass through the junction are not known at the outset.

In this forward Stage-to-Stage model of railway rescheduling, there are many different partial sequences that can reach the same state. In dynamic programming, each state has only one cost because only the optimal path to reach each state is saved. In Figure 3-6 one of the states of Stage 2 is shown; both partial sequence 1-2 and 2-1 result in this state being reached, but only the partial sequence with the lowest cost is used.

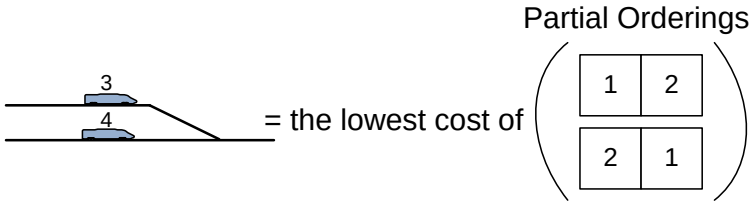


Figure 3-6: An example state of dynamic programming process (Fan)

To reduce the delay cost, in a railway rescheduling problem, using this method would lead to errors. For example, the partial sequence 1-2 gives the lowest delay cost at an intermediate stage due to the penalty costs of the specific train types, but this partial

sequence solution could then potentially cause a longer delay time for later trains. Therefore, all partial sequences and their delay costs, together with total journey time and section journey times must be considered.

The application of the dynamic programming algorithm in railway rescheduling comprises four steps:

Step 1: Identification of the number of stages. This is one more than the number of trains, numbered from stage 0.

Step 2: Identification of the states in each stage. Starting at Stage 0, the states in each stage are found by identifying all of the possible next configurations if one train is allowed to pass the junction. This finalises the formation of the graph.

Step 3: Calculation of the cost of each stage change. The cost is found by simulating the current train, plus previous trains, that have passed through the junction. By doing this, all partial sequences are evaluated once only.

Step 4: In the last stage, the lowest cost is known, then, the sequence of the lowest cost can be found by tracing back to the first stage.

### 3.3.3 Results and Summary

The results of dynamic programming are shown in Table 3-3. Optimal solutions are marked with an asterisk.

| | Computation Time (s) | Total Delay Cost (£) | Sequence and Delay Cost |
|---|---|---|---|
| **Scenario 1** | 318.092 | 90 * | Order: 2, 4, 3, 1, 5, 6, 7, 8, 9, 10, 11, 12<br>Delay: 1 (£60), 5 (£10), 8 (£20) |
| **Scenario 2** | 318.099 | 230 * | Order: 2, 3, 4, 5, 6, 7, 8, 1, 9, 10, 11, 12<br>Delay: 1 (£120), 9(£40), 10(£40), 11(£10), 12 (£20) |
| **Scenario 3** | 318.106 | 290 * | Order: 2, 3, 4, 5, 1, 6, 8, 9, 10, 11, 7, 12<br>Delay: 1(£80), 6(£20), 8(£20), 9(£40), 10(£40), 11(£10), 7(£40), 12(£40) |
| **Scenario 4** | 318.086 | 370 * | Order: 8, 1, 2, 9, 3, 4, 5, 6, 7, 10, 12, 11<br>Delay: 1(£60), 2(£120), 3(£10),4(£40),5(£10), 6(£20), 7(£10), 10(£40), 12(£20), 11(£40) |

Table 3-3*:* Results of dynamic programming for each scenario

The sequences (and hence the costs) are the same as the brute force method. However, the computation time is about 5 times quicker than brute force. The main reason is that DP evaluates each partial sequence just once and saves its cost, thereby avoiding the work of recalculating the cost every time it evaluates each sub-problem. For example, there are 1680 different feasible sequences which start with the partial sequence 1-2-3-4. The delay cost of this partial sequence 1-2-3-4 is evaluated just once rather than the 1680 times explained in BF.

## 3.4 Decision Tree Based Elimination

### 3.4.1 Introduction to Decision Tree Based Elimination

Since the search times of BF and DP are unsatisfactory for real-time rescheduling, Decision Tree Based Elimination (DTBE) is considered for this application. The approach is designed to reduce the search time by pruning the decision tree according to some heuristic rules. These rules are based on an evaluation of the costs of partial paths of decision trees; the partial paths with higher costs are restricted from generating any further branches. In the DTBE process, the high cost nodes are pruned

at each stage / layer until each branch from the root node is terminated with a leaf node and no further branching is required. DTBE can also be understood as a development of the branch and cut algorithm in combinatorial optimisation, see Section 2.3.2.

From the railway rescheduling point of view, evaluating partial tree paths does not necessarily give a reliable indication of the value of the full sequence. For example, in an sequence where the first three trains are late, but the next nine are on time, a partial evaluation of three trains would indicate that this is a poor solution, while overall it may be optimal or close to optimal.

Furthermore, in a full evaluation of all possible solutions the top three levels would only contain a few nodes, which can be fully evaluated in a short computing time. Therefore, for railway rescheduling, DTBE is designed such that the first three layers are fully generated and evaluated, and then the pruning starts with the evaluated nodes in the third layer.

### 3.4.2 Study of Decision Tree Based Elimination in Railway Rescheduling

Referring to Figure 3-7, the generalised algorithm can be described in a number of steps:

Step 1: The first three layers of the decision tree are generated; this corresponds to the first three trains passing the junction area.

Step 2: The delay cost for each leaf node is calculated, and the most costly 30% of the nodes are eliminated.

Step 3: The next layer of the tree is generated.

Step 4: Repeat Steps 2 and 3 until all trains have passed the junction. The lowest delay cost solution can then be found.

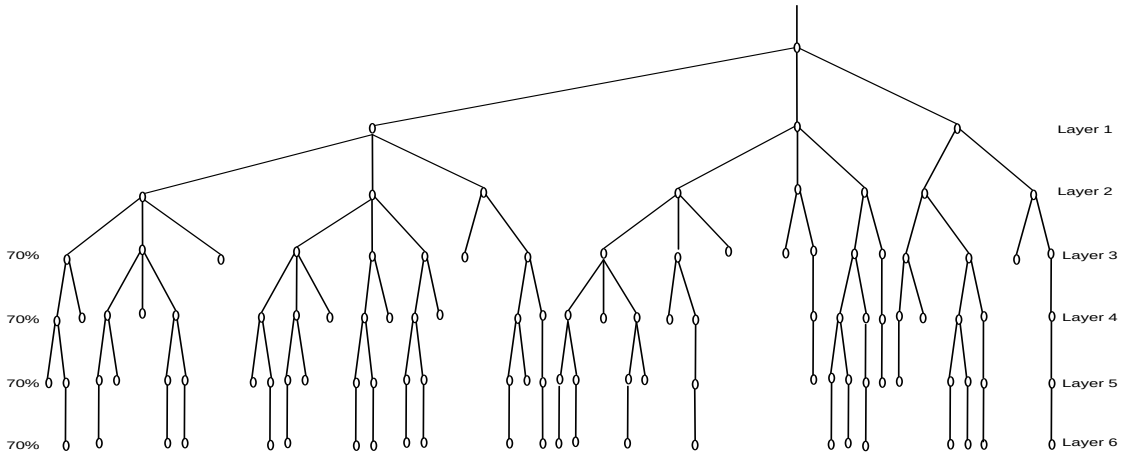


Figure 3-7: DTBE algorithm technique (Fan)

### 3.4.3 Results and Summary

The results of the DTBE method applied to scenarios 1 to 4 are shown in Table 3-4. Optimal solutions are marked with an asterisk.

| | Computation Time (s) | Total Delay Cost (£) | Sequence and Delay Cost |
|---|---|---|---|
| **Scenario 1** | 70.356 | 90 * | Order: 2, 4, 3, 1, 5, 6, 7, 8, 9, 10, 11, 12<br>Delay: 1 (£60), 5 (£10), 8 (£20) |
| **Scenario 2** | 70.437 | 300 | Order: 2, 3, 4, 5, 6, 7, 8, 1, 9, 10, 11, 12<br>Delay: 1(£120), 9(£80), 10(£80), 11(£20) |
| **Scenario 3** | 70.341 | 310 | Order: 2, 3, 4, 5, 6, 8, 9, 10, 11, 1, 7, 12<br>Delay: 1(£160), 7(£50), 12(£100) |
| **Scenario 4** | 70.652 | 570 | Order: 8, 9, 11, 1, 2, 3, 4, 5, 6, 7, 10, 12<br>Delay: 1(£80), 2(£160), 3(£40), 4(£80), 5(£20), 6(£40), 7(£10) , 10(£80), 12(£40) |

Table 3-4: Results of DTBE for each scenario

Due to the elimination approach, the method is not guaranteed to produce the optimal solution. For problems with 12 trains, only 4% (i.e. $0.7^{(12-3)}$ x 100%) of the possible solutions are fully generated. This results in a comparatively fast computation time, as

shown in Table 3-4. However, the earlier stages still require full computation of the partial costs.

The other main point for consideration is the percentage of the nodes that are eliminated in each layer. This percentage chosen is a trade off between search quality and computation time. This percentage can be adjusted depending on the complexity of the junction and the number of trains at the junction. For the case study, 10%, 20%, 30% and 40% have been tried, with 30% performing the best. The results for 30% are shown in this thesis.

## 3.5 Conclusion

For this chapter, the author introduced the application of four fundamental and graphical algorithms to a benchmark problem.

Brute force is an exhaustive search algorithm, and hence will always find the optimal solution. It can be concluded that due to the computation time required which is consistently longer than that of the other algorithms considered, BF is most appropriate to be implemented in simple scenarios. However, this may change as computer processing capabilities increase.

FCFS is the quickest and simplest algorithmic method and it is the best method to use for manual rescheduling by signallers. However, compared with the other methods considered in this chapter, the results are significantly suboptimal. The FCFS rescheduling method is unfair in the sense that a train which has a large built in recovery time may pass, whilst a tightly scheduled train may be forced to wait. Because it cannot efficiently reduce high delay costs, replacing FCFS is an inevitable

part of accomplishing control centre modernisation for railway rescheduling problems. Nevertheless, the FCFS solution can be used to initialise evolutionary algorithms.

Dynamic programming not only gives the optimal solution, but also reduces the computation time compared with BF. However, the computation time is still long. Using this method, it is difficult to obtain a quick operational decision in complex railway rescheduling problems.

DTBE gives a solution with the second shortest computing time of the 4 algorithms considered in this chapter. However, it cannot guarantee to give the optimal solution, as the optimal solution may be eliminated during the pruning of an early layer. The implementation of DTBE is quite simple compared to other methods. Furthermore, the search quality of the DTBE method can be improved by adapting it to the specific problem being considered.

In addition, DP and DTBE do not have a solution until the algorithm finishes, whereas other algorithms can be stopped at any time and the current best answer taken during an urgent dispatch.

# Chapter 4: Solutions of the Benchmark Problem Using Evolutionary Algorithms

Evolutionary algorithms, which are also called metaheuristic algorithms in some textbooks, are used in the fields of management science, information systems, business informatics and computer science. For this chapter, the author describes the use of four evolutionary algorithms applied to the benchmark problem. After this, all eight optimisation algorithms, including the four fundamental and graphical algorithms applied in the previous chapter, are discussed and compared.

## 4.1 Local Search and Solution Adjustment

### 4.1.1 Local Search

Local search method is a class of elementary evolutionary algorithms. It is necessary to introduce these algorithm first as they are used to form more advanced algorithms, such as the Tabu search and simulated annealing.

Local search is referred to as a guided random search approach. The principle of local search is to modify repeatedly, within a local region, candidate solutions until some termination condition is reached, such as the maximum number of iterations or until no further improvement over a finite number of iterations can be found (Maniezzo *et al.*, 2009*)*. The concept of "local" is defined by using neighbourhoods. in railway

rescheduling, a neighbouring solution is defined as one where the order of a pair of trains changes. There are three common ways to find a neighbouring solution:

The first method is normally applied in nonlinear, multi-parameter searching problems. A neighbouring solution is found by altering one or more parameter values according to a certain method (Abido, 2001), such as increasing or decreasing specific parameter values by a certain amount.

A second is generally used in combinational and routing problems, such as the classic travelling salesman problem and the N-Queens Problem (Gu and Huang, 1994, Sosic and Gu, 1994). In these cases, the solution is in the form of a sequence. To develop a new solution, part of the sequence of an existing solution is transplanted into a new position. This approach is used in the genetic local search algorithm (Liu, 2011, Merz and Freisleben, 1997).

The third method involves swapping two parts of the solution. This is the most common procedure used in the domain of scheduling problems, including railway rescheduling (Yeung and Ho, 2000).

The use of local search methods for railway junction conflict resolution can be explained in three steps. A simple example where 6 trains are required to pass through a junction is shown in Figure 4-1, is used to illustrate the method.

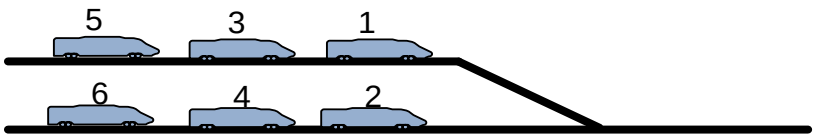


Figure 4-1: A simple junction conflict

Step 1: Select an sequence, $\hat{\theta}$, (it is common to use FCFS as an initial sequence), and calculate the delay cost, $J(\hat{\theta})$.

Step 2: Randomly, select two different trains from $\hat{\theta}$. The positions of these two trains are then swapped (pairwise exchange) to achieve a new sequence, $\theta$, see Figure 4-2. If the sequence is infeasible, then it should be converted to a feasible solution (see Section 4.1.2). When a feasible solution is found, the delay cost of the new sequence, $J(\theta)$, should be calculated. If $J(\theta) \leq J(\hat{\theta})$, then $\hat{\theta} := \theta$.

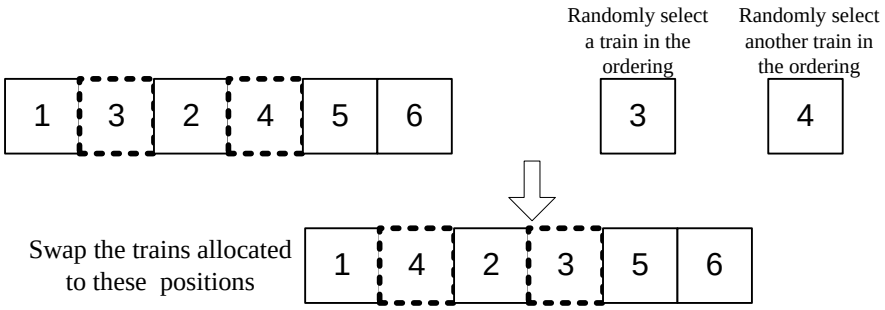


Figure 4-2: Local search technique in railway rescheduling study (Fan)

Step 3: Repeat Step 2 until some stopping criterion is met.

In evolutionary algorithms, the initial solution is important. A good start not only helps to achieve better solutions, but also reduces the computation time. The effectiveness of using FCFS as the start is demonstrated by Ho and Yeung (2001).

### 4.1.2 Solution Feasibility and Adjustment

In the local search application, random pairwise exchanges do not always, or even usually, result in feasible sequences. In the case study, train 3 cannot pass the junction before train 1, therefore swapping the positions of trains 1 and 3 in the sequence is not feasible. In principle, it would be possible to choose random pairwise exchanges until the new sequence is feasible but, in practice, a lot of possible swaps are not feasible so a lot of time can be wasted trying new possibilities. Therefore, a simple infeasible to feasible sequence converter has been constructed by the author (this converter can also be understood as elimination).

To convert an infeasible sequence to one that is feasible:

1. Firstly, the order of arrival of trains from each direction into the junction area is identified. These are referred to as the sequence constraints.

2. The candidate sequence is then examined, starting with the first train, to identify whether it meets the sequence constraints (i.e., is it feasible?).

3. If a train is identified as not meeting the sequence constraints, it is ‘pushed back’ in the sequence to the first point where the constraint is satisfied.

The approach is illustrated in Figure 4-3. The process begins by considering whether the first train in the sequence satisfies the sequence constraints (in this case it does as Train 1 is the first train in one of the constraint sequences). Trains are considered in turn and, if trains that do not meet the sequence constraints are found, the train is ‘pushed back’ to a feasible position; for example, Train 5 must be ‘pushed back’.

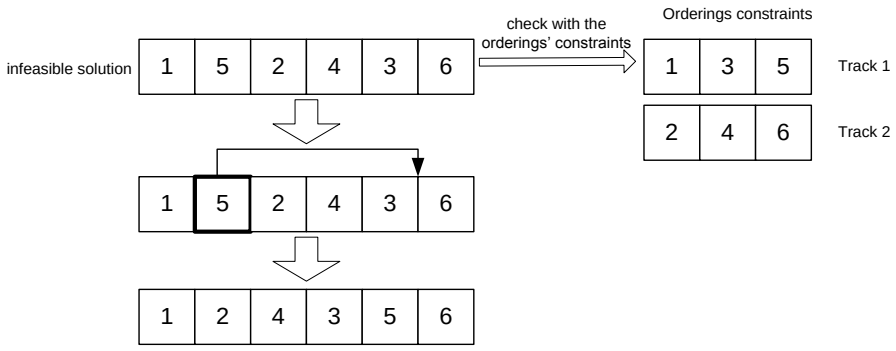


Figure 4-3: Solution adjustment technique (Fan)

### 4.1.3 Local Minima

In many studies it has been found that local search algorithms tend to get trapped into local minima (for combinatorial problems a local minimum is a solution from which an improved solution cannot be found in one step) (Elmihoub *et al.*, 2004). Improved evolutionary algorithms such as TS and SA have been developed to overcome the local minimum problem.

## 4.2 Tabu Search

### 4.2.1 Introduction to Tabu Search

The principle of TS was presented by Glover (1989, 1990) to solve large scale combinatorial optimisation problems. It has been shown that TS is more effective than local search (Reeves, 1993). TS enhances the performance of the local search method by using memory structures: once a potential solution or swap has been determined, it is marked as 'tabu' and saved in a tabu list. If a swap is present in the tabu list it cannot be repeated.

### 4.2.2 The Application of Tabu Search in Railway Rescheduling

Many researchers, such as (Pham and Karaboga, 2000), recommend that the length of the tabu list should be kept at around 20 to 30% of the number of possible swaps to

avoid all possible swaps appearing in the list and deadlock occurring. As a results, the number of possible swaps for a 12 train case study is:

$$C_{12}^{2} = 12 \times 11 = 132$$

The length of the tabu list should therefore be kept at between 26 and 40 for this case study. In the particular study, the tabu list is chosen to be of length 40 and is partitioned into four, with each partition relating to one iteration. Each tabu list partition is a $10 \times 2$ matrix. Once a swap has occurred it is added to the current partition. Once an iteration is complete (i.e., 10 swaps from the same seed have been carried out) the current partition is saved in memory for a further three iterations and then discarded.

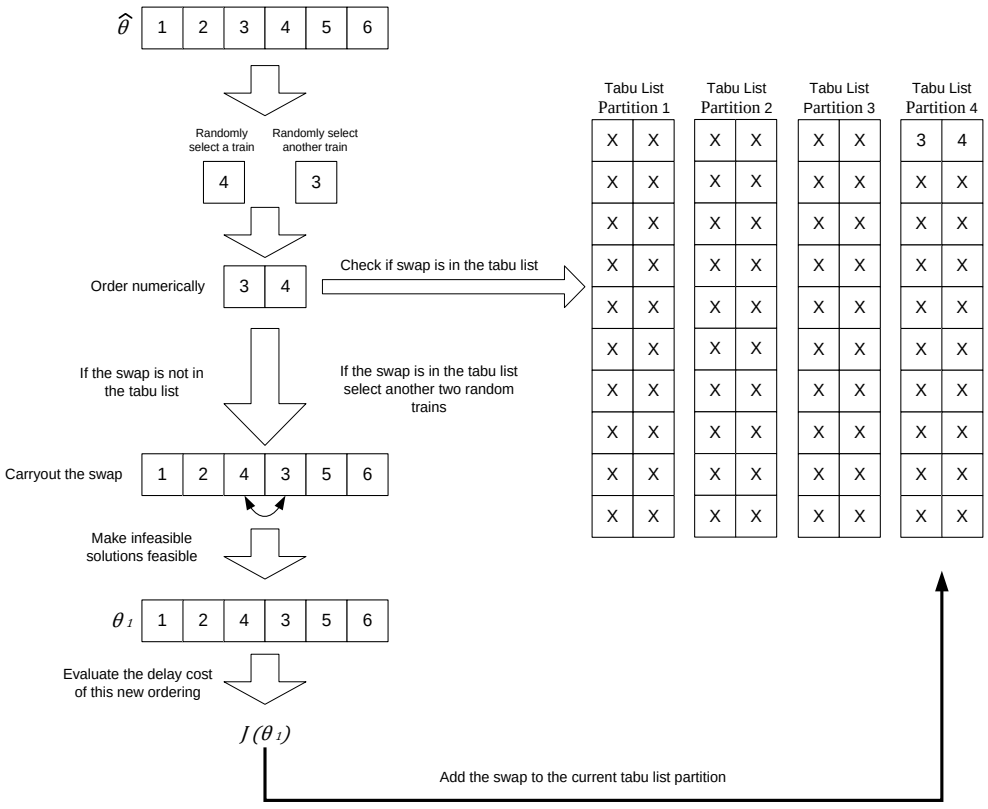


Figure 4-4: The technique of tabu search (Fan)

The application of the TS algorithm in railway rescheduling comprises four steps:

Step 1: Select the FCFS sequence $\hat{\theta}$ as the initial solution and evaluate its delay cost, $J(\hat{\theta})$.

Step 2: As shown in Figure 4-4, two different trains are randomly selected from sequence $\hat{\theta}$. These two trains are then arranged in ascending order. If this swap is in the tabu list, another two trains are randomly selected until a swap that is not in the tabu list is found. The positions of the selected trains are swapped to generate a new sequence, $\theta_n$. The feasibility of this sequence is checked and, if the sequence is found to be infeasible, it is made to be feasible. The corresponding delay cost $J(\theta_n)$ is calculated and the swap is added to the tabu list.

Step 3: Repeat Step 2 until the tabu list partition is full. Find the lowest $(\theta)$. If $J(\theta) \leq J(\hat{\theta}), then\ \hat{\theta} := \theta$.

Step 4: Go back to Step 2 unless the termination criterion has been met.

There are many termination criteria for evolutionary algorithms, two of which are introduced here:

1. The total number of iterations termination criterion (e.g., stop the algorithm after 20 iterations).
2. The no improvement termination criterion (e.g., stop the algorithm if no improvement is made after 5 consecutive iterations).

The no improvement termination criterion is used in this thesis.

### 4.2.3 Results and Summary

The result of the TS applied to scenarios 1 to 4 is shown in Table 4-1. Optimal solutions are marked with an asterisk.

| | Computation Time (s) | Total Delay Cost (£) | Sequence and Delay Cost |
|---|---|---|---|
| **Scenario 1** | 71. 642 | 150 | Order: 2, 3, 4, 5, 1, 6, 7, 8, 9, 10, 11, 12<br>Delay: 1( £80), 9( £40), 11( £10), 12( £20) |
| **Scenario 2** | 68. 546 | 230 * | Order: 2, 3, 4, 5, 6, 7, 8, 1, 9, 10, 11, 12<br>Delay: 1 ( £120), 9( £40), 10( £40), 11( £10), 12 ( £20) |
| **Scenario 3** | 72. 642 | 360 | Order: 2, 3, 4, 5, 6, 8, 1, 9, 10, 7, 11, 12<br>Delay: 1( £100), 9( £80), 10( £80), 7( £40), 11( £20), 12( £40) |
| **Scenario 4** | 69.267 | 570 | Order: 8, 9, 11, 1, 2, 3, 4, 5, 6, 7, 10, 12<br>Delay: 1( £80), 2( £160), 3( £40), 4( £80), 5( £20), 6( £40), 7( £10) , 10( £80), 12( £40) |

Table 4-1: Results of TS for each scenario

## 4.3 Simulated Annealing

### 4.3.1 Introduction to simulated annealing

Simulated Annealing (SA) was first introduced by Kirkpatrick *et al.* (1983). Like the Tabu Search, SA is a development of a local search. The algorithm is based on the annealing process used in metallurgy, where heating and controlled cooling of metal is used to control the size of crystal formation. From an initial seed, a new candidate solution is identified, based on a local search. If the candidate solution is an improvement on the original solution it is retained. If the candidate solution is worse, the candidate may still be selected based on a probabilistic function whereby the likelihood of accepting a worse solution is dependent on the cost difference and length of time the algorithm has been running (temperature). Such an approach allows

the SA algorithm the ability to move around the search space when the temperature is high, which prevents the algorithm being trapped at a local minimum.

### 4.3.2 Study of Simulated Annealing in Railway Rescheduling

For SA, each possible solution, $\theta_k$, in the search space represents a state of the system, and the objective is to find $\theta$ that minimises the cost, $J(\theta)$ (Pham and Karaboga, 2000).

In SA, the temperature decreases from 1 to 0. The common method used to decrease the temperature is the power function, $\alpha^j$, where $\alpha$ is the rate of temperature decrease, which should be in the range of $0 < \alpha < 1$, and $j$ is the iteration count, starting from 0. $\alpha$ is normally chosen to be bewteen 0.90 and 0.99 in order to prevent the temperature decreasing too rapidly (Azizi and Zolfaghari, 2004). 0.95 has been used for the examples presented in this thesis.

The algorithm can be summarised in the following steps:

Step 1： Set the initial solution, $\hat{\theta}$, to be the FCFS solution and set $j$, the iteration number, to 0.

Step 2： $\theta$ is selected using the local search method. If the newly proposed sequence is infeasible, the solution is modified until it becomes feasible. The difference in cost between the two (feasible) solutions is calculated by $\Delta J = J(\theta) - J(\hat{\theta})$.

If $\Delta J \leq 0$, or $\Delta J > 0$ and $exp\left(-\frac{\Delta J}{T_j}\right) > random\ (0, 1)$ then $\hat{\theta} = \theta$, where $T_j = \alpha^j$, else continue. Increment $j$ by 1.

Step 3: Go to Step 2 unless the termination criterion has been met.

### 4.3.3 Results and Summary

The results for using SA to solve the benchmark problem are shown in Table 4-2. Optimal solutions are marked with an asterisk.

| | Computation Time (s) | Total Delay Cost (£) | Sequence and Delay Cost |
|---|---|---|---|
| **Scenario 1** | 216.865 | 90 * | Order: 2, 4, 3, 1, 5, 6, 7, 8, 9, 10, 11, 12<br>Delay: 1 (£60), 5 (£10), 8 (£20) |
| **Scenario 2** | 223.214 | 230 * | Order: 2, 3, 4, 5, 6, 7, 8, 1, 9, 10, 11, 12<br>Delay: 1 (£120), 9(£40), 10(£40), 11(£10), 12 (£20) |
| **Scenario 3** | 208.396 | 290 * | Order: 2, 3, 4, 5, 1, 6, 8, 9, 10, 11, 7, 12<br>Delay: 1(£80), 6(£20), 8(£20), 9(£40), 10(£40), 11(£10), 7(£40), 12(£40) |
| **Scenario 4** | 227.951 | 370 * | Order: 8, 1, 2, 9, 3, 4, 5, 6, 7, 10, 12, 11<br>Delay: 1(£60), 2(£120), 3(£10),4(£40),5(£10), 6(£20), 7(£10), 10(£40), 12(£20), 11(£40) |

Table 4-2: Results of SA for each scenario

## 4.4 Genetic Algorithms

### 4.4.1 Introduction to Genetic Algorithms

Genetic Algorithms (GA) can trace their history back to the 1950s, to the work of mathematician Nils Aall Barricelli. More recently, Genetic Algorithms were developed further by Holland (1975). The Genetic Algorithm is loosely modelled on the principles of evolution via natural selection, as shown in Figure 4-5. The evolution of a population of individuals is carried out through competition, mating and variation. In a GA, competition is carried out by comparison and selection, and mating is carried out by genetic operators such as mutation and recombination (Rushton, 2004). A fitness function is used to evaluate individuals, and reproductive success varies with fitness. Over the last 20 years, the GAs have become very popular.

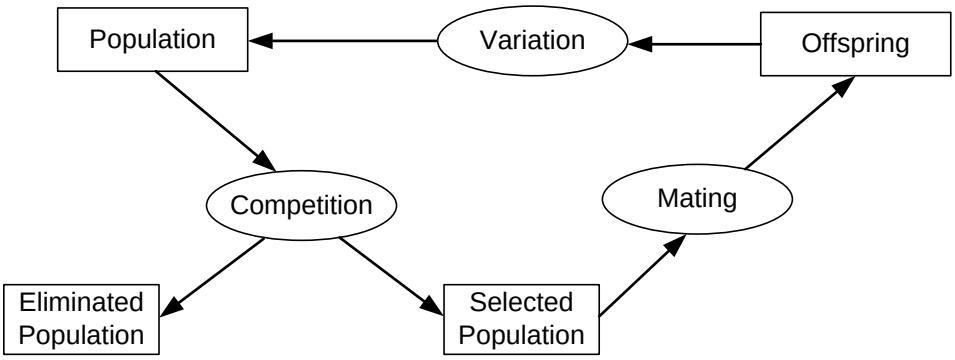


Figure 4-5: Organic evolution cycle (Reeves, 1993)

### 4.4.2 Study of a Genetic Algorithm in Railway Rescheduling

The GA can be described in a number of steps:

Step 1: 80 train sequences are generated for the initial GA population, and their costs are computed. The first sequence is the FCFS solution, while the other 79 are randomly generated.

In order to keep the competitiveness of the populations, the size of the initial population is normally chosen to be between 40 to 120, depending on the complexity of the problem. The number is kept the same for every generation (Rothlauf, 2006). In this thesis 80 sequences are selected. This has been arrived at through experimentation. It has also been found that using the FCFS solution increases the chances of finding a lower cost solution.

Step 2: The 40 lowest cost sequences are ordered from the lowest to the highest cost and saved as $G_1^1, G_2^1, G_3^1, \ldots, G_{40}^1$, where the superscript is the generation number and the subscript number is the rank.

Step 3: From the 40 lowest cost sequences, pairs of solutions are selected to parent a new solution until 80 offspring are produced. The new solutions are produced by using a two-point crossover operator, where, as shown in Figure 4-6, two points, X and Y, are randomly selected and the part of the solution

between these two points is swapped. Any illegal solutions (where trains are missing or repeated in an sequence) are further evolved to make legal solutions. The parents cannot be the same for an sequence, and each of the 80 new sequences must be unique.

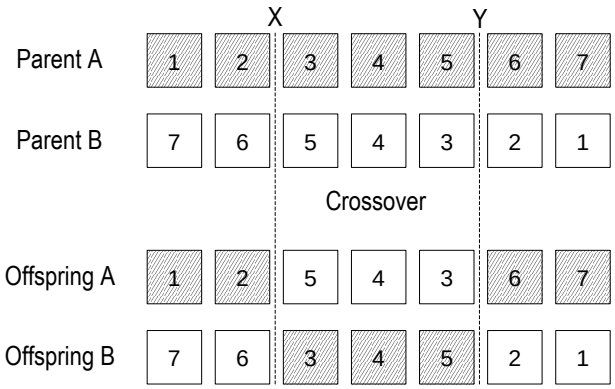


Figure 4-6: GA two-point crossover operator

In GA, the evolution of solutions not only depends on the selection operation, but is also likely to depend on the quality of the parents. In each pair, therefore, one of the parents is randomly selected from the five lowest cost sequences.

In GA studies, there are many crossover techniques, such as one-point crossover (as shown in Figure 4-7), two-point crossover (as shown in Figure 4-6), cut and splice crossover (as shown in Figure 4-8) and uniform crossover (as shown in Figure 4-9), etc. (Back, 1996).

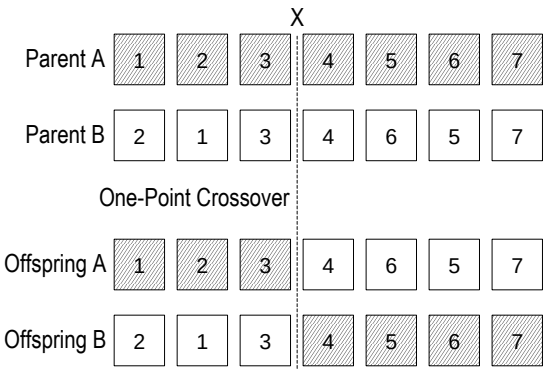


Figure 4-7: GA one-point crossover operator

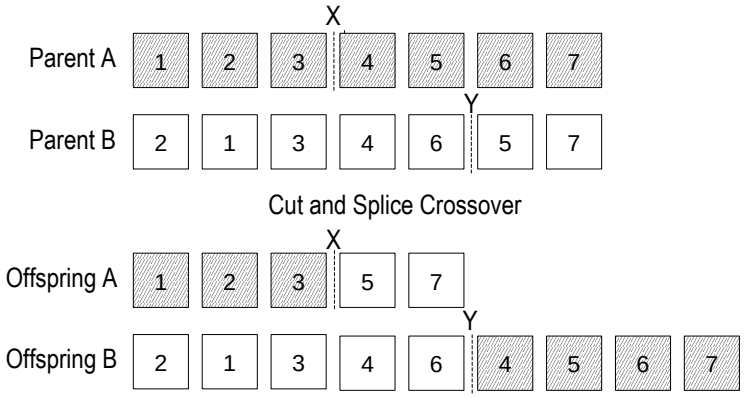


Figure 4-8: GA cut and splice crossover operator

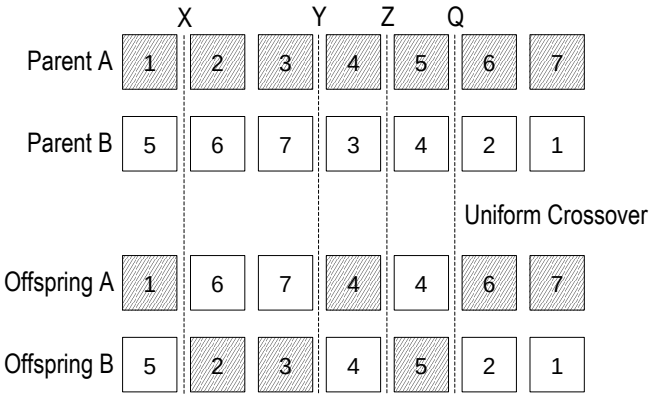


Figure 4-9: GA uniform crossover operator

The cut and splice crossover method cannot be used in railway rescheduling problems since it produces variable length solutions. The other three operators provide the correct length solutions but they may not provide legal solutions. For example, as shown in Figure 4-9, the offspring A and B, which are illegal solutions, are achieved

by using the uniform crossover operator. Trains 4, 6 and 7 are repeated twice, and trains 2, 3 and 5 are missing. Although these illegal solutions can be adjusted according to some rules, if the offspring include too many repeated trains and missed trains, the genetic algorithm is likely to lose its original meaning, which is the inheritance of the parents' characteristics. The two-point crossover operator, therefore, is used in this study since it provides the smallest change compared with the others. Nonetheless, a two-point crossover operator could still guarantee an illegal solution which includes repeated trains and missed trains. Therefore, a converter has been designed to restructure the illegal solution.

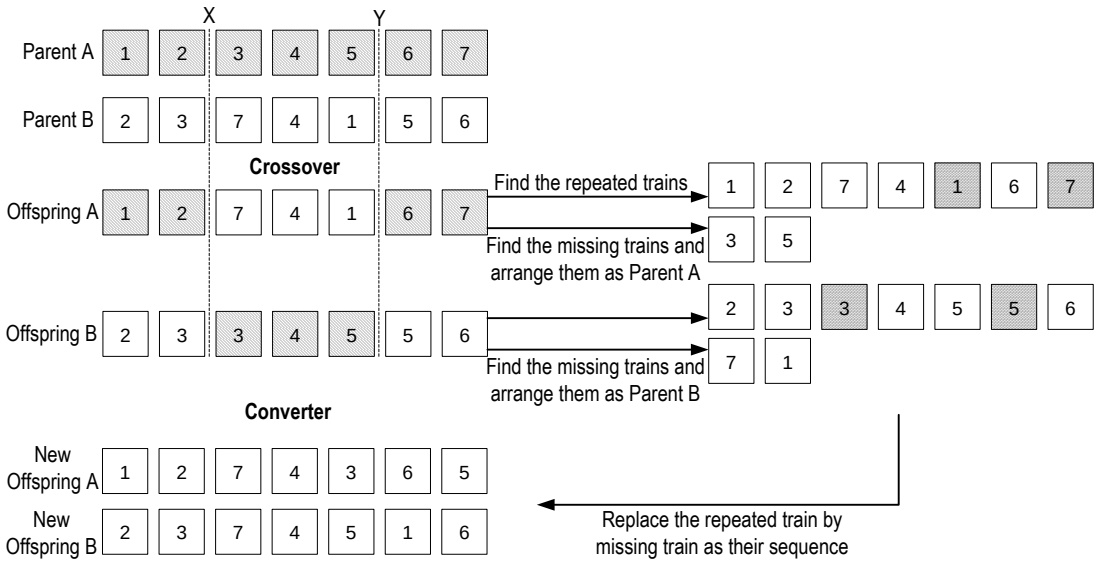


Figure 4-10: The GA infeasible offspring converter (Fan)

As Figure 4-10 shows, if an illegal solution is found by a GA two-point crossover operator, the repeated trains are found and their positions in this offspring sequence are recorded. Then, the missing trains are found and arranged according to the sequence of a parent; offspring A is referred to parent A and offspring B is referred to parent B. After this, the repeated trains are replaced by the missing train(s) according to the arranged sequence. Finally, since this converter only deals with repeated trains

and missing trains for the offspring, the solution feasibility is checked and any further solution adjustment is made if necessary.

Step 4: If the lowest cost solution has remained the same for five consecutive iterations, the algorithm is terminated, otherwise, go to step 2.

### 4.4.3 Results and Summary

The result of the GA applied to scenarios 1 to 4 is shown in Table 4-3. Optimal solutions are marked with an asterisk.

| | Computation Time (s) | Total Delay Cost (£) | Sequence and Delay Cost |
|---|---|---|---|
| **Scenario 1** | 93.349 | 90 * | Order: 2, 4, 3, 1, 5, 6, 7, 8, 9, 10, 11, 12<br>Delay: 1 (£60), 5 (£10), 8 (£20) |
| **Scenario 2** | 87.328 | 230 * | Order: 2, 3, 4, 5, 6, 7, 8, 1, 9, 10, 11, 12<br>Delay: 1 (£120), 9(40), 10(40), 11(£10), 12 (£20) |
| **Scenario 3** | 84.564 | 290 * | Order: 2, 3, 4, 5, 1, 6, 8, 9, 10, 11, 7, 12<br>Delay: 1(£80), 6(£20), 8(£20), 9(£40), 10(£40), 11(£10), 7(£40), 12(£40) |
| **Scenario 4** | 88.268 | 410 | Order: 8, 1, 2, 3, 4, 9, 5, 6, 7, 10, 12, 11<br>Delay: 1(£60), 2(£120), 3(£10), 4(£40), 9(£40), 5(£10), 6(£20), 7(£10), 10(£40), 12(£20), 11(£40) |

Table 4-3: Results of GA for each scenario

The GA provides the optimal solution in three out of the four scenarios.

## 4.5 Ant Colony Optimisation Algorithm

### 4.5.1 Introduction to Ant Colony Optimisation

The ant colony optimisation (ACO) algorithm was first proposed by Dorigo in his PhD thesis (Dorigo, 1992). In ACO algorithms, artificial ants construct solutions by probabilistically making a sequence of local decisions, following paths on a graph. At

each node on an ant's path, the ant follows one branch to extend the current partial solution (Dorigo *et al.*, 1999, Dorigo *et al.*, 2002). Each node is assigned a 'pheromone' coefficient, according to the cost of a solution passing through this node.

ACO has been studied in the railway domain in recent years, such as optimisation for rescheduling in the network (Zidi and Maouche, 2006), optimisation of train-speed trajectory and control by Ke *et al.* (2011) and freight train blocking problems by Yaghini *et al.* (2011). All these studies have a point in common, they all use a decision arc model, as shown in Figure 4-11.

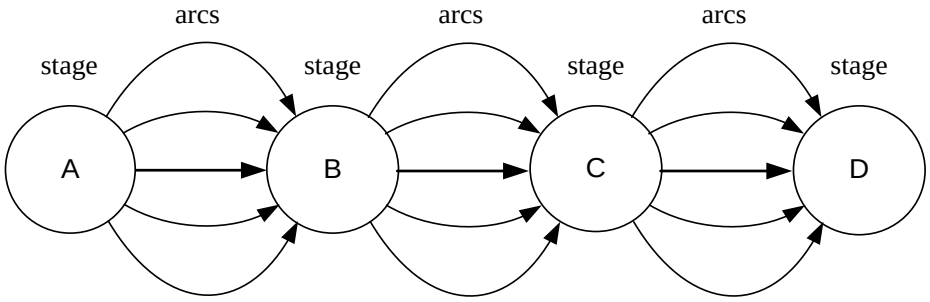


Figure 4-11: ACO decision arcs model

The decision arc model divides the processing of ant travel into several stages, and the optimal solution is found from the link of these optimal arcs between stages. For example, the arcs can be set as motoring, coasting and braking points when optimising train trajectories to minimise time and/or energy (Ke *et al.*, 2011); arcs can also be set as train block plans in a freight train blocking problem (Yaghini *et al.*, 2011). In the area of train rescheduling, Zidi and Maouche (2006) set arcs as the amount of time a train must wait at a red signal time and the stages are set as trains. However, this work can only find a conflict-free solution, not an optimal solution. A decision arc model cannot be used for finding an optimal rescheduling solution with

the purpose of minimising the delay cost. If train sequence is set as the stages, solutions that may be illegal or infeasible solution are likely to be created. Therefore, a decision tree is introduced here into ACO (Fan *et al.*, 2011), where the decision tree is constructed as described previously in Section 3.4.2.

### 4.5.2 Study of Ant Colony Optimisation Algorithm in Railway Rescheduling

The ant colony optimisation algorithm has been adapted to carry out a search on the decision tree for routing trains through a junction – called the Decision Tree Based Ant Colony Optimisation Algorithm (DTBACO). The full decision tree is constructed as the first step of this algorithm. Ants travel from the top node to one of the leaf nodes according to a probability related to the pheromone coefficient at each of the possible next nodes. Initially, the pheromone coefficients of all nodes in the tree are set to 1.

The FCFS solution is selected as the path of the first ant in the first iteration, while the remaining ants select paths randomly, as at this stage the pheromone coefficients of all nodes are the same. The number of ants remains fixed throughout the solution process; one iteration is considered complete when all ants have travelled on a path between the top node and a bottom node.

At the end of the iteration all paths are evaluated using the cost function and the pheromone coefficient of any node which has been visited is reduced. In this thesis we will use a pheromone coefficient decrease rate of 0.5. The pheromone coefficients for the nodes on the lowest cost path are set to 1.

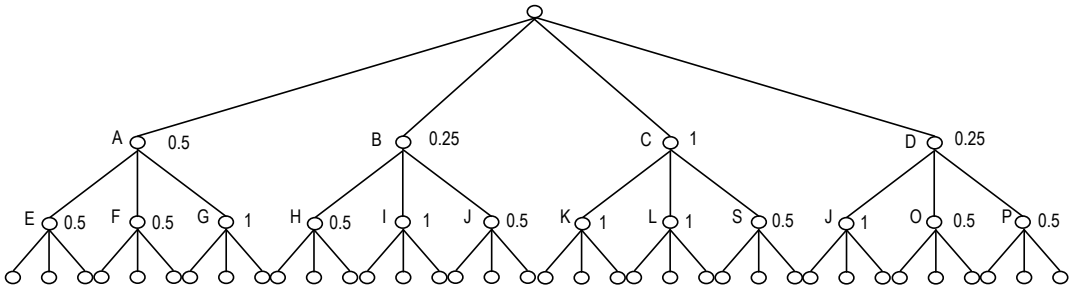


Figure 4-12: A partial decision tree representing ACO after two iterations (Fan)

During subsequent iterations each ant follows a path through the decision tree, choosing at each level the branch to be followed according to the pheromone coefficients on the potential next nodes. The probability of choosing one particular branch is in line with the pheromone coefficients of each potential next node. The probability, *r*, is chosen randomly between 0 and the sum of pheromone coefficients of all possible next nodes:

$$r = random\ (0, \sum_{i=1}^{n} \tau_i)$$

$$r \leq \tau_1 \text{---------- } node\ j_1\ is\ chosen$$

$$\sum_{1}^{k-1} \tau_i < r \leq \sum_{1}^{k} \tau_i \text{ ---------- } node\ j_k\ is\ chosen$$

where $\tau_i$ is the pheromone coefficient on each node, and $n$ is the total number of connected branches on the next level. For example, in Figure 4-12, the current pheromone coefficients are shown on the right of each node. For level 2 of this decision tree, each ant has a 1 in 4 probability of choosing *A*, a 1 in 8 probability of choosing *B*, a 1 in 2 probability of choosing *C* and a 1 in 8 probability of choosing *D* as its next step (Dorigo and Stützle, 2004).

Each iteration results in *n* ants choosing *n* paths from the top level node to the leaf level nodes, which aligns with *n* different sequences of trains through the junction

(two or more ants may follow identical paths, but this is unlikely in such a complex problem). These steps are repeated until convergence, which is defined as when the best solutions are the same for five consecutive iterations.

The number of ants used in the examples presented in this thesis has been selected in line with the travelling salesman problem considered by Dorigo and Stützle (2004). In this book, the authors suggest that the maximum number of ants is made equal to the number of cities in the travelling salesman problem; these can be considered to be comparable to the number of trains in the train rescheduling problem. The authors also suggest using 0.5 as the pheromone coefficient decrease rate. Other work by (Handl *et al.*, 2004) suggests undertaking experiments to refine parameter settings. By undertaking similar experiments, it was decided that the number of ants for the train rescheduling problem should be 10, while the pheromone coefficient decrease rate should be set to 0.5.

### 4.5.3 Results and discussion

The results are shown in Table 4-4. Optimal solutions are marked with an asterisk.

| | Computation Time (s) | Total Delay Cost (£) | Sequence and Delay Cost |
|---|---|---|---|
| **Scenario 1** | 78.942 | 90 * | Order: 2, 4, 3, 1, 5, 6, 7, 8, 9, 10, 11, 12<br>Delay: 1 (£60), 5 (£10), 8 (£20) |
| **Scenario 2** | 80.643 | 230 * | Order: 2, 3, 4, 5, 6, 7, 8, 1, 9, 10, 11, 12<br>Delay: 1 (£120), 9(40), 10(40), 11(£10), 12 (£20) |
| **Scenario 3** | 76.534 | 310 | Order: 2, 3, 4, 5, 6, 8, 9, 10, 11, 1, 7, 12<br>Delay: 1(£160), 7(£50), 12(£100) |
| **Scenario 4** | 73.326 | 370 * | Order: 8, 1, 2, 9, 3, 4, 5, 6, 7, 10, 12, 11<br>Delay: 1(£60), 2(£120), 3(£10),4(£40),5(£10), 6(£20), 7(£10), 10(£40), 12(£20), 11(£40) |

Table 4-4: Results of ACO algorithm for each scenario

In three of the scenarios, the optimal solution is found, in the other scenario (Scenario 3), the cost is only £20 greater than the lowest cost, found by BF. ACO has also been found to converge much quicker than the other search methods previously considered.

## 4.6 Comparisons and Discussions

Table 4-5 shows the delay cost results for all eight methods for the four scenarios considered in Chapter 3 and Chapter 4. In Table 4-5 the methods are ordered by the sum of the total solution cost over all four scenarios. Similarly, Table 4-6 shows the results of each method ordered by sum of the sum of the computation time over all four scenarios.

| Method | Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 |
|---|---|---|---|---|
| **BF** | £90 | £230 | £290 | £370 |
| **DP** | £90 | £230 | £290 | £370 |
| **SA** | £90 | £230 | £290 | £370 |
| **ACO** | £90 | £230 | £310 | £370 |
| **GA** | £90 | £230 | £290 | £410 |
| **TS** | £150 | £230 | £360 | £370 |
| **DTBE** | £90 | £300 | £310 | £570 |
| **FCFS** | £150 | £340 | £360 | £570 |

Table 4-5: The delay costs for all eight methods for each scenario

| Method | Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 |
|---|---|---|---|---|
| **FCFS** | 0.005 s | 0.005 s | 0.005 s | 0.005 s |
| **DTBE** | 70 s | 70 s | 70 s | 70 s |
| **TS** | 72 s | 68 s | 73 s | 69 s |
| **ACO** | 79 s | 81 s | 76 s | 73 s |
| **GA** | 93 s | 87 s | 84 s | 88 s |
| **SA** | 217 s | 223 s | 208 s | 228 s |
| **DP** | 318 s | 318 s | 318 s | 318 s |
| **BF** | 1692 s | 1692 s | 1692 s | 1692 s |

Table 4-6: The computation time for all eight methods for each scenario

Figure 4-13 shows the results for all eight methods for all four scenarios. Note the discontinuity in the x-axis from 350 to 1400 seconds.

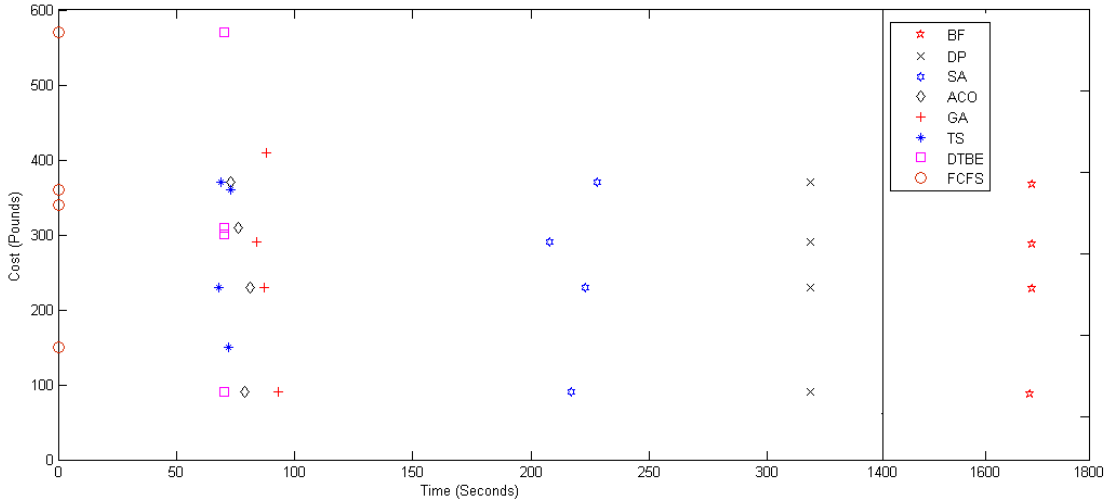


Figure 4-13: The delay cost vs computation time for all eight approaches for all four scenarios

BF and DP are both exhaustive search algorithms, and hence will always find the optimal solution. For all of the scenarios BF takes a significantly longer time than the DP algorithm to find the optimal solution. This is due to the manner in which the problem is formulated when using DP. However, both of these approaches are consistently slower than the other algorithms considered.

The evolutionary-based and graph-based approaches all perform relatively similarly. However, it can be seen that SA generally takes a longer time than the other algorithms to find a solution. The TS method is fast compared with the other evolutionary algorithms, but commonly finds a solution no better than FCFS. DTBE performs slightly better than the TS, but again, particularly as the scenarios become more complex, finds solutions only marginally better than FCFS. The GA commonly found similar results to the SA approach, but took less computation time. ACO gives the closest results to the optimum and has a computation time comparable with that of the fastest methods.

Apart from the length of computation time and amount of delay cost, there are two other important criteria for these approaches: terminability and controllability. When the rescheduling algorithm can be terminated at any time and the current best solution is available, an algorithm is referred to as being terminable. This is important because some algorithms have a high convergence time and, in train rescheduling scenarios, often only a limited response time is available. Of the approaches considered, BF, TS, SA, GA and ACO are terminable algorithms.

When the balance point between solution optimality and computation time can be controlled, the algorithm is referred to as being controllable. The following algorithms considered in this thesis are controllable: DTBE by adjusting the elimination percentage; SA by the temperature deceasing rate, and ACO through the pheromone coefficient reduction rate.

## 4.7 Conclusions

Chapter 3 and Chapter 4 compare eight different algorithms for the optimisation of train sequence through railway junctions when a timetable becomes disturbed. Using an example junction area, four scenarios were considered and the results are compared.

TS, as for the other evolutionary algorithms, cannot guarantee to find the optimal solution. For the four scenarios considered, the average delay cost using the TS algorithm is the highest of the four evolutionary algorithms. However, TS has the shortest computation time, compared with the other evolutionary algorithms. Therefore, TS can be considered suitable for dealing with large rescheduling problems

where computation time is important, such as complex junction or station rescheduling.

SA gives the optimal solution for all four scenarios considered; however, it is not expected to find the optimal solution in every scenario. SA takes the longest computation time, compared with the other evolutionary algorithms considered.

In addition, although SA is terminable and controllable, it will not necessarily provide a good solution if the algorithm is terminated at an early stage. Furthermore, SA generally takes a long computation time for simpler scenarios, which can make it unsuitable for the railway rescheduling problem.

GA provides a good solution with a reasonable computation time in the four scenarios used in this study. Since GA evaluates 80 sequences in each generation and saves the best sequences, it can give a good solution if the algorithm is terminated at any time. GA is also a strong, independent method; it does not require a parallel programming which needs updating in every generation, such as a tabu list or annealing temperature. Therefore, GA is not only suitable to be used singly in a railway timetable rescheduling application, but it is also suitable for parallel working with other algorithms or it could be part of a hybrid algorithm which could improve the searching efficiency. For example, Wegele *et al.* (2008) propose an algorithm which hybridises GA with a branch and bound algorithm for railway automatic rescheduling, and Zheng *et al.* (2006) propose an algorithm with parallel GA and SA for task schedules. Both of the results proved better than using a single algorithm.

ACO gives the closest solution to the optimum and takes the second shortest computation time of the evolutionary algorithms in these four scenarios; ACO gives

the optimal solution in three quarters of these scenarios. Compared with GA, ACO achieved a better solution with a shorter computation time. Furthermore, ACO is not only a terminable algorithm, but also a controllable algorithm, so ACO is appropriate to be implemented at any junction and in any scenario. Therefore, it can be concluded that ACO is the most appropriate algorithm for further research and implementation.

# Chapter 5: Single Junction Rescheduling with Hybrid Algorithm

In the last chapter, ACO has been shown to perform well in single junction rescheduling. Following this approach, in order to find more consistently a better solution which is optimum or close to optimum, while to reduce the computation time, a new algorithm is designed and introduced in this chapter that improves the searching efficiency. The author considers a modification of ACO that has a reduced computation time and also tends to provide good solutions.

## 5.1 Introduction

### 5.1.1 Local Minima of ACO

Based on the discussion in the last chapter the performance of ACO offers a good compromise between convergence speed and quality of solution. It has been found to converge in a short time relative to the other tested evolutionary algorithms (it is the second fastest of the four tested), whilst on average it gives the second best solutions, relative to the optimum, found by BF. In the ACO case study with 12 trains, the decision tree has 13 levels. A reasonable computation time is obtained because all but one of the pheromone coefficients on the nodes on levels 2—4 reduce rapidly, making it unlikely that different nodes to the FCFS solution will be selected on levels 2—4. This effectively reduces the size of the search space at an early stage.

However, the main disadvantage of ACO is also obvious: the optimal path might start with a choice of nodes that initially have low pheromone coefficients. As an example, the first four levels of a decision tree and corresponding pheromone coefficients after three complete iterations are shown in Figure 5-1.

In this example, the attempted best sequences always starts with the third node of the second level, which then retains the pheromone coefficient "1", while the pheromone coefficients of the other three nodes of the second level are reduced three times to "0.125". However, the optimal solution is illustrated as a dashed line, which starts the pheromone coefficient "0.125" of its first node. After three iterations, the possibility of choosing this node is only 9%, and the possibility of choosing the whole optimal path is much lower than this. At the same time, the possibility of choosing the third node which has pheromone coefficient "1" is 73%. Therefore, at the termination of the algorithm, the best solution is likely to be found to start with this node. For this reason, the best solution found by ACO can frequently be a local minimum rather than the global minimum.

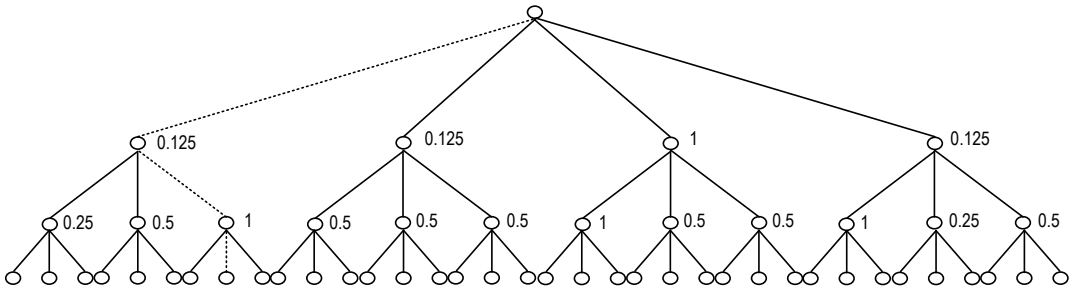


Figure 5-1: Partial decision tree with pheromone coefficients after three iterations

In theory, decreasing the pheromone coefficient decrease rate can enlarge the search space of ACO, which gives a higher probability of finding the optimum. However, this is at the cost of increasing the computation time. An improvement is introduced

here which is capable of finding a better solution, without increasing the computation time.

### 5.1.2 Methodology of Stepping Out From Local Minima in ACO

The local minima problem of ACO also has been found in many other studies, such as travelling salesman problems (Shang *et al.*, 2007) and the quadratic assignment problem (Qi *et al.*, 2009). Researchers have proposed to use a new algorithm which could hybridise with ACO, in order to step out from local minima, without increasing the search time.

Shang *et al.* (2007) propose GA hybridised with ACO for travelling salesman problems. The authors also tried the standard SA, the standard GA and the standard ACO for the same problem. The hybrid algorithm was shown to be more effective than the three above-mentioned algorithms. Furthermore, there are many other researchers who show hybrid ACO and GA to provide a more effective search. Chen *et al.* (2005) and Liu and Meng (2008) studied this hybrid algorithm for continuous-space/domain optimisation problems. However, none of these ACO applications are based on a decision tree.

In the mathematical model of ACO used in this thesis, a GA could aggravate decision tree based ACO into dropping into local minima.

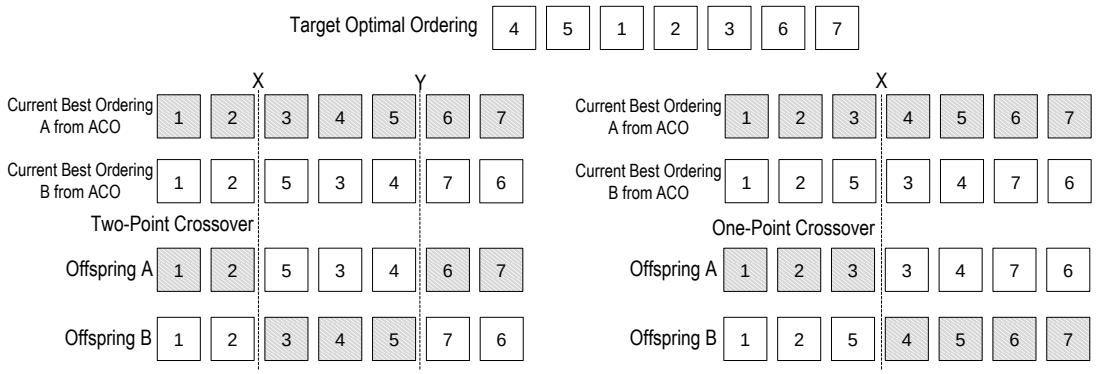


Figure 5-2: Example of ACO-GA hybrid where the GA uses the results of ACO

When ACO drops into local minima, the first several trains of the current best sequences tend to be the same. Using any kind of GA crossover operator is likely to only swap the middle part or the latter part of the current sequences. An example is given in Figure 5-2; GA could change the first several trains of the current best sequences with a very low probability (for two-point crossover, the first train will be changed 8% of the time; the second train will be changed 16% of the time). Furthermore, after the processing of the GA, every time the pheromone coefficients of the new current best solution will be updated to 1. GA, therefore, increases the likelihood of ACO dropping into local minima. Therefore, another algorithm is introduced to enable the solution to escape from local minima, which is a type of local search.

Gambardella and Dorigo (2000), who first proposed the concept of ACO, describe a hybrid ACO and local search algorithm for improving the search quality of ACO without increasing the computational time and complexity. The authors also apply a hybrid algorithm, which is built on a feasible solution model similar to a decision tree model, in travelling salesmen problems. The random swapping nature of local search can help ACO to step out from local minima, and the hybrid algorithm works much

more effectively than using standard ACO. Furthermore, Qi *et al.* (2009) analyse the constructive procedure of ACO and local search algorithm, and then the authors describe the complementarities of the two algorithms. They then demonstrate the results of the hybrid of local search and ACO, which is found to be competitive in travelling salesmen problems.

### 5.1.3 Review of Local Search and Its Local Minima

As stated in the introduction to local search in Chapter 3 (Section 3.1), the solution of local search can also frequently be a local minimum rather than the global minimum. This local minimum, nevertheless, is very different from the local minimum found by ACO.

In a local search algorithm, a local minimum means that there is no better solution which can be found in one iteration from the current solution, but some solution exists which is better; the current solution is then called the local minimum of a local search.

## 5.2 Hybrid Concept of ACO and Local Search

Based on the different strengths and weaknesses of the two algorithms, the shortcomings can cancel each other out by hybridising the algorithms. The local search is inserted between iterations of the ACO.

The starting point for the local search is the current best solution, which may or may not be trapped in a local minimum. If the ACO search has become trapped in a local minimum, the random nature of local search within the hybrid algorithm will potentially allow the search to step out of the local minimum, which would not be possible by ACO alone.

From the current best solution generated by ACO, a number of neighbouring solutions, $k$, is generated. From each of these, a further neighbouring solution is generated, resulting in $2k$ new potential solutions. Their costs are evaluated and the pheromones updated in the same way as in the ACO algorithm. By this hybridising, the algorithm can take full advantage of the rapid convergence of ACO, while stepping out from local minima by using local search.

In selecting an algorithm to hybridise with ACO, the main aim is to be able to step away from local minima into different areas of the search space. The simplicity of local search is advantageous because it can provide this function quickly and simply. If another of the evolutionary algorithms such as TS or SA were selected for use in the hybrid algorithm it would be less efficient. If SA were selected, the algorithm would certainly be slowed down, while the lack of randomness of TS would prove inefficient in moving away from local minima.

## 5.3 Study of Hybrid Algorithm

The steps for the hybrid algorithm are given below.

Step 0: Set the number of iterations, $g = 1$

Construct the decision tree. Set all pheromone coefficients to 1.

Select the FCFS sequence and 9 other randomly selected possible sequences. Then go to step 1 (step 0 is only the start of this algorithm).

This step is selection of the initial solutions. Since this hybrid algorithm will also be tested using the case study introduced in Chapter 2, 10 ants per iteration is still used here, and the FCFS sequence is selected as one of the initial sequences.

Step 1: From the top node, the branches are selected one by one until the bottom node is reached.

To select each node a random number, r, is chosen, $r = random\ (0, \sum_{i=1}^{n} \tau_i)$

$r \leq \tau_1$---------- *node* $j_1$ *is chosen*

$\sum_1^{k-1} \tau_i < r \leq \sum_1^k \tau_i$ ---------- *node* $j_k$ *is chosen*

where $\tau_i$ is the pheromone on the node $j_i$, and $n$ is the total number of branches on the next lower level.

This step is exactly the same as ACO, which is introduced in Chapter 4 (Section 4.5).

Step 2: Evaluate the cost of each sequence, $J(\theta_1)$ to $J(\theta_{10})$.

Identify the sequence with the lowest cost: $\hat{\theta}$, and list the decision tree nodes of this sequence.

List all decision tree nodes of all other evaluated sequences. Divide their corresponding pheromone coefficients by 2.

The pheromone coefficients of nodes on the $S_l$ path are set to 1.

This step is also the same as ACO, and with the intention of taking full advantage of rapid convergence, the pheromone coefficient decrease rate is set to 0.5.

Step 3: Five neighbourhood sequences of $\hat{\theta}$ are chosen: $N_1 = N(\hat{\theta}), N_2 = N(\hat{\theta}), \cdots\cdots, N_5 = N(\hat{\theta})$. A neighbourhood sequence is selected by swapping two trains in the sequence randomly. The feasibilities of these five neighbourhood sequences are checked, and infeasible sequences are converted using the method described in Section 4.1.2. -

Another neighbourhood sequence of each of the last five is chosen randomly: $N_6 = N(N_1), N_7 = N(N_2), \cdots\cdots, N_{10} = N(N_5)$. Any infeasible sequence is converted.

Evaluate the cost of these 10 sequences.

Identify the lowest cost sequence: $\hat{\theta}'$

Divide by 2 the pheromone coefficients of the nodes visited on 10 tree paths corresponding to $N_1$ to $N_{10}$.

If $J(\hat{\theta}) > J(\hat{\theta}')$, list the decision tree nodes of $\hat{\theta}'$, and set its pheromone coefficients to 1.

This step is the local search part of the hybrid, which potentially helps ACO to step out from its local minima. In hybrid algorithm theory, the local search should be based on the current best solution of ACO, and the number of local search individuals should be the same as the ACO (Dorigo and Stützle, 2004), which is 10 individuals per iteration. In order to increase the possibility of stepping out from the local minima of ACO, the local search is done twice on 5 individual solutions. The first five new individuals are from the current best solution of ACO, and the other five are based on these five individuals.

As noted before, local search may lead to infeasible sequences, therefore the infeasible-feasible converter is used here. Then the best solution from these 10 new solutions is selected and compared with the best ACO solution; the better of these two solutions is kept for later use and the pheromone coefficients of all nodes are updated based on this.

Step 4: Increment g by 1.

Go back to step 1 unless the best solution has remained the same for 5 consecutive iterations, in which case, terminate the algorithm.

Termination criteria of the same solution for 3, 4, 5, 6, 7, 8, 9 and 10 iterations were tested, before adopting 5.

## 5.4 Results and Discussion: Case Study Scenarios

In this section, the hybrid algorithm is applied to the four case study scenarios that were introduced in Section 2.5. The result of the hybrid algorithm in each case is shown in Table 5-1. Optimal solutions are marked with an asterisk.

| | Computation time (s) | Total delay cost (£) | Sequence and Delay cost |
|---|---|---|---|
| **Scenario 1** | 68.236 | 90 * | Order: 2, 4, 3, 1, 5, 6, 7, 8, 9, 10, 11, 12<br>Delay: 1 (£60), 5 (£10), 8 (£20) |
| **Scenario 2** | 74.891 | 230 * | Order: 2, 3, 4, 5, 6, 7, 8, 1, 9, 10, 11, 12<br>Delay: 1 (£120), 9(£40), 10(40), 11(£10), 12 (£20) |
| **Scenario 3** | 80.274 | 290 * | Order: 2, 3, 4, 5, 1, 6, 8, 9, 10, 11, 7, 12<br>Delay: 1(£80), 6(£20), 8(£20), 9(£40), 10(£40), 11(£10), 7(£40), 12(£40) |
| **Scenario 4** | 77.402 | 370 * | Order: 8, 1, 2, 9, 3, 4, 5, 6, 7, 10, 12, 11<br>Delay: 1(£60), 2(£120), 3(£10),4(£40),5(£10), 6(£20), 7(£10), 10(£40), 12(£20), 11(£40) |

Table 5-1: Results of hybrid algorithm for each scenario

The hybrid algorithm gives the optimal solution in each scenario. The delay costs and computation times are compared with BF, local search algorithm and standard ACO, and are shown in Table 5-2 and Table 5-4.

| Method | Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 |
|---|---|---|---|---|
| **BF** | £90 | £230 | £290 | £370 |
| **Local Search** | £150 | £300 | £360 | £370 |
| **ACO** | £90 | £230 | £310 | £370 |
| **Hybrid Algorithm** | £90 | £230 | £290 | £370 |

Table 5-2: The delay costs for the hybrid algorithm for each scenario

| Method | Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 |
|---|---|---|---|---|
| **BF** | 1692 s | 1692 s | 1692 s | 1692 s |
| **Local Search** | 91 s | 82 s | 93 s | 79 s |
| **ACO** | 79 s | 81 s | 76 s | 73 s |
| **Hybrid Algorithm** | 68 s | 75 s | 80 s | 77 s |

Table 5-3: The computation time for the hybrid algorithm for each scenario

Compared with the delay cost of standard local search and standard ACO, the hybrid algorithm gives a better solution. From a computation time point of view, although the computation time of the hybrid algorithm is slightly longer than standard ACO in scenarios 3 and 4, the average computation time of the hybrid algorithm over the four scenarios is the shortest compared with the standard local search algorithm and standard. For these four scenarios, the average computation time of the hybrid algorithm is 75 s; the average computation time of ACO is 77 s.

## 5.5 Results and Discussion: 100 Random Scenarios of the Case Study

As a simple statistical test, 100 different delay scenarios were randomly generated by choosing a delay for each train from independent uniform distributions from -4 to 9 minutes (negative value means a train running before its timetabled time). This interval was selected by referring back to Figure 1-1, where it is shown that about 80%

of trains arrive at their destination within a interval of early arrival by 4 minutes to late arrival by 9 minutes. This range of delays is large enough, compared to the spacing between trains in the nominal timetable, to cause some trains to approach the junction in a different order from normal. The train positions and the headways are checked in each random scenario; if two or more trains overlap or have an illegal headway, the following trains are adjusted to the legal headway position with the corresponding number of delay minutes.

The results of rescheduling each randomly generated delay scenario using the hybrid algorithm, BF, ACO and FCFS methods are shown in Table 5-4. Four differences are calculated and are also shown in Table 5-4:

$$D1 = cost\ obtained\ using\ FCFS\ - minimum\ cost\ from\ BF\ method$$

$$D2 = cost\ obtained\ using\ the\ ACO - minimum\ cost\ from\ BF\ method$$

$$D3 = cost\ obtained\ using\ the\ hybrid\ algorithm - minimum\ cost\ from\ BF\ method$$

$$D4 = cost\ obtained\ using\ the\ hybrid\ algorithm - cost\ obtained\ using\ the\ ACO$$

| **Scenario** | BF | FCFS | ACO | Hybrid | D1 | D2 | D3 | D4 |
|---|---|---|---|---|---|---|---|---|
| **1** | 290 | 320 | 290 | 290 | 30 | 0 | 0 | 0 |
| **2** | 380 | 410 | 400 | 400 | 30 | 20 | 20 | 0 |
| **3** | 150 | 300 | 150 | 150 | 150 | 0 | 0 | 0 |
| **4** | 50 | 380 | 50 | 50 | 330 | 0 | 0 | 0 |
| **5** | 340 | 400 | 340 | 340 | 60 | 0 | 0 | 0 |
| **6** | 200 | 220 | 200 | 200 | 20 | 0 | 0 | 0 |
| **7** | 290 | 420 | 320 | 290 | 130 | 30 | 0 | 30 |
| **8** | 290 | 390 | 390 | 290 | 100 | 100 | 0 | 100 |
| **9** | 180 | 180 | 180 | 180 | 0 | 0 | 0 | 0 |
| **10** | 280 | 380 | 300 | 280 | 100 | 20 | 0 | 20 |
| **11** | 270 | 340 | 320 | 320 | 70 | 50 | 50 | 0 |
| **12** | 230 | 230 | 230 | 230 | 0 | 0 | 0 | 0 |
| **13** | 90 | 120 | 110 | 90 | 30 | 20 | 0 | 20 |
| **14** | 220 | 230 | 230 | 220 | 10 | 10 | 0 | 10 |
| **15** | 320 | 380 | 370 | 320 | 60 | 50 | 0 | 50 |
| **16** | 180 | 230 | 180 | 180 | 50 | 0 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **17** | 310 | 320 | 320 | 310 | 10 | 10 | 0 | 10 |
| **18** | 290 | 360 | 360 | 360 | 70 | 70 | 70 | 0 |
| **19** | 410 | 420 | 410 | 410 | 10 | 0 | 0 | 0 |
| **20** | 130 | 310 | 230 | 130 | 180 | 100 | 0 | 100 |
| **21** | 150 | 340 | 150 | 150 | 190 | 0 | 0 | 0 |
| **22** | 170 | 230 | 180 | 170 | 60 | 10 | 0 | 10 |
| **23** | 20 | 100 | 60 | 20 | 80 | 40 | 0 | 40 |
| **24** | 270 | 270 | 270 | 270 | 0 | 0 | 0 | 0 |
| **25** | 240 | 360 | 330 | 270 | 120 | 90 | 30 | 60 |
| **26** | 160 | 290 | 200 | 160 | 130 | 40 | 0 | 40 |
| **27** | 180 | 300 | 180 | 180 | 120 | 0 | 0 | 0 |
| **28** | 200 | 390 | 390 | 200 | 190 | 190 | 0 | 190 |
| **29** | 140 | 410 | 140 | 140 | 270 | 0 | 0 | 0 |
| **30** | 100 | 350 | 310 | 100 | 250 | 210 | 0 | 210 |
| **31** | 50 | 140 | 110 | 50 | 90 | 60 | 0 | 60 |
| **32** | 60 | 310 | 230 | 190 | 250 | 170 | 130 | 80 |
| **33** | 400 | 420 | 410 | 400 | 20 | 10 | 0 | 10 |
| **34** | 210 | 330 | 230 | 210 | 120 | 20 | 0 | 20 |
| **35** | 320 | 410 | 320 | 320 | 90 | 0 | 0 | 0 |
| **36** | 140 | 400 | 290 | 140 | 260 | 150 | 0 | 150 |
| **37** | 10 | 270 | 230 | 10 | 260 | 220 | 0 | 220 |
| **38** | 260 | 260 | 260 | 260 | 0 | 0 | 0 | 0 |
| **39** | 160 | 200 | 180 | 160 | 40 | 20 | 0 | 20 |
| **40** | 280 | 280 | 280 | 280 | 0 | 0 | 0 | 0 |
| **41** | 360 | 420 | 380 | 360 | 60 | 20 | 0 | 20 |
| **42** | 50 | 230 | 50 | 50 | 180 | 0 | 0 | 0 |
| **43** | 250 | 390 | 350 | 250 | 140 | 100 | 0 | 100 |
| **44** | 250 | 330 | 260 | 250 | 80 | 10 | 0 | 10 |
| **45** | 220 | 320 | 290 | 220 | 100 | 70 | 0 | 70 |
| **46** | 200 | 290 | 200 | 200 | 90 | 0 | 0 | 0 |
| **47** | 410 | 420 | 420 | 410 | 10 | 10 | 0 | 10 |
| **48** | 80 | 380 | 250 | 80 | 300 | 170 | 0 | 170 |
| **49** | 240 | 360 | 240 | 240 | 120 | 0 | 0 | 0 |
| **50** | 270 | 310 | 310 | 270 | 40 | 40 | 0 | 40 |
| **51** | 20 | 130 | 110 | 20 | 110 | 90 | 0 | 90 |
| **52** | 270 | 300 | 300 | 270 | 30 | 30 | 0 | 30 |
| **53** | 360 | 400 | 360 | 360 | 40 | 0 | 0 | 0 |
| **54** | 170 | 370 | 240 | 170 | 200 | 70 | 0 | 70 |
| **55** | 190 | 380 | 220 | 190 | 190 | 30 | 0 | 30 |
| **56** | 40 | 350 | 150 | 80 | 310 | 110 | 40 | 70 |
| **57** | 250 | 290 | 280 | 250 | 40 | 30 | 0 | 30 |
| **58** | 370 | 410 | 410 | 370 | 40 | 40 | 0 | 40 |
| **59** | 90 | 300 | 110 | 90 | 210 | 20 | 0 | 20 |
| **60** | 40 | 130 | 110 | 40 | 90 | 70 | 0 | 70 |
| **61** | 80 | 210 | 80 | 80 | 130 | 0 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **62** | 290 | 350 | 340 | 290 | 60 | 50 | 0 | 50 |
| **63** | 310 | 370 | 340 | 310 | 60 | 30 | 0 | 30 |
| **64** | 270 | 350 | 270 | 270 | 80 | 0 | 0 | 0 |
| **65** | 200 | 270 | 200 | 200 | 70 | 0 | 0 | 0 |
| **66** | 140 | 220 | 150 | 140 | 80 | 10 | 0 | 10 |
| **67** | 270 | 380 | 280 | 270 | 110 | 10 | 0 | 10 |
| **68** | 240 | 410 | 280 | 240 | 170 | 40 | 0 | 40 |
| **69** | 280 | 360 | 300 | 280 | 80 | 20 | 0 | 20 |
| **70** | 60 | 270 | 190 | 60 | 210 | 130 | 0 | 130 |
| **71** | 290 | 390 | 290 | 290 | 100 | 0 | 0 | 0 |
| **72** | 290 | 340 | 340 | 340 | 50 | 50 | 50 | 0 |
| **73** | 400 | 410 | 410 | 400 | 10 | 10 | 0 | 10 |
| **74** | 350 | 390 | 350 | 350 | 40 | 0 | 0 | 0 |
| **75** | 80 | 400 | 170 | 80 | 320 | 90 | 0 | 90 |
| **76** | 400 | 410 | 410 | 400 | 10 | 10 | 0 | 10 |
| **77** | 90 | 300 | 180 | 90 | 210 | 90 | 0 | 90 |
| **78** | 100 | 250 | 130 | 100 | 150 | 30 | 0 | 30 |
| **79** | 10 | 240 | 160 | 10 | 230 | 150 | 0 | 150 |
| **80** | 130 | 140 | 130 | 130 | 10 | 0 | 0 | 0 |
| **81** | 120 | 360 | 130 | 120 | 240 | 10 | 0 | 10 |
| **82** | 250 | 370 | 270 | 250 | 120 | 20 | 0 | 20 |
| **83** | 170 | 220 | 180 | 170 | 50 | 10 | 0 | 10 |
| **84** | 0 | 80 | 0 | 0 | 80 | 0 | 0 | 0 |
| **85** | 170 | 360 | 250 | 170 | 190 | 80 | 0 | 80 |
| **86** | 70 | 270 | 240 | 70 | 200 | 170 | 0 | 170 |
| **87** | 350 | 400 | 370 | 350 | 50 | 20 | 0 | 20 |
| **88** | 70 | 290 | 70 | 70 | 220 | 0 | 0 | 0 |
| **89** | 290 | 390 | 370 | 290 | 100 | 80 | 0 | 80 |
| **90** | 320 | 410 | 320 | 320 | 90 | 0 | 0 | 0 |
| **91** | 0 | 40 | 0 | 0 | 40 | 0 | 0 | 0 |
| **92** | 60 | 100 | 80 | 60 | 40 | 20 | 0 | 20 |
| **93** | 20 | 310 | 150 | 20 | 290 | 130 | 0 | 130 |
| **94** | 340 | 380 | 350 | 340 | 40 | 10 | 0 | 10 |
| **95** | 20 | 280 | 20 | 20 | 260 | 0 | 0 | 0 |
| **96** | 280 | 370 | 300 | 280 | 90 | 20 | 0 | 20 |
| **97** | 200 | 350 | 250 | 230 | 150 | 50 | 30 | 20 |
| **98** | 340 | 340 | 340 | 340 | 0 | 0 | 0 | 0 |
| **99** | 160 | 220 | 170 | 160 | 60 | 10 | 0 | 10 |
| **100** | 20 | 80 | 40 | 20 | 60 | 20 | 0 | 20 |
| **Average** | **200.8** | **309.9** | **240.7** | **205.0** | **109.1** | **39.9** | **4.2** | **35.7** |
| **Occurrence Frequency of "0"** | | | | | **6** | **32** | **92** | **36** |

Table 5-4: The costs of different algorithms across 100 delay scenarios, in £

The average computation time of using each algorithm is shown in Table 5-5.

| Algorithm | BF | FCFS | ACO | Hybrid |
|---|---|---|---|---|
| **Average Computation Time** | 1692 | 0.005 | 74 | 72 |

Table 5-5: The average computation time of different algorithms in 100 delay scenarios, in seconds

The BF results give the lowest possible cost for each scenario, as all 369,600 possible solutions were checked. This computation took 1692 seconds. The minimum cost given by BF is not necessarily £0 – in these 100 scenarios, only two of the initial delays can be fully recovered. The hybrid algorithm never gives a higher cost than FCFS, as FCFS is used as one of the initial sequences in this algorithm. In some scenarios, the BF, FCFS, ACO and hybrid algorithms give the same cost. In these cases the hybrid algorithm necessarily stops after the first 5 iterations as FCFS is the optimum result and is also one of the starting sequences.

In average value analysis, the average cost for BF is £201, for FCFS it is £310, for ACO it is £241 and for the hybrid algorithm it is £205. The BF cost is the best that can be achieved for each scenario. The average hybrid algorithm cost is very close to that of BF, but the average computation time is improved by a factor of 23.5. The average cost for the hybrid algorithm is £35.7 lower than ACO, equivalent to a 14.8% improvement. The computation time is slightly better on average; 72 seconds for the hybrid algorithm compared to 74 seconds for ACO.

By using probability analysis, in Table 5-4, the occurrence frequency of "0" for D1, D2 and D3 is 6, 32 and 92. FCFS has a possibility of 6% of achieving the optimal solution; for ACO the possibility is 32% and the hybrid algorithm has a possibility of 92% of achieving the optimal solution. The occurrence frequency of "0" for D4 is 36,

which means that the hybrid algorithm has a possibility of 64% of selecting a better solution than ACO.

In the worst case analysis, the largest difference between BF and the hybrid algorithm is in scenario 32 (£130). Although the result of the hybrid algorithm is £190, which is about three times as high as the BF results (£60), the result of the hybrid algorithm is still much better than the FCFS result (£310) and significantly (25.8%) better than the ACO results (£230) for the same scenario.

## 5.6 Conclusions

The ACO algorithm easily becomes trapped in local minima. Generally, this is because all but one of the pheromone coefficients of the upper level nodes decrease significantly within a few iterations, resulting in all but one potential solution paths being unlikely to be tried later on.

The local search gives a change in train sequence, moving to a different part of the search space, but on its own the local search also tends to get trapped into local minima. The hybridisation of DTBACO with the local search helps to escape local minima.

A comparison between ACO, BF, and FCFS using randomised train delays shows the improvement that is possible using the hybrid approach. Over the 100 scenarios considered here, the hybrid algorithm has a possibility of 92% of achieving the optimal solution, and it gives an average cost very close to BF and much better than FCFS. Furthermore, this hybrid algorithm only takes 4% of the computing time of BF. In terms of the quality of the solution, the hybrid algorithm shows a significant improvement over ACO alone, while the speed of convergence remains similar.

# Chapter 6: Optimal Multi-junction Rescheduling and Rescheduling Region

As is well known, a delayed train can cause knock-on delays, which may lead to a whole network delay. An optimal rescheduling decision of the trains at one junction area may not be the optimal decision for the whole network. Therefore, a new approach is introduced in this chapter to find the optimal solution for multi-junction train rescheduling.

## 6.1 Introduction to Multi-junction Rescheduling

### 6.1.1 Difference between Single-junction Rescheduling and Multi-junction Rescheduling

As discussed in the last few chapters, the aim of optimal single-junction rescheduling is to find the optimal train sequence for a single junction area. However, trains may have one or more conflicts at another junction area during the remainder of their journey after the single junction rescheduling has taken place.

In the simulator used in this thesis, single junction rescheduling gives no further intelligent instructions to the trains after they have passed the junction area under consideration, and it is assumed that they travel to their destination according to their legal possible maximal speed. The newly rescheduled order and relative positions of trains may be inappropriate for following junctions and could cause a more serious

conflict than at the original junction area. Only considering one junction area may therefore lead to a higher overall network delay cost, which is produced by disturbed trains at other junctions.

Optimal multi-junction rescheduling (OMJR) takes a control region into account, within which there may be more than one junction area. A multi-junction train rescheduling system optimises a series of rescheduling solutions with the aim of finding the lowest delay cost for all disturbed trains which travel through this region. OMJR emphasises reducing the delay of the whole control region rather than reducing individual junction area delay.

### 6.1.2 Review of Multi-junction Rescheduling Approaches

Multi-junction rescheduling is an extremely complex problem, and it cannot be fully described by one decision tree. The number of possible solutions for a multi-junction rescheduling problem is more than the product of the number of solutions for each single junction. There is not much published literature focused on this problem. While it seems clear that Rule Based (RB) approaches and alternative graph based approaches can give a conflict-free solution for multi-junction rescheduling or even whole network control (Cheng, 1998b, D'Ariano, 2008), they frequently do not provide optimal or even close to optimal solutions. Therefore, agent-based modelling methods are proposed to break down the multi-junction rescheduling problem.

The majority of reported multi-junction rescheduling is based on the agent-based modelling system, which is composed of decentralised individual agents (e.g. train control centres) and exchanges of their localised information (Teodorovic, 2008). This localised information exchange uses the achieved results from each agent as inputs to

the multi-junction rescheduling problem. Davidsson *et al*. (2005) introduces a survey of existing research on agent-based approaches to traffic control, and concludes that agent-based approaches are suitable for the traffic control domain.

In the railway multi-junction rescheduling domain, Proenca and Oliveira (2004) suggest a multi-agent system for communications based railway traffic control. Their modelling system is composed of two independent layers: 'Control' and 'Learning'. 'Control' is responsible for single-junction rescheduling and station control. 'Learning' considers an optimal global solution by coordinating the results of the 'Control' layer. Chou (2009) introduces a similar rescheduling approach by using the multi-agent train traffic control principle for solving the multi-junction rescheduling problem. In his study, the multi-junction rescheduling solution is achieved by adjustment of the solution of each single-junction area until it does not produce further conflict in the overall control region. These agent based methods are more successful than RB methods and alternative graph based methods, but cannot be guaranteed to find solutions that acceptably close to optimal, in most cases.

A study of the agent-based modelling approach shows that the multi-junction rescheduling problem can be solved by rescheduling in individual junction areas (agents). However, it is unlikely to provide an optimal solution. As shown in Figure 6-1, each single-junction area can be considered as being controlled by an independent agent, such that junction area A is controlled by agent A; junction area B is agent B. In the agent-based modelling approach, the optimal multi-junction solution is searched for by adjustment of the solution of each agent using a method such as local search or similar. The important fact is that the solution of agent A can cause

conflict with the solution of agent B, and that at least some conflict is unavoidable in a busy railway.

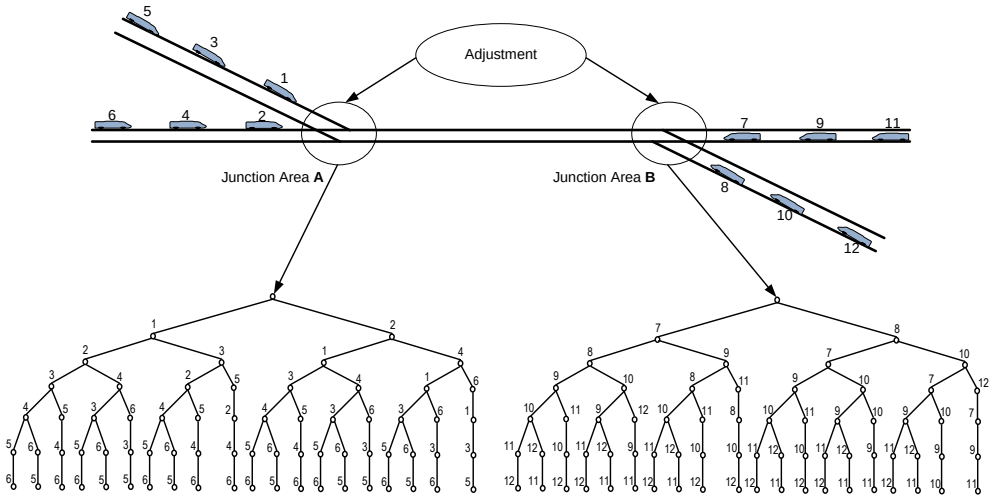


Figure 6-1: The principle of agent-based modelling approach (Fan)

The design of a novel OMJR approach is introduced in the next section.

## 6.2 Concept of Optimal Multi-junction Rescheduling Design

In the OMJR approach design in this thesis, the system is addressed as a single control problem which contains many time stages, rather than a series of communications based distributed individual control problems. In other words, the OMJR system is broken down by units of time, based on the size of the individual junctions and the characteristics of the trains under consideration, as follows.

In each time stage, trains can be rescheduled no more than once since they can only reach one junction area during any stage due to the choice of the unit time. Trains will be rescheduled stage by stage to give the best chance of optimality, while ensuring that they are conflict-free. This OMJR will finally give a series of rescheduling solutions with the lowest overall delay cost of all disturbed trains which travel within this region.

### 6.2.1 Unit Time

It is important to first know the length of the unit time in this OMJR system. This unit time is found by a simple train running time evaluation: all trains which are timetabled to cross two or more junction areas within the control region are selected, and the corresponding conflict-free running times from their starting positions to the edge of their first timetabled junction area are evaluated for maximum legal speed. For example, for Train 1 in Figure 6-2, the running time for the distance between the current position of Train 1 and the border line of junction area B is evaluated. After all running times are evaluated, the train with the shortest running time is deduced, and this time *n* is chosen to be the unit time used to divide the stages.

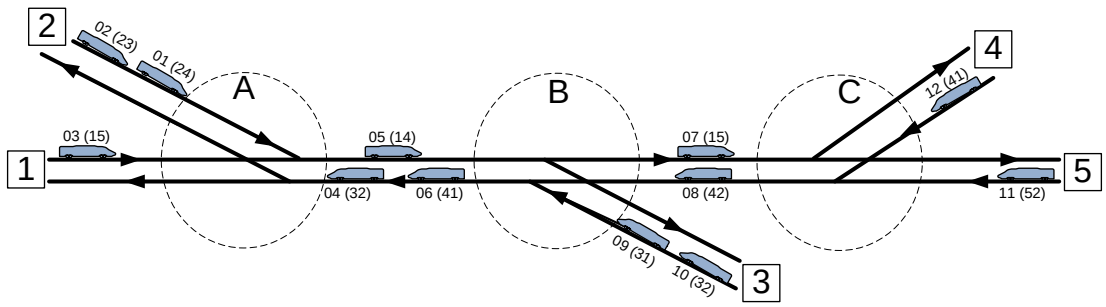


Figure 6-2: A control region example of optimal multi-junction rescheduling (Fan)

### 6.2.2 Trains in the Control System

After the unit time is known, the trains which must be considered in this optimal multi-junction rescheduling can be calculated. These trains can be found by a simple train running time evaluation, as follows. The conflict-free running times of all trains timetabled to travel within the multi-junction control area from their starting position at the designated initial time to the border line of the edge of the first junction area reached are evaluated. The longest of these running times, *m*, is selected. Then all the

trains that can arrive at the edge of any of the junction areas within time *m* are selected for consideration in this train rescheduling. This guarantees that all of the conflicts and potential conflicts can be fully solved for a timetable disturbance in the control region.

### 6.2.3 Control Stages

The number of control stages *S* is found from the longest running time *m* and the shortest time *n* from one junction area to another:

$$S = round\ up\ (\frac{m}{n}) \tag{6-1}$$

Based on the unit time, all considered trains are allocated to different control stages. In each stage, trains allocated to it at each junction area are rescheduled using a synchronous single-junction rescheduling approach.

The control stage allocation is also found from the train running time evaluation. All considered trains are evaluated *S* times to find their positions at times $1n, 2n, 3n, \cdots, Sn$. If they can reach or cross their next scheduled junction areas within a given time interval, they are allocated to the corresponding stage.

For example, in Figure 6-2, assume that Train 11 can reach junction area C within time *n*, then it is considered for rescheduling. If it cannot reach junction area C within time *n*, Train 11 is then not considered in stage 1 of this optimal multi-junction rescheduling. If Train 11 can reach or cross junction area C between time *n* and *2n* and reach or cross junction area B between time *3n* and *4n*, then Train 11 will be considered for rescheduling in stage 2 and stage 4.

### 6.2.4 Control System

Once the unit time, trains in the control area and the control stages are known, the optimal multi-junction rescheduling system can be built, as shown in Figure 6-3. All considered trains are rescheduled stage by stage until stage *S*. The running time and delay penalties of all considered trains will be evaluated according to the rescheduling results of every stage by train running time estimation. At the end of each stage, this estimation calculates the positions of trains. The position information is passed from one stage to the next until stage *S*. Additionally, the total running time and associated delay cost from each train is calculated when the train has passed all its timetabled junction areas in this control region. The running time estimation calculation method used by the simulator is explained in detail in Appendix A.

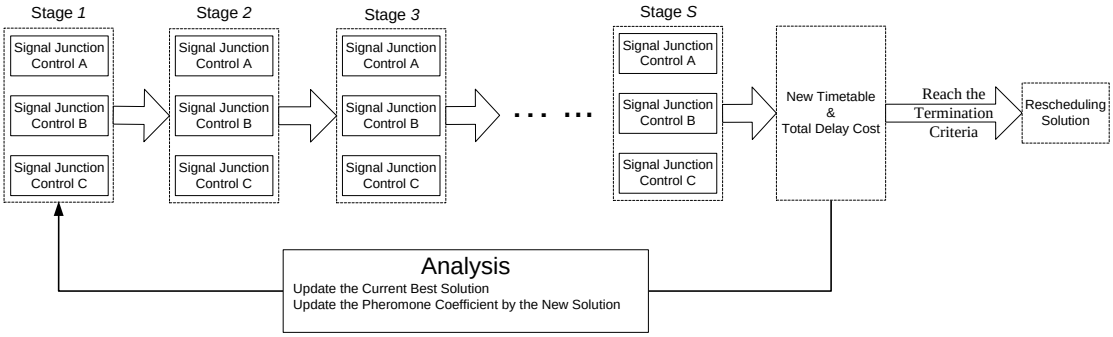


Figure 6-3: Multi-junction rescheduling system (Fan)

This OMJR system is designed using an evolutionary approach; the total delay cost at the end of every iteration (corresponding to rescheduling in each of the *S* stages) is compared, updated and analysed with the previous cost. It is derived from the basic evolutionary algorithms that were introduced in Chapter 4. These basic algorithms can be understood as feedback control systems which analyse and update the current best solution by methods such as tabu list, annealing function or pheromone coefficient.

### 6.2.5 Summary

This OMJR approach design is composed of two main steps: “prediction” and “control”. In the “prediction” step, the unit time, considered trains and the control stages are calculated by conflict-free train running time predictions. In the “control” step, a rescheduling algorithm is used to search for the optimal rescheduling solution, which contains a series of rescheduling solutions by stage.

The allocation of trains to stages for rescheduling means that once a feasible sequence has been calculated, no further conflict will occur within the given stage. This is because the unit time is calculated such that all trains may only reach a maximum of one junction area during a unit time. Therefore the stage by stage division of the multi junction rescheduling problem is capable of producing good quality, conflict-free solutions.

## 6.3 Introduction to Multi-junction Case Study

### 6.3.1 Introduction to the Multi-junction rescheduling Region

The multi-junction case study is an extension of the single-junction case study (North Stafford Junction and Stenson Junction) introduced in Chapter 2. The control area t is located on the Derby to Birmingham Line, and is shown in Figure 6-4. A detailed track diagram of this region is shown in Appendix C. There are six junctions considered in this region, namely, Branston Junction, Birmingham Curve Junction, Leicester Junction, North Stafford Junction, Stenson Junction and Melbourne Junction.

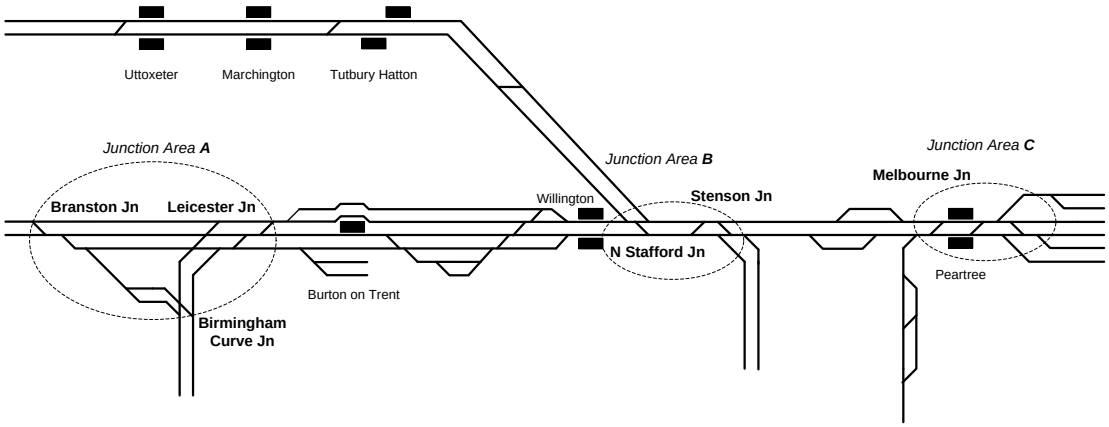


Figure 6-4: Multi-junction case study on the Derby-Birmingham line

The single junction case study rail infrastructure considered previously is a junction area consisting of two simple adjacent junctions (North Stafford Junction and Stenson Junction). In the multi junction case study, four further simple junctions in two distinct junction areas are added. Junction area A contains Branston Junction, Birmingham Curve Junction and Leicester Junction; Junction area B contains North Stafford Junction and Stenson Junction; Junction area C contains only Melbourne Junction.

### 6.3.2 Introduction to the Initial Timetable and Delay Scenarios

In order to test the OMJR approach in depth, the original timetable is set to be busier than real operations in this region. With the purpose of describing this case study clearly, all of the considered trains are shown at their timetabled initial positions on the simplified track map of the control region, in Figure 6-5. The seven directions are named A to G; the junction areas are named X, Y and Z; and trains are numbered 1 to 31. The letters (shown in brackets) are of the form (origin, destination). An origin may be either a direction or a junction area, while a destination may only be a direction. When the initial position of a train is between two junction areas, the origin is marked as the junction area it has just passed to distinguish its position.

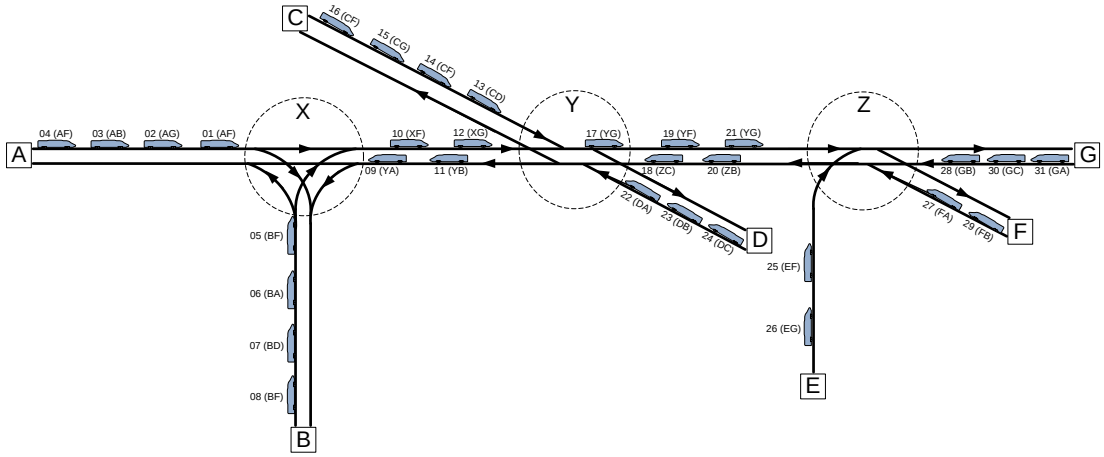


Figure 6-5: Considered trains in simplified track diagram of example OMJR control region

The initial conflict-free timetable of these 31 trains is shown in Table 6-1. The same three types of train considered previously are used again here, although the delay penalties are different. The distances between current position and next timetabled junction area, and the directions in which the destinations lie are given. The table also shows an ordered list of junction areas that trains need to pass.

| Train number | Train type | Delay penalty (£/min) | Initial distance from next junction area (km) | Destination direction | Junction areas to cross |
|---|---|---|---|---|---|
| **1** | Class 150 | 20 | 1 | F | X, Y, Z |
| **2** | Class 220 | 40 | 8 | G | X, Y, Z |
| **3** | Freight | 10 | 11 | B | X |
| **4** | Class 150 | 20 | 13 | F | X, Y, Z |
| **5** | Class 220 | 40 | 3 | F | X, Y, Z |
| **6** | Class 220 | 40 | 7 | A | X |
| **7** | Freight | 10 | 11 | D | X, Y |
| **8** | Class 150 | 20 | 16 | F | X, Y, Z |
| **9** | Class 220 | 40 | 3 | A | X |
| **10** | Class 220 | 40 | 9 | F | Y, Z |
| **11** | Freight | 10 | 10 | B | X |
| **12** | Class 150 | 20 | 3 | G | Y, Z |
| **13** | Class 150 | 20 | 1 | D | Y |
| **14** | Class 220 | 40 | 7 | F | Y, Z |
| **15** | Freight | 10 | 14 | G | Y, Z |
| **16** | Class 220 | 40 | 16 | F | Y, Z |
| **17** | Freight | 10 | 5 | G | Z |

| Train number | Train type | Delay penalty (£/min) | Initial distance from next junction area (km) | Destination direction | Junction areas to cross |
|---|---|---|---|---|---|
| **18** | Class 150 | 20 | 5 | C | Y |
| **19** | Freight | 10 | 5 | F | Z |
| **20** | Class 150 | 20 | 6 | B | Y, X |
| **21** | Class 220 | 40 | 1 | E | Z |
| **22** | Class 220 | 40 | 1 | A | Y, X |
| **23** | Freight | 10 | 3 | B | Y, X |
| **24** | Class 150 | 20 | 10 | C | Y |
| **25** | Class 150 | 20 | 8 | F | Z |
| **26** | Freight | 10 | 14 | G | Z |
| **27** | Class 150 | 20 | 3 | A | Z, Y, X |
| **28** | Class 220 | 40 | 1 | B | Z, Y, X |
| **29** | Class 220 | 40 | 6 | B | Z, Y, X |
| **30** | Freight | 10 | 11 | C | Z, Y |
| **31** | Class 150 | 20 | 14 | A | Z, Y, X |

Table 6-1: The conflict-free timetable and train delay penalties

Four scenarios are considered, as follows:

Scenario (1): Train 1 is delayed by 3 minutes, resulting in a conflict with Train 9 in junction area X - for this scenario, the first train to pass through junction area X is delayed. In this scenario, different rescheduling decisions could cause different delay scenarios and conflicts in the other junction areas.

Scenario (2): Train 1 is delayed by 7 minutes and Train 22 is delayed by 7 minutes– for this scenario, the delay happens at two junction areas; the order of Train 1 and Train 2 is reversed and Train 22 and Train 23 is reversed.

Scenario (3): Train 1 is delayed by 5 minutes and Train 27 is delayed by 5 minutes, resulting in Train 2 being delayed by 2 minutes and Train 29 is delayed by 4 minutes – for this scenario, the delays of Train 1 and Train 27 result in a knock-on delay to Train 7 and Train 29 at two junction areas.

Scenario (4): Train 5 is delayed by 4 minutes, Train 13 is delayed by 4 minutes, Train 22 is delayed by 4 minutes and Train 28 is delayed by 4 minutes – for this scenario multiple trains at all three junction areas travelling in four directions are delayed and knock-on delays occur.

## 6.4 Implementation of the Optimal Multi-junction Rescheduling Approach

The steps for the optimal multi-junction rescheduling approach are given below. An example illustrating the steps is also included in the description.

Step 1

Identify the different track distances between entrance points to all pairs of junction areas.

An example is shown in Figure 6-6.

Save the distances as $D_1, D_2, D_3, \cdots, D_w$.

The sequence of the distances, $D_i$, is unimportant.

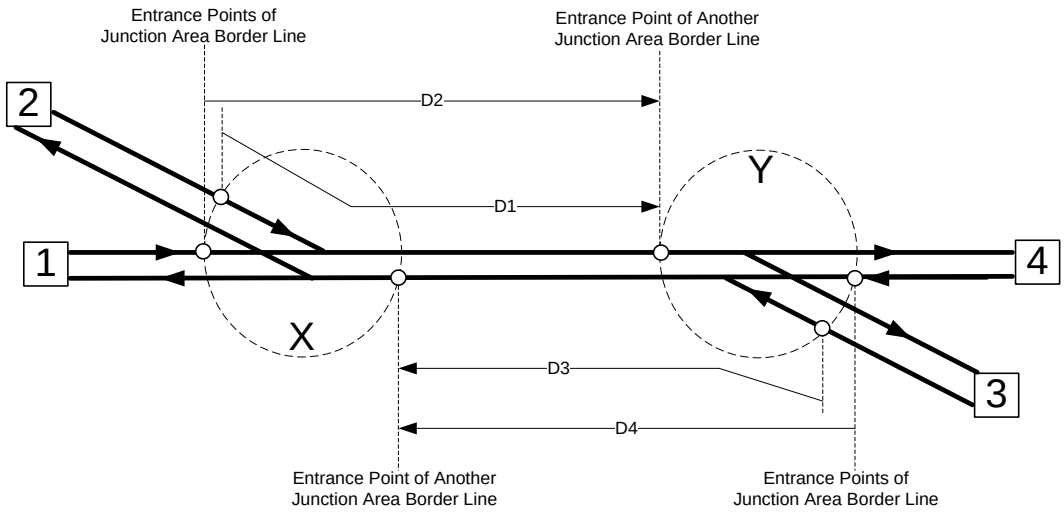


Figure 6-6: Distance between entrance points of pairs of junction areas

Deduce which $D_i$ apply to each train by observing which junction areas they need to pass and save as $J_d$.

The junction areas that need to be passed in this example are shown in Table 6-1. For example, Train 1 needs to cross junction areas X, Y and Z, so $J_d(1) = \{D_1, D_5, D_9\}$; Train 31 needs to cross junction area Z, Y and X; it is saved as $J_d(31) = \{D_{12}, D_8, D_4\}$. If a train only needs to cross one junction area in this control region no $J_d$ will be saved, e.g., for Train 6, $J_d(6) = \emptyset$.

For each $J_d$ use the simulator to evaluate the conflict-free running time over each of its corresponding distances.

The simulator must use the appropriate track and train parameters, e.g. line speed limit, train class.

Amongst all evaluated running times between pairs of junctions, select the shortest running time, and declare it to be the unit time, *n*.

Evaluate the total running time of every train for the distance between its starting position and the exit point of the last junction area that it needs to cross.

Select the longest running time and denote it *m*.

Find the number of stages *S* using Equation 6-1: $S = round\ up\ (m/n)$.

For each train, calculate the distances between their starting position and the entrance point to every junction area that the train needs to cross and list in the variable $J_c$.

For example, Train 1 needs to pass junction areas X, Y and Z. $J_c(1) = \{K_X^1, K_Y^1, K_Z^1\}$, where $K_X^1$ is the distance from the starting position of Train 1 to the entrance point of junction area X; $K_Y^1$ is the distance from the starting position of Train 1 to the entrance point of junction area Y; $K_Z^1$ is the distance from the starting position to the entrance point of junction area Z.

Using the $J_c$ of each train, evaluate the conflict-free running time, *Rt*, of each distance *K* and save as $T_j$.

For example, Train 1 needs to pass junction areas X, Y and Z. $T_j(1) = \{Rt_X^1, Rt_Y^1, Rt_Z^1\}$, where $Rt_X^1$ is the running time of Train 1 for distance $K_X^1$; $Rt_Y^1$ is the running time of Train 1 for distance $K_Y^1$ and $Rt_Z^1$ is the running time of Train 1 for distance $K_Z^1$.

It is always true that for any of the trains $0 \leq Rt_X \leq m \leq S \times n$.

For each train, identify the interval in which each *Rt* falls from *[0, n), [n, 2n), [2n, 3n),…, [(S-1)n, Sn]* and saved as $F_s = \{ I_X, I_Y, I_Z\}$; where, if $Rt_X \in (i, (i+1)n)$, $I_X = i + 1$, for all *i* between 0 and *S-1*.

Trains cannot be rescheduled more than once in a single stage.

For example, if $n = 360$ seconds, *S* = 5, and $T_j(1) = \{Rt_X^1, Rt_Y^1, Rt_Z^1\} = \{425s, 850s, 1793s\}$, then $F_s(1) = \{I_X^1, I_Y^1, I_Z^1\} = \{2, 3, \mathrm{S}\}$, which means: train 1 will be rescheduled in stage 2 at junction area X, in stage 3 at junction area Y and in stage *5 (which is the final stage, S)* at junction area Z.

List all trains corresponding to each rescheduling stage and junction area in a series of functions $C(\alpha, \beta)$, where $\alpha$ is the number of the stage, $\beta$ is the junction area.

For example, if $C(1, X) = \{1, 5, 9, 11\}$, it means that Train 1, Train 5, Train 9 and Train 11 will be rescheduled at junction area X in stage 1; if $C(3, \mathrm{Y}) = \{4, 8, 16, 26, 31\}$, then Train 4, Train 8, Train 16, Train 26 and Train 31 will be rescheduled at junction area Y in stage 3.

Generate a two dimensional matrix sequence constraint function, *CONS*.

The rows of this matrix correspond to each of the train lines within the control area and the trains are listed in the order in which they lie on the line at the beginning of the stage. Trains listed within any given row of the constraint matrix cannot overtake one another during any given stage.

This sequence constraint function will be updated after every rescheduling stage with the new train positions. An example sequence constraint function for this case study is:

$$CONS = \begin{bmatrix} 21 & 19 & 17 & 12 & 10 & 01 & 02 & 03 & 04 \\ 09 & 11 & 18 & 20 & 28 & 30 & 31 & NAN & NAN \\ 05 & 06 & 07 & 08 & NAN & NAN & NAN & NAN & NAN \\ 13 & 14 & 15 & 16 & NAN & NAN & NAN & NAN & NAN \\ 20 & 23 & 24 & NAN & NAN & NAN & NAN & NAN & NAN \\ 25 & 26 & NAN & NAN & NAN & NAN & NAN & NAN & NAN \\ 27 & 29 & NAN & NAN & NAN & NAN & NAN & NAN & NAN \end{bmatrix}$$

This corresponds to the initial sequence shown in Figure 6-5.

Create an independent ant path function $AP(\alpha,\beta)$ for every junction area at every stage. $AP(\alpha,\beta)$ is an $Lc$ by $Lc$ matrix, where $Lc$ is the length of $C(\alpha,\beta)$ and every row is a copy of $C(\alpha,\beta)$.

Since there are three junction areas in the example shown here, there are $3 \times S$ ant path functions created here. Each element in *AP* can be understood as equivalent to a decision tree node since a path between nodes from the top to the bottom provides a potential solution.

Create the corresponding pheromone coefficient functions $PC(\alpha,\beta)$ for each $AP(\alpha,\beta)$. Every pheromone coefficient function is an $Lc \times Lc$ matrix.

Set the initial value of all $PC(\alpha,\beta)$ elements to 1.

Every value of $PC(\alpha,\beta)$ represents the corresponding pheromone coefficient of the train in the same position of $AC(\alpha,\beta)$. Figure 6-7 shows an example of an ant path function and corresponding initial pheromone coefficient function.

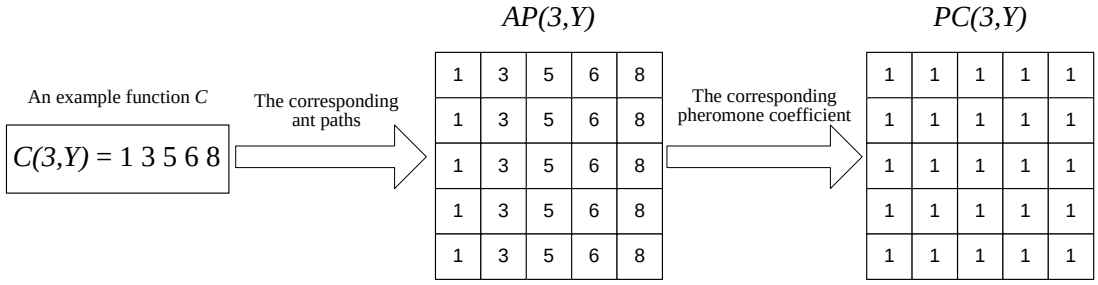


Figure 6-7: An example of function *C* and its corresponding ant path pheromone coefficient function in the initial setting (Fan)

As introduced previously, this optimal multi-junction design is composed of two main steps: 'prediction' and 'control'; Step 1 is the completed 'prediction' step. All the preparation work before train rescheduling is done in this step. The important outputs for later use in this step are: the number of stages, *S*, which are a unit time *n* in duration; the stage to which each train is allocated and at which junction, which is contained in the three-dimensional matrix $C(\alpha,\beta)$, the sequence constraint function *CONS* and the pheromone coefficient function $PC(\alpha,\beta)$. Only the sequence constraint function *CONS* and the pheromone coefficient function *PC* are variables, which will be updated after every stage and for each iteration as

the train positions change. All four other outputs *S*, *n*, *C* and *AP* are constant, for later use once the final rescheduling solution is found.

Step 2

Set the number of iterations, $g = 1$.

Select the FCFS sequence and another nine randomly selected feasible sequences for the trains in each junction area of stage 1 (the number of ants is selected refer to Section 4.5.2).

For the first junction area in stage 1 the sequences are $O_1^1(1,X), O_2^1(1,X), O_3^1(1,X), \cdots, O_{10}^1(1,X)$, where, $O_1^1(1,X)$ is the FCFS sequence of $C(1,X)$; and the other nine sequences are random feasible sequences of $C(1,X)$. Similarly, the ten sequences for the other junctions areas for stage 1 are $O_1^1(1,Y), O_2^1(1,Y), O_3^1(1,Y), \cdots, O_{10}^1(1,Y)$ and $O_1^1(1,Z), O_2^1(1,Z), O_3^1(1,Z), \cdots, O_{10}^1(1,Z)$.

Construct the first rescheduling solution $R_1^1$ of stage 1 as $\{O_1^1(1,X), O_1^1(1,Y), O_1^1(1,Z)\}$, which is also known as the FCFS solution.

Construct the remaining nine rescheduling solutions of the first stage, $R_2^1 = \{O_2^1(1,X), O_2^1(1,Y), O_2^1(1,Z)\}$, …, $R_{10}^1 = \{O_{10}^1(1,X), O_{10}^1(1,Y), O_{10}^1(1,Z)\}$.

Update the sequence of the rescheduling constraint function *CONS* to reflect the new positions of the trains based on the rescheduling solutions $R_1^1$ to $R_{10}^1$..

Since different rescheduling solutions give different sequence constraints, the sequence constraints are saved as ten functions: $CONS_1$, $CONS_2$, $\cdots\cdots, CONS_{10}$. Figure 6-8 shows an example of the junction area and corresponding constraint function for two different example stages.

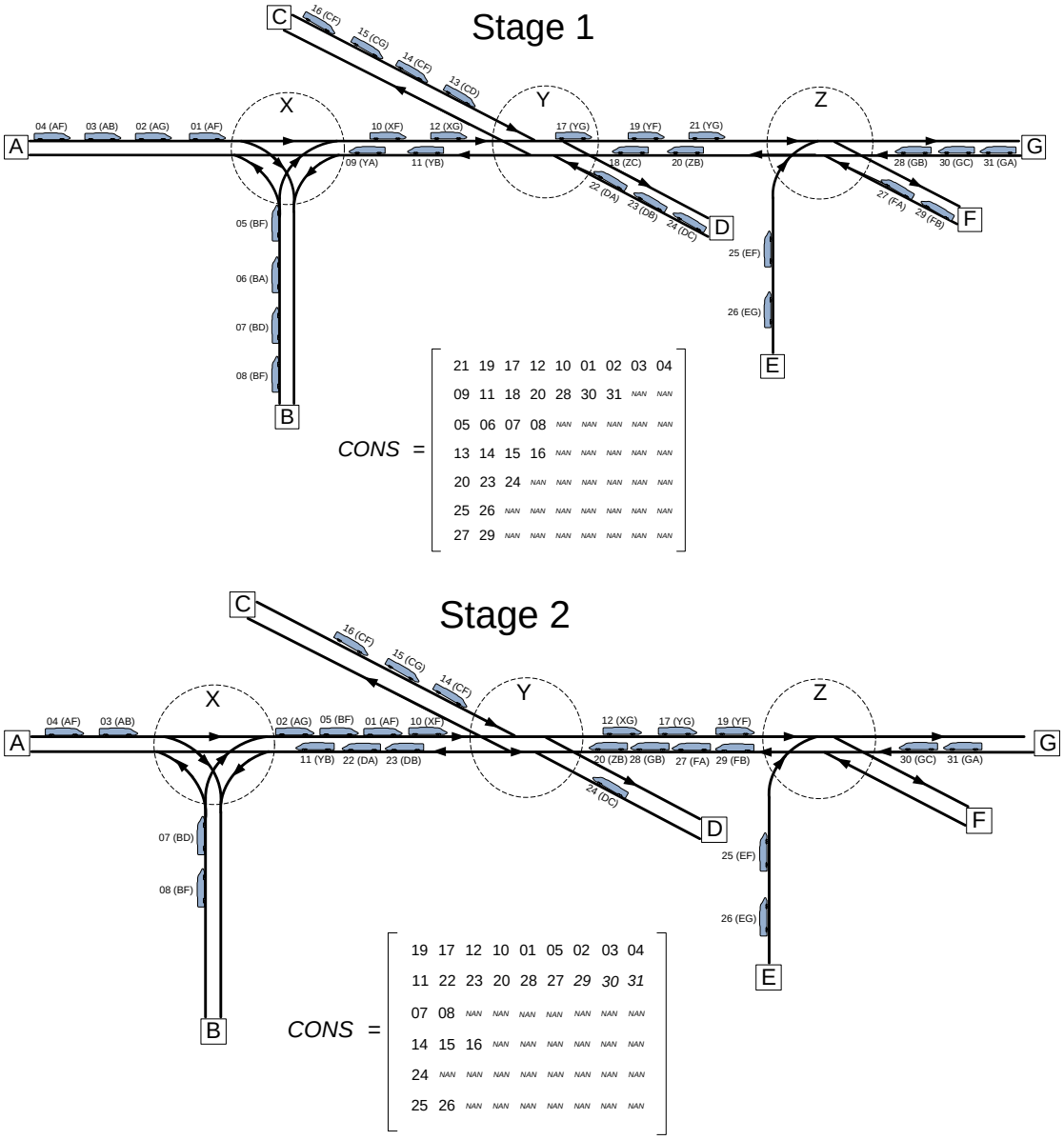


Figure 6-8: An example of sequence constraints in two different stages (Fan)

Increase the stage by 1. For the trains in the next stage, $C(2,X), C(2,Y)$ and $C(2,Z)$, select the FCFS sequence $R_1^2$ that includes FCFS sequences for each junction area: $O_1^2(1,X), O_1^2(1,Y), O_1^2(1,Z)$. Randomly select the other nine sequences, $R_2^2$ to $R_{10}^2$, so that they satisfy respectively the sequence constraints $CONS_2$ to $CONS_{10}$. Update $CONS_1$ to $CONS_{10}$ with the new train positions in $R_1^2$ to $R_{10}^2$.

Repeat the last sub-step until stage = *S*.

In every sub-step, ten rescheduling sub-solutions have been achieved. In total, there are $10 \times S$ sub-solutions. The *i*th full solution consists of the *i*th sub-solution of every stage.

For each of the ten full solutions, estimate the running time of each train from its initial position to its destination using the simulator.

Evaluate the delay cost of all full solutions and save the solution with the lowest delay cost as: $\hat{\theta}$.

This $\hat{\theta}$ contains a series of optimal rescheduling sub-solutions for each stage and every junction area ($R_{\hat{\theta}}^{1}$ to $R_{\hat{\theta}}^{S}$ ).

Finally, update the pheromone coefficient functions $PC(\alpha, \beta)$ so that any node which has been visited is reduced by multiplying with 0.5.

Set the pheromone coefficients for the nodes on the lowest cost path, $\hat{\theta}$, to 1.

An example is shown in Figure 6-9. Firstly, list the FCFS and randomly selected sequences. Then, the elements that have appeared in each row of the listed sequences are selected. Compare these elements with the corresponding ant path function *AP*. The corresponding elements of pheromone coefficient function *PC* of the listed are divided by 2. Finally, the corresponding pheromone coefficients of the best solution are reset to 1.

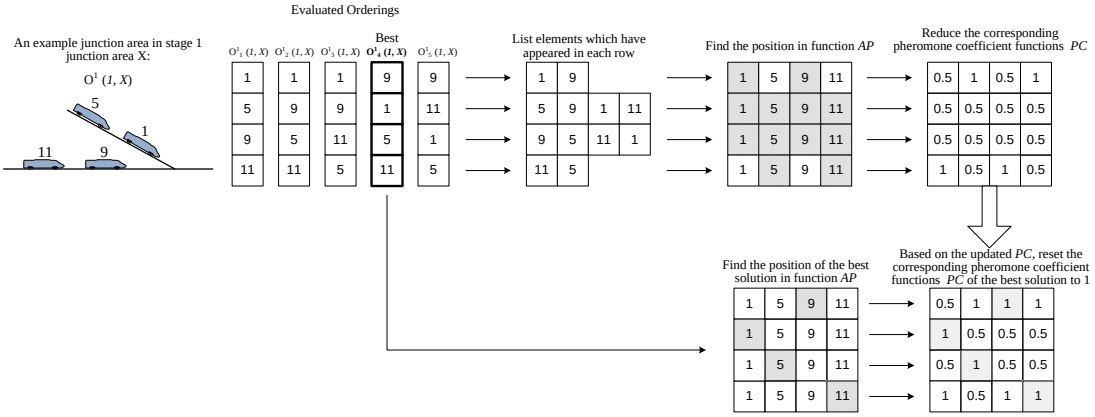


Figure 6-9: The technique of updating pheromone coefficient functions (Fan)

As introduced previously in the concept of this multi-junction rescheduling (section 6.2), the aim of this step is to initialise the rescheduling approach. Nine rescheduling solutions together with FCFS solution are evaluated and compared. The pheromone coefficient

functions are updated for the first time here. In later steps, ACO will be used and ants will select their path based on the updated pheromone coefficient functions.

Step 3

Increase g by 1

Reset the sequence constraint functions *CONS* to the initial train positions.

Reset the rescheduling stage to 1.

Select ten new different sequences $R_1^1$ *to* $R_{10}^1$ from the current optimal sequence $R_{\hat{\theta}}^1$ by using the hybrid algorithm, which was described in detail in Section 5.2.

Update $CONS_1$ to $CONS_{10}$ according to the positions of trains in $R_1^1$ to $R_{10}^1$.

In this multi-junction rescheduling approach, the hybrid algorithm cannot be used in exactly the same way as in Chapter 5. This is because the sequence of constraints *CONS* is a variable when multiple junction areas are involved. Therefore, the hybrid algorithm is modified here. Since the infeasible-feasible converter is used after local search and it makes reference to the updated sequence of constraints in every stage, the different constraint sequence functions do not affect the local search part of the hybrid algorithm. Therefore, only the ACO part needs to be modified.

In the modified ACO, every train is generated in each row of *AP* (similar to a single level of a decision tree; see Section 2.3.2) as shown in Figure 6-10. Every train in each level has its corresponding pheromone coefficient function $PC(\alpha, \beta)$. In the search processing, only the next possible trains are generated, based on the updated sequence constraints. After this, the possibility of choosing the next train is based on the pheromone coefficient, in the same way as introduced for DTBACO in Chapter 5. Finally, the pheromone coefficients are updated based on the comparison of the new solution and the solution from the previous iteration.

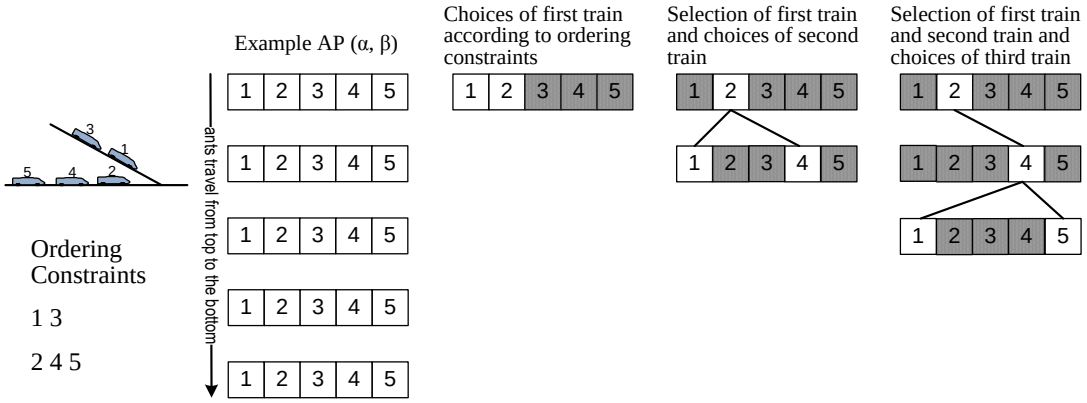


Figure 6-10: The technique of ACO in multi-junction control (Fan)

Select ten sequences, $R_1^2$ to $R_{10}^2$, for the next stage from $R_{\hat{\theta}}^2$ using the hybrid algorithm.

Update $CONS_1$ to $CONS_{10}$ according to the positions of trains in $R_1^2$ to $R_{10}^2$.

Repeat the last sub-step until stage = *S*.

After this sub-step has been completed, ten new rescheduling solutions have been generated.

Estimate the running time of each train to its destination and evaluate the delay costs of all solutions.

Select the solution $\theta$ with the lowest delay cost. If $J(\theta) \leq J(\hat{\theta})$, then $\hat{\theta} := \theta$.

Reset the sequence of constraints *CONS* to the initial train positions.

Update the pheromone coefficient functions $PC(\alpha, \beta)$ by dividing the pheromone coefficients of visited nodes by 2 and resetting the pheromone coefficients of nodes on the $\hat{\theta}$ path to 1.

Step 3 executes an iteration of the modified hybrid algorithm, in which one iteration contains three sub-steps: 1. finding ten new rescheduling solutions based on the value of the pheromone coefficients; 2. using local search and solution adjustment to achieve an improvement; 3. updating the pheromone coefficient functions of the evaluated solutions for the next iteration. This step of the algorithm will be repeated until the termination criterion is met, as described in Step 4 below. In this modified hybrid algorithm, with the intention of taking full advantage of rapid convergence, the pheromone coefficient decrease rate is set to 0.5.

Step 4

Go back to step 3, unless the best solution has remained the same for 10 consecutive iterations, in which case, terminate the algorithm.

## 6.5 Results and Discussion of the Optimal Multi-junction Rescheduling Approach

In this section, the OMJR approach is applied to the four delay scenarios described in Section 6.3.2. For each of the delay scenarios, the total number of stages, *S*, is 4. Table 6-2, Table 6-3, Table 6-4 and Table 6-5 show the results of these four scenarios, including the rescheduling results of each stage and the delay cost for each delayed train.

| | **Junction area** | **Order in which the trains pass the junction area** |
|---|---|---|
| **Stage 1** | X<br>Y<br>Z | 09, 01, 05, 02, 06<br>13, 22, 12, 23, 18, 20<br>21, 28, 27, 29, 19 |
| **Stage 2** | X<br>Y<br>Z | 03, 11, 07, 04, 22<br>10, 14, 28, 01, 24<br>17, 25, 12, 30, 31 |
| **Stage 3** | X<br>Y<br>Z | 04, 23, 08, 20, 28<br>15, 27, 05, 29, 02, 16<br>26, 10, 14, 01 |
| **Stage 4** | X<br>Y<br>Z | 27, 29<br>07, 30, 04, 31, 08<br>15, 05, 02, 16 |
| **Delayed Trains (Delay Cost in £)** | | 01(40), 02(80), 05(40), 07(30), 14(40), 27(40) |
| **Total Delay Cost** | | £260 |

Table 6-2: The results of scenario 1 of the OMJR approach

| | Junction area | Order in which the trains pass the junction area |
|---|---|---|
| **Stage 1** | X<br>Y<br>Z | 09, 05, 02, 06, 01<br>13, 12, 23, 18, 22, 20<br>21, 28, 27, 29, 19 |
| **Stage 2** | X<br>Y<br>Z | 03, 11, 07, 04, 22<br>10, 14, 28, 01, 24<br>17, 25, 12, 30, 31 |
| **Stage 3** | X<br>Y<br>Z | 03, 11, 07, 04, 23<br>10, 14, 28, 02, 24,<br>17, 25, 12, 30, 31 |
| **Stage 4** | X<br>Y<br>Z | 27, 29<br>07, 30, 04, 31, 08<br>15, 05, 02, 01, 16 |
| **Delayed Trains (Delay Cost in £)** | | 1(120), 3(20), 7(20), 22(200), 08(20), 20(30) |
| **Total Delay Cost** | | £410 |

Table 6-3: The results of scenario 2 of the OMJR approach

| | Junction area | Order in which the trains pass the junction area |
|---|---|---|
| **Stage 1** | X<br>Y<br>Z | 09, 05, 01, 06,<br>13, 22, 12, 23, 18, 20<br>21, 28, 19, 27 |
| **Stage 2** | X<br>Y<br>Z | 29, 17, 25, 12, 30, 31<br>10, 14, 28, 24, 27<br>02, 03, 11, 07, 04, 22 |
| **Stage 3** | X<br>Y<br>Z | 4, 23, 8, 20, 28<br>15, 05, 01, 16, 29, 02<br>26, 10, 14, 01 |
| **Stage 4** | X<br>Y<br>Z | 27, 29<br>07, 30, 04, 31, 08<br>15, 05, 01, 02, 16 |
| **Delayed Trains (Delay Cost in £)** | | 1(100), 2(80), 03(20), 06(80), 27(100), 29(80), 17(20), 19(20), 25(20) |
| **Total Delay Cost** | | £520 |

Table 6-4: The results of scenario 3 of the OMJR approach

| | Junction area | Order in which the trains pass the junction area |
|---|---|---|
| **Stage 1** | X<br>Y<br>Z | 01, 09, 02, 05, 06,<br>12, 22, 23, 18, 20, 13<br>21, 27, 28, 29, 19 |
| **Stage 2** | X<br>Y<br>Z | 03, 11, 07, 04<br>10, 14, 01, 24,<br>17, 25, 12, 30, 31 |
| **Stage 3** | X<br>Y<br>Z | 04, 22, 08, 23, 20,<br>26, 10, 14, 01<br>15, 27, 28, 29, 02, 16, 05 |
| **Stage 4** | X<br>Y<br>Z | 15, 02, 16, 05<br>07, 30, 04, 31, 08<br>27, 28, 29 |
| **Delayed Trains (Delay Cost in £)** | | 05(160), 06(80), 13(120), 22(160), 23(40), 28(200), 29(80), 18(40), 20,(40), 19(30) |
| **Total Delay Cost** | | £930 |

Table 6-5: The results of scenario 4 of the OMJR approach

In order to compare the differences between this optimal multi-junction rescheduling approach and FCFS, the delay costs of the FCFS solutions for these four scenarios are shown in Table 6-6. The FCFS solutions are achieved without dividing the problem into stages.

| **Scenarios**<br>**Method** | **Scenario 1** | **Scenario 2** | **Scenario 3** | **Scenario 4** |
|---|---|---|---|---|
| **FCFS** | £420 | £690 | £780 | £1410 |
| **OMJR** | £260 | £410 | £520 | £930 |

Table 6-6: The delay costs of FCFS and optimal multi-junction rescheduling approach for the four scenarios introduced in Section 6.3.2

It can be seen from Table 6-6 that the OMJR approach is always significantly better than FCFS in these four scenarios in terms of delay cost.

In the OMJR problem the sequence constraints differ from stage to stage. The graphical algorithms such as conventional decision tree and SST cannot be used when the constraints are different. Furthermore, evolutionary algorithms such as TS and GA

struggle to guide the search direction to the optimum when the constraints are different. In TS, the better swapping pairs cannot be distinguished for different constraints and they are may be saved in tabu list; and even in the worst case scenario, new sequence constraints could lock every possible swap. In GA, changes to the constraint function are high likely to produce illegal and infeasible solutions. GAs will lose some of their advantage when the illegal-legal and infeasible-feasible converter must forced to be applied. Therefore, the modified hybrid algorithm is used here since it can readily give new solutions with different constraints. This modified hybrid algorithm also can be used to full advantage in each single junction area within each iteration since it will not produce illegal and infeasible solutions since the appropriate decision tree is re-generated in each iteration.

## 6.6 Control Region Discussion

The optimal multi-junction rescheduling approach has been proved to be better than FCFS in four delay scenarios in the previous section. In this section the results of extensive further testing are analysed; 100 random delay scenarios were generated and tested using multiple junction rescheduling, single junction rescheduling and FCFS approaches and the results are discussed.

These 100 delay scenarios were generated within 10 different delay intervals. The control region, the number and type of trains and their initial timetables are the same as in the case study which was introduced in Section 6.3.2.

In these 100 scenarios, a negative number of delay minutes means the early arrival of a train. The resulting positions of trains and the headways are checked for each of the delay scenarios; if two or more trains overlap or have an illegal headway, the

following trains are adjusted to the legal headway position with the corresponding number of delay minutes.

These 100 delay scenarios are rescheduled by FCFS, the optimal single-junction rescheduling approach (applied separately to any junction areas in which delays are introduced) and the optimal multi-junction rescheduling. The resulting delay costs and the obtained delay cost differences are shown in Table 6-7 to Table 6-16 as follows.

$$D1 = cost\ obtained\ using\ FCFS\ - cost\ from\ OSJR\ approach$$

$$D2 = cost\ obtained\ using\ FCFS - cost\ from\ OMJR\ approach$$

$$D3 = cost\ obtained\ using\ OSJR\ approach - cost\ from\ OMJR\ approach$$

After this, the improvement over FCFS of the OSJR approach and the OMJR approach are given, where:

$$D1' = \frac{cost\ obtained\ using\ FCFS\ - cost\ from\ OSJR\ approach}{cost\ obtained\ using\ FCFS}$$

$$D2' = \frac{cost\ obtained\ using\ FCFS - cost\ from\ OMJR\ approach}{cost\ obtained\ using\ FCFS}$$

$$D3' = \frac{cost\ obtained\ using\ OSJR\ approach - cost\ from\ OMJR\ approach}{cost\ obtained\ using\ OSJR\ approach}$$

1. Randomly select one junction area out of the three; randomly delay every train which must pass this junction area by a number of minutes between -2 and 2. Repeat this process another nine times to make 10 delay scenarios (Scenarios 1 to 10).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **1** | 280 | 270 | 260 | 10 | 20 | 10 |
| **2** | 340 | 190 | 150 | 150 | 190 | 40 |
| **3** | 410 | 410 | 410 | 0 | 0 | 0 |
| **4** | 320 | 280 | 200 | 40 | 120 | 80 |
| **5** | 370 | 340 | 210 | 30 | 160 | 130 |
| **6** | 120 | 70 | 0 | 50 | 120 | 70 |
| **7** | 310 | 280 | 260 | 30 | 50 | 20 |
| **8** | 350 | 190 | 120 | 160 | 230 | 70 |
| **9** | 320 | 320 | 300 | 0 | 20 | 20 |
| **10** | 110 | 60 | 50 | 50 | 60 | 10 |
| **Average** | ***293*** | ***241*** | ***196*** | ***52*** | ***97*** | ***45*** |
| **Occurrence Frequency of "0"** | | | | ***2*** | ***1*** | ***1*** |
| **Performance Improvements** | | | | ***D1' 18%*** | ***D2' 33%*** | ***D3' 19%*** |

Table 6-7: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 1-10, in £

2. Randomly select two junction areas out of the three; randomly delay every train which needs to pass at least one of these two junction areas by between −2 and 2 minutes. Repeat for the process another nine times to make 10 delay scenarios (Scenarios 11 to 20).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **11** | 500 | 410 | 370 | 90 | 130 | 40 |
| **12** | 870 | 830 | 750 | 40 | 120 | 80 |
| **13** | 720 | 610 | 550 | 110 | 170 | 60 |
| **14** | 440 | 420 | 420 | 20 | 20 | 0 |
| **15** | 530 | 520 | 490 | 10 | 40 | 30 |
| **16** | 410 | 340 | 170 | 70 | 240 | 170 |
| **17** | 390 | 320 | 320 | 70 | 70 | 0 |
| **18** | 300 | 220 | 180 | 80 | 120 | 40 |
| **19** | 450 | 450 | 390 | 0 | 60 | 60 |
| **20** | 530 | 410 | 360 | 120 | 170 | 50 |
| **Average** | ***514*** | ***453*** | ***400*** | ***61*** | ***114*** | ***53*** |
| **Occurrence Frequency of "0"** | | | | ***0*** | ***1*** | ***1*** |
| **Performance Improvements** | | | | ***D1' 12%*** | ***D2' 22%*** | ***D3' 12%*** |

Table 6-8: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 11-20, in £

3. Randomly delay every train by between -2 and 2 minutes. Repeat another nine times to make 10 delay scenarios (Scenarios 21 to 30).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **21** | 600 | 590 | 540 | 10 | 60 | 50 |
| **22** | 530 | 430 | 390 | 100 | 140 | 40 |
| **23** | 1040 | 810 | 710 | 230 | 330 | 100 |
| **24** | 960 | 920 | 920 | 40 | 40 | 0 |
| **25** | 720 | 690 | 650 | 30 | 70 | 40 |
| **26** | 730 | 730 | 730 | 0 | 0 | 0 |
| **27** | 1130 | 1030 | 940 | 100 | 190 | 90 |
| **28** | 330 | 330 | 290 | 0 | 40 | 40 |
| **29** | 650 | 540 | 410 | 110 | 240 | 130 |
| **30** | 630 | 620 | 360 | 10 | 270 | 260 |
| **Average** | ***732*** | ***669*** | ***594*** | ***63*** | ***138*** | ***75*** |
| **Occurrence Frequency of "0"** | | | | ***2*** | ***1*** | ***2*** |
| **Performance Improvements** | | | | ***D1'*** ***9%*** | ***D2'*** ***19%*** | ***D3'*** ***11%*** |

Table 6-9: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 21-30, in £

4. Randomly select one junction area out of the three; randomly delay every train which needs to pass this junction area by between 2 and 6 minutes. Repeat another nine times to make 10 delay scenarios (Scenarios 31 to 40).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **31** | 720 | 710 | 650 | 10 | 70 | 60 |
| **32** | 1080 | 960 | 850 | 120 | 230 | 110 |
| **33** | 870 | 860 | 530 | 10 | 340 | 330 |
| **34** | 810 | 740 | 600 | 70 | 210 | 140 |
| **35** | 1290 | 1270 | 1120 | 20 | 170 | 150 |
| **36** | 1130 | 1020 | 870 | 110 | 260 | 150 |
| **37** | 990 | 850 | 660 | 140 | 330 | 190 |
| **38** | 1390 | 1380 | 1240 | 10 | 150 | 140 |
| **39** | 970 | 640 | 640 | 330 | 330 | 0 |
| **40** | 1580 | 1340 | 1140 | 240 | 440 | 200 |
| **Average** | ***1083*** | ***977*** | ***830*** | ***106*** | ***253*** | ***147*** |
| **Occurrence Frequency of "0"** | | | | ***0*** | ***0*** | ***1*** |
| **Performance Improvements** | | | | ***D1'*** ***10%*** | ***D2'*** ***23%*** | ***D3'*** ***15%*** |

Table 6-10: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 31-40, in £

5. Randomly select two junction areas out of the three; randomly delay every train which needs to pass this junction area by between 2 and 6 minutes. Repeat another nine times to make 10 delay scenarios (Scenarios 41 to 50).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **41** | 630 | 620 | 620 | 10 | 10 | 0 |
| **42** | 1430 | 1400 | 1110 | 30 | 320 | 290 |
| **43** | 960 | 510 | 470 | 450 | 490 | 40 |
| **44** | 1340 | 1240 | 1080 | 100 | 260 | 160 |
| **45** | 1640 | 1460 | 1270 | 180 | 370 | 190 |
| **46** | 930 | 900 | 870 | 30 | 60 | 30 |
| **47** | 1760 | 1710 | 1560 | 50 | 200 | 150 |
| **48** | 810 | 800 | 680 | 10 | 130 | 120 |
| **49** | 1400 | 1360 | 1330 | 40 | 70 | 30 |
| **50** | 1500 | 1360 | 1270 | 140 | 230 | 90 |
| **Average** | ***1240*** | ***1136*** | ***1026*** | ***104*** | ***214*** | ***110*** |
| **Occurrence Frequency of "0"** | | | | ***0*** | ***0*** | ***1*** |
| **Performance Improvements** | | | | ***D1' 8%*** | ***D2' 17%*** | ***D3' 10%*** |

Table 6-11: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 41-50, in £

6. Randomly delay every train by between 2 and 6 minutes. Repeat another nine times to make 10 delay scenarios (Scenarios 51 to 60).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **51** | 3670 | 3670 | 3130 | 0 | 540 | 540 |
| **52** | 1690 | 1680 | 1650 | 10 | 40 | 30 |
| **53** | 2630 | 2630 | 2190 | 0 | 440 | 440 |
| **54** | 1300 | 1180 | 1180 | 120 | 120 | 0 |
| **55** | 1720 | 1740 | 1690 | -20 | 30 | 50 |
| **56** | 2660 | 2660 | 2660 | 0 | 0 | 0 |
| **57** | 2580 | 2570 | 2560 | 10 | 20 | 10 |
| **58** | 1510 | 1280 | 960 | 230 | 550 | 320 |
| **59** | 900 | 900 | 830 | 0 | 70 | 70 |
| **60** | 1350 | 1330 | 1210 | 20 | 140 | 120 |
| **Average** | ***2001*** | ***1964*** | ***1806*** | ***37*** | ***195*** | ***158*** |
| **Occurrence Frequency of "0"** | | | | ***4*** | ***1*** | ***2*** |
| **Performance Improvements** | | | | ***D1' 2%*** | ***D2' 10%*** | ***D3' 8%*** |

Table 6-12: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 51-60, in £

7. Randomly select one junction area out of the three; randomly delay every train which needs to pass this junction area by between 6 and 10 minutes. Repeat another nine times to make 10 delay scenarios (Scenarios 61 to 70).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **61** | 1490 | 1390 | 1350 | 100 | 140 | 40 |
| **62** | 1700 | 1700 | 1700 | 0 | 0 | 0 |
| **63** | 1580 | 1470 | 1160 | 110 | 420 | 310 |
| **64** | 1670 | 1580 | 1500 | 90 | 170 | 80 |
| **65** | 1280 | 1280 | 1020 | 0 | 260 | 260 |
| **66** | 1920 | 1960 | 1710 | -40 | 210 | 250 |
| **67** | 1230 | 920 | 680 | 310 | 550 | 240 |
| **68** | 860 | 830 | 720 | 30 | 140 | 110 |
| **69** | 1450 | 1240 | 1010 | 210 | 440 | 230 |
| **70** | 1810 | 1670 | 1640 | 140 | 170 | 30 |
| **Average** | 1499 | 1404 | 1249 | 95 | 250 | 155 |
| **Occurrence Frequency of "0"** | | | | ***2*** | ***1*** | ***1*** |
| **Performance Improvements** | | | | ***D1'*** ***6%*** | ***D2'*** ***17%*** | ***D3'*** ***11%*** |

Table 6-13: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 51-60, in £

8. Randomly select two junction areas out of the three; randomly delay every train which needs to pass this junction area by between 6 and 10 minutes. Repeat another nine times to make 10 delay scenarios (Scenarios 71 to 80).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **71** | 1880 | 1810 | 1590 | 70 | 290 | 220 |
| **72** | 2740 | 2730 | 2700 | 10 | 40 | 30 |
| **73** | 1390 | 1250 | 950 | 140 | 440 | 300 |
| **74** | 950 | 940 | 870 | 10 | 80 | 70 |
| **75** | 1460 | 1370 | 1290 | 90 | 170 | 80 |
| **76** | 1940 | 1800 | 1730 | 140 | 210 | 70 |
| **77** | 3120 | 3040 | 2920 | 80 | 200 | 120 |
| **78** | 2340 | 2380 | 2200 | -40 | 140 | 180 |
| **79** | 1870 | 1750 | 1720 | 120 | 150 | 30 |
| **80** | 2930 | 2870 | 2840 | 60 | 90 | 30 |
| **Average** | ***2062*** | ***1994*** | ***1881*** | ***68*** | ***181*** | ***113*** |
| **Occurrence Frequency of "0"** | | | | **0** | **0** | **0** |
| **Performance Improvements** | | | | ***D1'*** ***3%*** | ***D2'*** ***9%*** | ***D3'*** ***6%*** |

Table 6-14: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 71-80, in £

9. Randomly delay every train by between 6 and 10 minutes. Repeat another nine times to make 10 delay scenarios (Scenarios 81 to 90).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **81** | 2170 | 2210 | 1980 | -40 | 190 | 230 |
| **82** | 1250 | 1240 | 1240 | 10 | 10 | 0 |
| **83** | 3710 | 3700 | 3660 | 10 | 50 | 40 |
| **84** | 1240 | 1190 | 1070 | 50 | 170 | 120 |
| **85** | 1970 | 1970 | 1970 | 0 | 0 | 0 |
| **86** | 4870 | 4870 | 4860 | 0 | 10 | 10 |
| **87** | 2250 | 2330 | 2120 | -80 | 130 | 210 |
| **88** | 1210 | 1120 | 1110 | 90 | 100 | 10 |
| **89** | 3010 | 2920 | 2860 | 90 | 150 | 60 |
| **90** | 2890 | 2910 | 2890 | -20 | 0 | 20 |
| **Average** | ***2457*** | ***2446*** | ***2376*** | ***11*** | ***81*** | ***70*** |
| **Occurrence Frequency of "0"** | | | | **2** | **2** | **2** |
| **Performance Improvements** | | | | ***D1'*** ***0.4%*** | ***D2'*** ***3%*** | ***D3'*** ***3%*** |

Table 6-15: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 81-90, in £

10. Randomly delay every train by between -2 and 10 minutes. Repeat another nine times to make 10 delay scenarios (Scenarios 91 to 100).

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **91** | 3030 | 2900 | 2720 | 130 | 310 | 180 |
| **92** | 2180 | 2110 | 1650 | 70 | 530 | 460 |
| **93** | 1210 | 1140 | 810 | 70 | 400 | 330 |
| **94** | 2450 | 2440 | 2380 | 10 | 70 | 60 |
| **95** | 2910 | 2980 | 2790 | -70 | 120 | 190 |
| **96** | 1120 | 1120 | 1120 | 0 | 0 | 0 |
| **97** | 1180 | 1170 | 1160 | 10 | 20 | 10 |
| **98** | 1220 | 1140 | 990 | 80 | 230 | 150 |
| **99** | 2780 | 2690 | 2670 | 90 | 110 | 20 |
| **100** | 1670 | 1640 | 1590 | 30 | 80 | 50 |
| **Average** | ***1975*** | ***1933*** | ***1788*** | ***42*** | ***187*** | ***145*** |
| **Occurrence Frequency of "0"** | | | | ***1*** | ***1*** | ***1*** |
| **Performance Improvements** | | | | ***D1'*** ***2%*** | ***D2'*** ***9%*** | ***D3'*** ***8%*** |

Table 6-16: The costs of FCFS, OSJR and OMJR approaches in delay scenarios 91-100, in £

After this, the overall results of these 100 delay scenarios are shown in Table 6-17.

| Scenario | FCFS | OSJR | OMJR | D1 | D2 | D3 |
|---|---|---|---|---|---|---|
| **Total Average** | **1385.6** | **1321.7** | **1214.6** | **69.2** | **165.7** | **107.1** |
| **Total Occurrence Frequency of "0"** | | | | **13** | **8** | **12** |
| **Total Occurrence Frequency of Negative Value** | | | | **6** | **0** | **0** |
| **Total Performance Improvements** | | | | ***D1' 5%*** | ***D2' 12%*** | ***D3' 8%*** |

Table 6-17: The overall results of FCFS, OSJR and OMJR approaches in 100 delay scenarios, in £

The average computation time of different rescheduling approaches are shown in Table 6-18.

| Rescheduling Approach | FCFS | OSJR | OMJR |
|---|---|---|---|
| **Average Computation Time** | 0.005 | 59 | 107 |

Table 6-18: The average computation time of different rescheduling approaches in 100 delay scenarios, in seconds

Based on the average rescheduling performance improvements of these 100 delay scenarios, both the OSJR approach and the OMJR approach provide a better solution than FCFS, by 5% (*D1*') and 12% (*D2*'), respectively. However, since a negative value of *D1* is recorded 6 times, the OSJR approach has a possibility of 6% of producing a higher delay cost solution than FCFS in these scenarios. The total occurrence frequency of a negative value of *D2* and *D3* is 0, which means that OMJR is always at least the same or better than both the FCFS and OSJR approaches. In these scenarios the OMJR approach produces a better solution than FCFS in 92% of cases and the same solution 8% of the time.

The delay cost obtained from OSJR has the potential to be higher than the FCFS solution. This is the case in 6 out of 100 of these delay scenarios. This proves the

early hypothesis of this chapter that an optimal rescheduling decision of the trains at one junction area may not be the optimal decision for a multi junction region.

From the point of view of the OSJR and OMJR approaches, the OMJR approach gives a better solution than OSJR in 88% of cases, and in the other 12% they are the same. On average for these 100 delay scenarios, the OMJR approach improves the delay cost by £107 compared to the OSJR approach.

The average computation time for each rescheduling approach is shown in Table 6-18. Because the number of trains at each junction area is less than the 12 train case studies in previous chapters, the computation time of OSJR is lower than the average computation time of the single junction rescheduling case studies. In Table 6-18 the computation time shown for OSJR is for junction area Y, which is the longest of the three single junction rescheduling areas (where the average OSJR computation time for junction area X is 41 seconds; the average OSJR computation time for junction area Y is 59 seconds and the average OSJR computation time for junction area Z is 47 seconds). Although the computation time of OMJR is the longest of these three rescheduling approaches, OMJR solves the rescheduling problem completely for a given control area and gives a conflict-free solution. OSJR can only solve the problem at single junction areas and is likely to produce further conflicts at other junction areas. In addition, in order to make the computation time of OMJR acceptable for use in a real railway control system, it could be reduced by using better computer, a lower level programming language (e.g., C++) and parallel computation. This OMJR is implemented in Matlab 2011a on a Dell Optiplex 755 computer (Intel Core 2 Duo CPU E8400 @ 3.00 GHZ, 1.96 GB of RAM).

The average delay costs obtained by using FCFS, OSJR and OMJR for the 10 different delay intervals are shown in Figure 6-11. The average delay cost increases as the number of delay minutes and delayed junction areas increases.

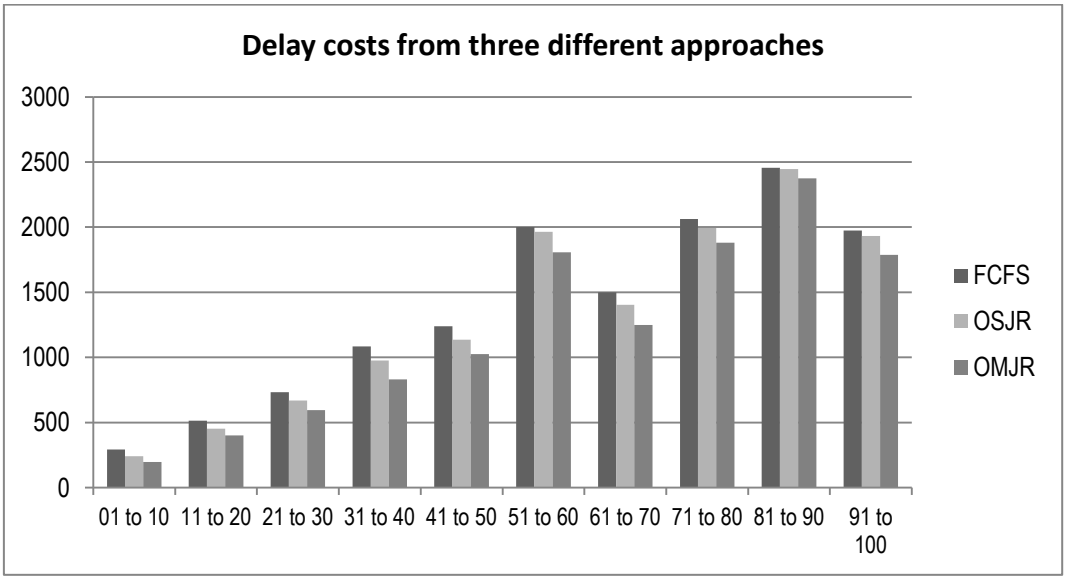


Figure 6-11: Average delay costs (in £) obtained by using FCFS, OSJR and OMJR for the 10 different delay intervals

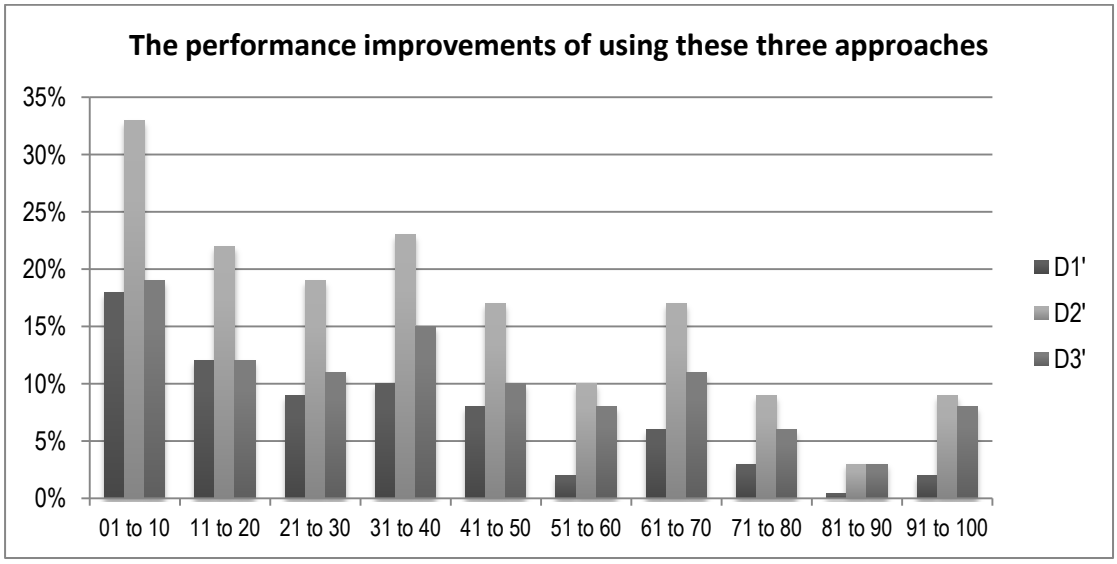


Figure 6-12: The performance improvements obtained by using FCFS, OSJR and OMJR for the 10 different delay intervals on average

Finally, the performance improvements (*D1'*, *D2*' and *D3'*) from each delay interval are shown in Figure 6-12. It is clear that the OMJR approach achieves a better solution in every delay interval than FCFS. Furthermore, it can achieve more than 30% benefit in slight delay intervals and gives about 20% benefit in moderate delay interval. In delay interval 9 (scenarios 81 to 90), every train in the control region is delayed, and so FCFS is highly likely to provide a good solution which can make the trains run as according to their initial plan (timetable). In this case, OSJR only can improve 0.2%, but OMJR can still achieve an improvement of 3% over FCFS. Furthermore, OMJR always achieves a better solution than OSJR in these case studies.

## 6.7 Conclusions

Optimal multi-junction rescheduling is a complex problem. A timetable disturbance could cause knock-on delays at more than one junction area. This chapter introduced a real-time optimal multi-junction rescheduling system that can deal with very busy control regions. Trains can be rescheduled by time stage and it is capable of providing an updated timetable for the current stage and dealing with newly ordered trains arriving in the following stages.

Since the stages and the number of trains at the junction are variable in the new system, methods such as the traditional decision tree cannot be generated, as well as graphical algorithms (e.g. STS). The hybrid algorithm is modified here, and the FCFS solutions of each stage are used as a start of the search to improve the searching quality since it provides reasonable initial solutions.

By using the OMJR approach in the 100 delay scenarios, the delay cost can be reduced by an average of £166 compared with FCFS. In addition, the delay cost can

be reduced by an average of £107 compared with the OSJR approach. Moreover, by comparing the 100 delay scenarios within the 10 different delay intervals, an important conclusion can be made that, as the initial delay time of the trains or the number of delayed trains increases, the rescheduling effects become closer to the FCFS results. In other words, the FCFS solution is likely to cause a knock-on delay and provide a high overall delay cost in slight delay scenarios. The OMJR approach, can give much better solutions than FCFS in slight delay scenarios.

# Chapter 7: Conclusion and Further Work

## 7.1 General Development

In this thesis, a number of existing railway rescheduling optimisation approaches have been introduced and applied to a series of common scenarios for minimising the delay. Typically, researchers compare only one or two rescheduling algorithms to FCFS. To the best of the author's knowledge, this thesis presents the first extensive comparative study of different railway rescheduling algorithms to a common benchmark case study. The results have determined the appropriate applications for each approach by comparing them in the same delay scenarios.

It can be concluded that, due to the computation time required, currently, it is most appropriate to implemented brute force and dynamic programming in simple scenarios. However, they could be used in more demanding cases as computers increase in capacity. FCFS is the quickest and simplest method, however, compared with the other methods considered in this comparison, the results are suboptimal. FCFS should primarily be used as an initial solution to initiate other methods, or as a fallback solution if insufficient time is available to use other methods.

DTBE and TS perform well in complex scenarios where little computation time is available. However, a compromise must be made on the quality of the solution since GA, SA and ACO all perform comparably well with regard to solution optimality,

however, for the scenarios considered in this comparison, ACO provides close to optimum solutions within practical computation times.

In the comparison, ACO on average gives the closest solution to the optimum and takes the second shortest computation time of the evolutionary algorithms in the four case study delay scenarios. ACO gives the optimal solution in three quarters of these scenarios. Compared with GA, it has been shown that ACO can achieve a better solution with a shorter computation time. Furthermore, ACO is not only a terminable algorithm, but also a controllable algorithm, so it is appropriate to implement ACO at any junction and in any scenario. ACO could be terminated and the current best result taken at any time; the computation speed can be controlled by the parameter settings based on the time available to produce a rescheduling response. Therefore, it can be concluded that ACO is the most appropriate of the currently available algorithms for implementation in railway rescheduling problems and as the subject of further research.

## 7.2 Main Achievement

Based on the conclusions of the comparison of the studied railway rescheduling approaches, ACO has been selected for further study. In this research, the focus was upon eliminating or reducing the weakness of ACO, which is that ACO is easily trapped in local minima. Generally, this is because all but one of the pheromone coefficients of the upper level nodes decrease rapidly within a few iterations, resulting in potential solution paths being unlikely to be explored later on. Local search gives a change in train order, mixing up the search, but on its own local search also tends to get trapped into local minima. However, the hybridisation of the DTBACO with the

local search helps the combined algorithm to escape local minima, while effectively guiding the search towards a good or optimal solution.

A comparison between ACO, BF, and FCFS using randomised train delays shows the improvement that is possible using the hybrid approach. The comparison results have confirmed the strength of the hybrid algorithm using a discussion of average value analysis, possibility analysis and worst case analysis. Over the 100 considered scenarios, the hybrid algorithm has a probability of 92% of achieving the optimal solution, and it gives an average cost very close to that achieved with BF and much better than FCFS. Furthermore, this hybrid algorithm only takes 4% of the computing time of BF. In terms of the quality of the solution, the hybrid algorithm shows a significant improvement over ACO alone, while the speed of convergence remains similar.

From the previous discussion it is known that a delayed train could cause knock-on delays, which may lead to a whole network delay. An optimal rescheduling decision of the trains at one junction area may not be the optimal decision for the whole network. Therefore, a novel approach is designed in this thesis to solve the optimal multi-junction train rescheduling problem.

This novel real-time optimal multi-junction rescheduling approach is composed of two main steps: “prediction” and “control”. In the prediction step, the unit time, considered trains and the control stages are calculated by conflict-free train running time predictions. In the control step, a multi-junction rescheduling system searches for the optimal rescheduling solution, which contains a series of rescheduling solutions by stage.

The hybrid algorithm is used in the control step. However, since the stages and the number of trains at the junction are variable in the new system, methods such as traditional decision tree cannot be generated. Furthermore, evolutionary algorithms such as TS and GA struggle to guide the search direction to the optimum when the constraints are different. Therefore, the modified hybrid algorithm is used here since it can readily give new solutions with different constraints. This modified hybrid algorithm also can be used to full advantage in each single junction area within each iteration since it will not produce illegal and infeasible solutions since the appropriate decision tree is re-generated in each iteration.

This OMJR approach is designed for dealing with any delay scenarios in any control region. It has been shown that it provides a better solution than FCFS in four typical delay scenarios. Further testing of the OMJR approach in 100 random delay scenarios showed that the delay cost was reduced by an average of £166 compared with FCFS. Furthermore, an average improvement in delay cost of £107 over the OSJR approach was obtained. It is important to recognise that all the above improvements are based on a small control region of Britain's railway network and that only 30 minutes of operations time are covered. It is expected that larger control regions and longer time durations will result in a significantly better performance, once OMJR is applied.

## 7.3 Further Work

Based on the introduction and case studies of the optimal single-junction rescheduling approach and the optimal multi-junction control approach, there are two areas of further work that can be developed from this thesis:

1. Further work on the rescheduling approach; and

2. Further work taking optimal energy consumption into account.

Although the hybrid OSJR approach was shown to be more effective than the other algorithms in the case studies, it could be improved by developing a multi-started parallel computing processing. This means two or more algorithms run at the same time, possibly with from different initial solutions, and their current best solutions are compared to each other every few iterations. Progress is then made from the best of these. By doing this, it will increase the probability of finding the optimal solution, as well as making progress more quickly.

Furthermore, the algorithms themselves can be made parallel. For example, in ACO, ten ant paths could be processed in parallel, and so the computation time of ACO could be significantly reduced in this way (Ling *et al.*, 2012).

With the increasing concern for the environment and energy efficiency, the transportation industry is facing increasing pressure to improve the energy performance of its vehicles. Oil and electricity prices have increased sharply in the last 10 years. It is imperative to optimise the rail network for energy efficiency. Avoiding unnecessary train stops and providing driving instructions to drivers could lead to carbon emission reduction and performance improvement. An energy efficient driving strategy can be designed as another constraint of these. In the train running estimation modelling of this thesis (which is detailed in Appendix A), since the train parameters, gradients, braking rate and running times are known, the energy consumption of each section (acceleration, coasting and braking) can be worked out. Therefore, the optimal train running trajectory of each train considered during rescheduling could be studied in combination with delay cost reduction.

# References

ABIDO, M. A. 2001. Parameter optimization of multimachine power system stabilizers using genetic local search. *International Journal of Electrical Power & Energy Systems,* 23**,** 785-794.

ALBRECHT, T. 2004. Train Running Time Control Using Genetic Algorithms for the Minimization of Energy Costs in DC Rapid Transit Systems. *In:* ROY, A. T. A. R. (ed.) *Decision Engineering Report Series.* Cranfield University.

ALBRECHT, T. 2009. The Influence of Anticipating Train Driving on the Dispatching Process in Railway Conflict Situations. *Networks & Spatial Economics,* 9**,** 85-101.

ALBRECHT, T. & OETTICH, S. 2002. A new integrated approach to dynamic-schedule synchronization and energy-saving train control. *Computers in Railways VIII,* 13**,** 847-856.

ALEXANDER, S. 2003. *Combinatorial Optimization : polyhedra and efficiency,* Berlin, Springer.

ANANY, L. 2003. *Introduction to The Design & Analysis of Algorithms,* United States, Pearson Education.

AZIZI, N. & ZOLFAGHARI, S. 2004. Adaptive temperature control for simulated annealing: a comparative study. *Computers & Operations Research,* 31**,** 2439-2451.

BACK, T. 1996. *Evolutionary Algorithms in Theory and Practice,* New York, Oxford University Press.

BALFE, N., WILSON, J. R., SHARPLES, S. & CLARKE, T. 2007. Analysis of current UK rail signalling systems. *Human Factors and Ergonomics Society*

*European Chapter conference on Human Factors for Assistance and Automation.* Braunschweig, Germany.

BEST, N. & HYLAND, B. 2012. UK Railway Systems Reliability - Modelling the Future - a case study *In:* UK (ed.). Network Rail.

BOCHARNIKOV, Y. V., TOBIAS, A. M., ROBERTS, C., HILLMANSEN, S. & GOODMAN, C. J. 2007. Optimal driving strategy for traction energy saving on DC suburban railways. *Iet Electric Power Applications,* 1**,** 675-682.

BROWN, D. E., HUNTLEY, C. L., MARKOWICZ, B. P. & SAPPINGTON, D. E. 1992. Rail Network Routing and Scheduling Using Simulated Annealing. *1992 IEEE International Conference on Systems, Man, and Cybernetics, Vol 1***,** 589-592.

BRUCKER, P. 2007. *Scheduling algorithms* Berlin ; New York, Springer.

CHEN, C., TIAN, Y. X., ZOU, X. Y., CAI, P. X. & MO, J. Y. 2005. A hybrid ant colony optimization for the prediction of protein secondary structure. *Chinese Chemical Letters,* 16**,** 1551-1554.

CHEN, L., SCHMID, F., DASIGI, M., NING, B., ROBERTS, C. & TANG, T. 2010. Real-time train rescheduling in junction areas. *Proc. IMechE, Part F: J. Rail and Rapid Transit*.

CHENG, Y. 1996. Optimal train traffic rescheduling simulation by a knowledge-based system combined with critical path method. *Simulation Practice and Theory,* 4**,** 399-413.

CHENG, Y. 1998a. Hybrid simulation for resolving resource conflicts in train traffic rescheduling. *Computers in Industry,* 35**,** 233-246.

CHENG, Y. 1998b. Rule-based train traffic reactive simulation model. *Applied Artificial Intelligence,* 12**,** 5-27.

CHIANG, T. W., HAU, H. Y., CHIANG, H. M., KO, S. Y. & HSIEH, C. H. 1998. Knowledge-based system for railway scheduling. *Data & Knowledge Engineering,* 27**,** 289-312.

CHOU, Y. H. 2009. *Distributed Approach for Rescheduling Railway Traffic in the Event of Disturbances.* PhD Thesis, The University of Birmingham.

CHOU, Y. H., WESTON, P. & ROBERTS, C. 2007. Dynamic Distributed Control for Real-time Rescheduling of Railway Networks. *Rail Hannover.* Hannover.

CIANCARINI, P. & FAVINI, G. P. 2010. Solving Kriegspiel Endings with Brute Force: The Case of KR vs. K. *Advances in Computer Games,* 6048**,** 136-145.

CORMAN, F., D'ARIANO, A., PACCIARELLI, D. & PRANZO, M. 2009. Evaluation of green wave policy in real-time railway traffic management. *Transportation Research Part C-Emerging Technologies,* 17**,** 607-616.

CORMAN, F., D'ARIANO, A., PACCIARELLI, D. & PRANZO, M. 2010. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B-Methodological,* 44**,** 175-192.

CORMAN, F., D'ARIANO, A., PACCIARELLI, D. & PRANZO, M. 2012. Optimal inter-area coordination of train rescheduling decisions. *Transportation Research Part E-Logistics and Transportation Review,* 48**,** 71-88.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. & STEIN, C. 2001. *Introduction to Algorithms,* Cambridge, MIT Press.

D'ARIANO, A. 2008. *Improving Real-Time Train Dispatching: Models, Algorithms and Applications,* Delft, TRAIL.

D'ARIANO, A., PACCLARELLI, D. & PRANZO, M. 2007a. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research,* 183**,** 643-657.

D'ARIANO, A., PRANZO, M. & HANSEN, I. A. 2007b. Conflict resolution and train speed coordination for solving real-time timetable perturbations. *IEEE Transactions on Intelligent Transportation Systems,* 8**,** 208-222.

DAVIDSSON, P., HENESEY, L., RAMSTEDT, L., TORNQUIST, J. & WERNSTEDT, F. 2005. An analysis of agent-based approaches to transport logistics. *Transportation Research Part C-Emerging Technologies,* 13**,** 255-271.

DINGLER, M. 2008. Using the RTC Simulation Model to Evaluate Effects of Operating Heterogeneity on Railway Capacity. *In:* WILSON, E. (ed.) *The William W. Hay Railroad Engineering Seminar Series.* ILLINOIS.

DORIGO, M. 1992. *Optimization, Learning and Natural Algorithms.* PhD, Politecnico di Milano.

DORIGO, M., DI CARO, G. & GAMBARDELLA, L. M. 1999. Ant algorithms for discrete optimization. *Artificial Life,* 5**,** 137-172.

DORIGO, M., GAMBARDELLA, L. M., MIDDENDORF, M. & STUTZLE, T. 2002. Special section on ant colony optimization. *IEEE Transactions on Evolutionary Computation,* 6**,** 317-320.

DORIGO, M. & STÜTZLE, T. 2004. *Ant colony optimization,* Cambridge, Mass., MIT Press.

DREYFUS, S. 2002. Richard Bellman on the birth of dynamic programming. *Operations Research,* 50**,** 48-51.

ELMIHOUB, T., HOPGOOD, A. A., NOLLE, L. & BATTERSBY, A. 2004. Performance of hybrid genetic algorithms incorporating local search. *Esm'2004: 18th European Simulation Multiconference***,** 154-160.

FAN, B., ROBERTS, C. & WESTON, P. 2011. A hybrid algorithm for optimal junction traffic control. *In:* HANSEN, I. A. (ed.) *4th International Association of Railway Operations Research.* Rome.

FAY, A. 2000. A fuzzy knowledge-based system for railway traffic control. *Engineering Applications of Artificial Intelligence,* 13**,** 719-729.

GAMBARDELLA, L. M. & DORIGO, M. 2000. An ant colony system hybridized with a new local search for the sequential ordering problem. *Informs Journal on Computing,* 12**,** 237-255.

GAMBARDELLA, L. M., TAILLARD, E. D. & DORIGO, M. 1999. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society,* 50**,** 167-176.

GIBSON, S., COOPER, G. & BALL, B. 2002. Developments in transport policy - The evolution of capacity charges on the UK rail network. *Journal of Transport Economics and Policy,* 36**,** 341-354.

GLOVER, F. 1989. Tabu Search — Part I. *INFORMS Journal on Computing,* 1**,** 190-206.

GLOVER, F. 1990. Tabu Search — Part II. *INFORMS Journal on Computing,* 2**,** 4-32.

GOOSSENS, J. W., VAN HOESEL, S. & KROON, L. 2004. A branch-and-cut approach for solving railway line-planning problems. *Transportation Science,* 38**,** 379-393.

GU, J. & HUANG, X. F. 1994. Efficient Local Search with Search Space Smoothing - a Case-Study of the Traveling Salesman Problem (Tsp). *Ieee Transactions on Systems Man and Cybernetics,* 24**,** 728-735.

HANDL, J., KNOWLES, J. & DORIGO, M. 2004. Strategies for the increased robustness of ant-based clustering. *Proceding of Engineering Self-Organising Systems,* 2977**,** 90-104.

HANSEN, I. A. & PACHL, J. 2008. *Railway Timetable & Traffic*, Eurailpress.

HILL, R. J. 1994a. Electric Railway Traction .1. Electric Traction and DC Traction Motor-Drives. *Power Engineering Journal,* 8**,** 47-56.

HILL, R. J. 1994b. Electric Railway Traction .2. Traction Drives with 3-Phase Induction-Motors. *Power Engineering Journal,* 8**,** 143-152.

HILLMANSEN, S., ROBERTS, C., MCGORDON, A. & JENNINGS, P. 2008. Concept Validation for Hybrid Trains. Birmingham: Birmingham Research and Development Limited.

HILLMANSEN, S., ROBERTS, C., MCGORDON, A. & JENNINGS, P. 2009. DMU Hybrid Concept Evaluation. Birmingham: Birmingham Research and Development Limited.

HO, S. C. & HAUGLAND, D. 2011. Local search heuristics for the probabilistic dial-a-ride problem. *Or Spectrum,* 33**,** 961-988.

HO, T. K., NORTON, J. P. & GOODMAN, C. J. 1997. Optimal traffic control at railway junctions. *Iee Proceedings-Electric Power Applications,* 144**,** 140-148.

HO, T. K. & YEUNG, T. H. 2000. Railway junction conflict resolution by genetic algorithm. *Electronics Letters,* 36**,** 771-772.

HO, T. K. & YEUNG, T. H. 2001. Railway junction traffic control by heuristic methods. *Iee Proceedings-Electric Power Applications,* 148**,** 77-84.

HOLLAND, J., H. 1975. *Adaptation in Natural and Artificial Systems,* Ann Arbor, University of Michigan Press.

ISAAI, M. T. & SINGH, M. G. 2001. Hybrid applications of constraint satisfaction and meta-heuristics to railway timetabling: A comparative study. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews,* 31**,** 87-95.

ISMAIL, A., QUAFAFOU, M., NACHOUKI, G. & HAJJAR, M. 2010. A decision tree for queries routing in hierarchical peer-to-peer network. *2010 The 7th International Conference on Informatics and Systems, Proceedings***,** 1-8.

JACOBS, G. 2005. *Railway Track Diagrams Book 4: Midlands & North West,* Bradford, Trackmaps.

JÜNGER, M. 2001. *Computational combinatorial optimization : optimal or provably near-optimal solutions* Springer, Berlin; London.

KE, B. R., LIN, C. L. & LAI, C. W. 2011. Optimization of train-speed trajectory and control for mass rapid transit systems. *Control Engineering Practice,* 19**,** 675-687.

KIRKPATRICK, S., GELATT, C. D. & VECCHI, M. P. 1983. Optimization by Simulated Annealing. *Science,* 220**,** 671-680.

KODEEKHA, E. 2007. Brute force method for lot streaming in FMS scheduling problems. *Ines 2007: 11th International Conference on Intelligent Engineering Systems, Proceedings***,** 179-184.

KUHN, M. 1998. Automatic route setting integrated in a modern traffic management system. *International Conference on Developments in Mass Transit Systems***,** 140-145.

LAWSON, H. W., WALLIN, S., BRYNTSE, B. & FRIMAN, B. 2001. Twenty years of safe train control in Sweden. *Eighth Annual Ieee International Conference and Workshop on the Engineering of Computer Based Systems, Proceedings***,** 289-297.

LEW, A. & MAUCH, H. 2007. *Dynamic Programming,* Springer.

LING, C., SUN, H. Y. & SHU, W. 2012. A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem. *Information Sciences,* 199**,** 31-42.

LIU, B. & MENG, P. S. 2008. Hybrid Algorithm Combining Ant Colony Algorithm with Genetic Algorithm for Continuous Domain. *Proceedings of the 9th International Conference for Young Computer Scientists, Vols 1***,** 1819-1824.

LIU, Y. H. 2011. A genetic local search algorithm with a threshold accepting mechanism for solving the runway dependent aircraft landing problem. *Optimization Letters,* 5**,** 229-245.

LOCKYER, K. & GORDON, J. 1991. *Critical Path Analysis and Other Project Network Techniques,* Prentice Hall.

LOHER, U. 1998. Efficiency of First-Come First-Served Algorithms. *1998 IEEE International Symposium on Information Theory - Proceedings***,** 108.

LOPEZ, M. R. & TUNON, M. I. C. 2005. Design and use of the CPAN Branch & Bound for the solution of the Travelling Salesman Problem (TSP). *Simulation in Wider Europe***,** 760-765.

LUKASZEWICZ, P. 2007. Running resistance - results and analysis of full-scale tests with passenger and freight trains in Sweden. *Proceedings of the Institution of Mechanical Engineers Part F-Journal of Rail and Rapid Transit,* 221**,** 183-193.

MANIEZZO, V., STUTZLE, T. & VOB, S. 2009. *Matheuristics : hybridizing metaheuristics and mathematical programming,* New York, Springer.

MASCIS, A. & PACCIARELLI, D. 2002a. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research,* 143**,** 498-517.

MASCIS, A. & PACCIARELLI, D. 2002b. Models and algorithms for traffic management of rail networks. Roma.

MAZZARELLO, M. & OTTAVIANI, E. 2007. A traffic management system for real-time traffic optimisation in railways. *Transportation Research Part B-Methodological,* 41**,** 246-274.

MERZ, P. & FREISLEBEN, B. 1997. Genetic local search for the TSP: New results. *Proceedings of 1997 Ieee International Conference on Evolutionary Computation (Icec '97)***,** 159-164.

PAAR, C. & PELZL, J. 2010. *Understanding cryptography : a textbook for students and practitioners*, Heidelberg, Springer.

PADBERG, M. & RINALDI, G. 1987. Optimization of a 532-City Symmetrical Traveling Salesman Problem by Branch and Cut. *Operations Research Letters,* 6**,** 1-7.

PENG, Q. & YING, S. 2008. *Organization of Train Operation*, Southwest Jiaotong University.

PHAM, D. T. & KARABOGA, D. 2000. *Intelligent Optimisation Techniques*, Springer-Verlag, London.

PROENCA, H. & OLIVEIRA, E. 2004. MARCS - Multi-agent railway control system. *Advances in Artificial Intelligence - Iberamia 2004,* 3315**,** 12-21.

QI, C. M., TIAN, W. J. & SUN, Y. C. 2009. Hybrid Ant Colony Algorithm for QAP. *2009 Isecs International Colloquium on Computing, Communication, Control, and Management, Vol III,* 213-216.

QI, M. Y., MIAO, L. X., ZHANG, L. & XU, H. Y. 2008. A New Tabu Search Heuristic Algorithm for the Vehicle Routing Problem with Time Windows. *2008 International Conference on Management Science & Engineering (15th), Vols I and II, Conference Proceedings***,** 1648-1653.

REEVES, C. R. 1993. *Modern heuristic techniques for combinatorial problems,* New York, Halsted Press.

ROCHARD, B. P. & SCHMID, F. 2000. A review of methods to measure and calculate train resistances. *Proceedings of the Institution of Mechanical Engineers Part F-Journal of Rail and Rapid Transit,* 214**,** 185-199.

ROTHLAUF, F. 2006. *Representations for genetic and evolutionary algorithms,* Berlin, Springer.

RUSHTON, J. N. 2004. Learning natural language using genetic algorithms. *IC-AI '04 & Mlmta'04 , Vol 1, Proceedings***,** 1092-1096.

SCHMID, U. & BLIEBERGER, J. 1992. Some Investigations on Fcfs Scheduling in Hard Real-Time Applications. *Journal of Computer and System Sciences,* 45**,** 493-512.

SHAH, S. & SASTRY, P. S. 1999. New algorithms for learning and pruning oblique decision trees. *Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews,* 29**,** 494-505.

SHANG, G., JIANG, X. Z. & TANG, K. Z. 2007. Hybrid algorithm combining ant colony optimization algorithm with genetic algorithm. *Proceedings of the 26th Chinese Control Conference, Vol 5***,** 701-704.

SOSIC, R. & GU, J. 1994. Local Search with Conflict Minimization - a Case-Study of the N-Queens Problem. *Ieee Transactions on Knowledge and Data Engineering,* 6**,** 661-668.

STOLK, A. 1998. Automatic conflict detection and advanced decision support for optimal usage of railway infrastructure - Purpose and concepts. *Computers in Railways VI,* 2**,** 629-638.

TEODOROVIC, D. 2008. Swarm intelligence systems for transportation engineering: Principles and applications. *Transportation Research Part C-Emerging Technologies,* 16**,** 651-667.

THEEG, G. & VLASENKO, S. (eds.) 2009. *Railway Signalling & Interlocking,* Hamburg: Eurailpress.

TORNQUIST, J. & PERSSON, J. A. 2005. Train Traffic Deviation Handling Using Tabu Search and Simulated Annealing. *2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on System Sciences*.

WANNER, E. F., TAKAHASHI, R. H. C., GUIMARAES, F. G. & RAMIREZ, J. A. 2007. Hybrid genetic algorithms using quadratic local search operators. *Compel-the International Journal for Computation and Mathematics in Electrical and Electronic Engineering,* 26**,** 773-787.

WEGELE, S., SLOVÁK, R. & SCHNIEDER, E. Year. Automatic dispatching of train operations using a hybrid optimisation method. *In:* CD-ROM Proceedings of the 8th World Congress on Railway Research, 2008 Seoul, Korea.

WESTON, P., GOODMAN, C., TAKAGI, R., BOUCH, C., ARMSTRONG, J. & PRESTON, J. 2006. Minimising Train Delay in a Mixed-Traffic Railway Network with Consideration of Passenger Delay. *WCRR*. Montreal, Canada.

YAGHINI, M., FOROUGHI, A. & NADJARI, B. 2011. Solving railroad blocking problem using ant colony optimization algorithm. *Applied Mathematical Modelling,* 35**,** 5579-5591.

YEUNG, T. H. & HO, T. K. 2000. Railway junction conflict resolution by local search method. *Computers in Railways VII,* 7**,** 769-778.

ZHENG, S. J., SHU, W. N. & GAO, L. 2006. Task scheduling using parallel genetic simulated annealing algorithm. *2006 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI 2006), Proceedings***,** 46-50.

ZIDI, S. & MAOUCHE, S. 2006. Ant colony optimization for the rescheduling of multimodal transport networks. *2006 IMACS: Multiconference on Computational Engineering in Systems Applications, Vols 1***,** 965-971.

# Appendix A: Train Running Time Estimation

For all of the rescheduling methods in this thesis, when the sequence of the trains to pass the junction area is given, it is very important to estimate the running time as closely to the real operational time as possible since the correctness of estimation significantly affects the accuracy of the rescheduling. In addition, the simulator should estimate the running time quickly; this is because the simulator will run as many times as the search method requires. In this chapter, an accurate and speedy train running time estimation method is introduced.

## A.1 Brief Overview of Traction System

### 7.3.1 A.1.1 Traction Effort and Vehicle Resistance

Train performance calculations are based on the relationship defined in Newton's Second Law (Hill, 1994a, Hill, 1994b). The factors in this equation are explained below.

$$Acceleration = \frac{d\,v}{d\,t} = \frac{Tractive\ Effort - Train\ Resistance - Mg\sin\alpha}{Effecive\ Mass} \qquad \text{(A-1)}$$

#### *A.1.1.1 Train Resistance*

Train motion is opposed by friction of various sorts, principally bearing friction and aerodynamic drag. Bearing friction is mostly characterised by a constant component proportional to weight and a viscous term proportional to speed and weight. Aerodynamic drag also exhibits viscous characteristics but tends to be mostly

proportional to speed squared or even cubed. Davis gives an equation based on rolling stock measurements (Rochard and Schmid, 2000).

$$Drag\ Force = a + b\,v + c\,v^2 \quad \text{(A-2)}$$

where *a*, *b*, and *c* are Davis coefficients for particular stock in particular conditions.

#### *A.1.1.2 Gradient*

If a train is on a slope, there is an angle α between the vertical force $Mg$ and the horizontal, which can be resolved into $Mg\,sin\,\alpha$ along the track and $Mg\cos\alpha$ perpendicular to the track.

#### *A.1.1.3 Effective Mass*

When a train accelerates along a track, the total mass is accelerated linearly but the rotating parts are also accelerated in a rotational sense.

$$Effective\ Mass = Actual\ Tare\ Mass \times (1 + rotary\ Allowance) + Passenger\ or\ Freight\ Load \quad \text{(A-3)}$$

The value of the rotary allowance varies from 5% to 15% depending on the number of motored axles, the gear ratio and the type of vehicle construction.

### 7.3.2 A.1.2 Moving Sections

A train running trajectory is made up of four moving sections as shown in Figure A-1 acceleration, cruising, coasting and braking.

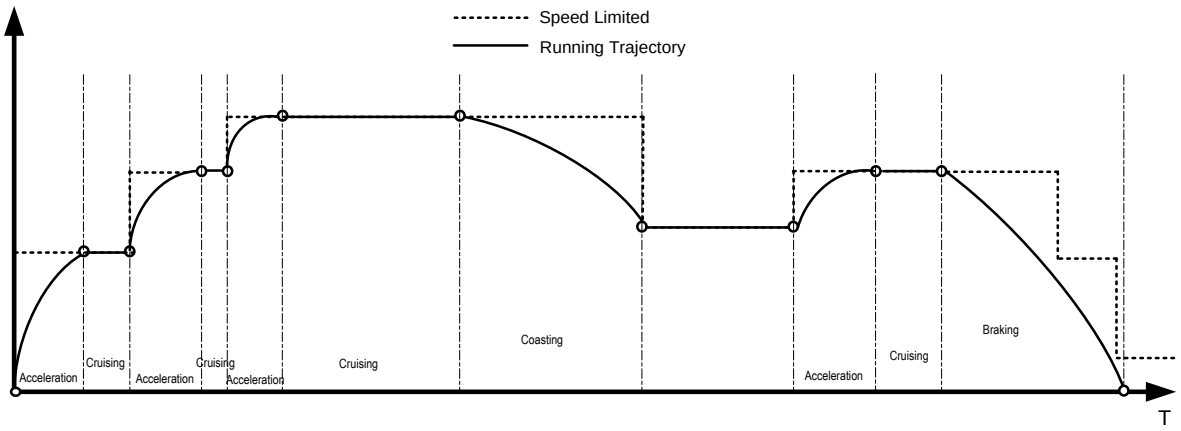


Figure A-1: An example of a train running trajectory (Fan)

In general, an energy efficient driving strategy is applied according to the timetable. However, in most delay cases, delay penalties are much more expensive than energy costs. Energy efficient driving, therefore, is not considered in this simulator. This means that the train will run as fast as possible, at maximum acceleration, maximum cruising speed under the speed limit and service braking.

### *A.1.2.1 Acceleration Section*

Since the various resistances depend on the velocity, as equations A-1 and A-2 show, if a constant acceleration rate is used in this simulator, it will greatly affect the estimation correctness. Therefore, in the acceleration part, all the available tractive effort is used when the shortest running time is calculated. In order to have simulation results as close to actual train running performance as possible, the tractive effort is set as constant at its maximum value in the acceleration section.

In order to find the acceleration time quickly during rescheduling, two homogeneous strip modelling methods are often used to replace the differential equations. Namely, distance homogeneous strip and acceleration homogeneous strip modelling. In this

simulator, the acceleration is updated at every speed increase of 1 km/h, since the initial speed ($v_o$) and final speed ($v_t$) are known. For example, when a train is accelerated from 0 to 80 km/h, the acceleration is partitioned by 80 fractions; the running distance and time are found from sum of these fractions. In this simulator the gradient is ignored; the equation of acceleration time $t_a$ and distance $d_a$is:

$$t_a(v_o, v_t) = \sum_{i=v_o}^{v_t} \frac{1}{\frac{maximum\ tractive\ effort - (a + b * \frac{i+(i-1)}{2} + c * (\frac{i+(i-1)}{2})^2)}{actual\ tare\ Mass \times (1 + rotary\ allowance)}} \quad \text{(A-4)}$$

$$d_a(v_o, v_t) = \sum_{i=v_o}^{v_t} \frac{i^2 - (i-1)^2}{\frac{maximum\ tractive\ effort - (a + b * \frac{i+(i-1)}{2} + c * (\frac{i+(i-1)}{2})^2)}{actual\ tare\ Mass \times (1 + rotary\ allowance)}} \quad \text{(A-5)}$$

### *A.1.2.2 Cruising Section*

When trains reach their speed limit and are constantly running at the speed limit, this constant speed running section gives constant resistance and is simplified by Newton's formula:

$$S = V \cdot T \quad \text{(A-6)}$$

where the distance $S$ is known from railway track information, and running speed $V$ is known from the speed limits. The running time in the cruising section, then, is found simply from equation A-6.

### *A.1.2.3 Coasting Section*

Although coasting is not applied in this simulator, it is necessary to introduce it here since the coasting section is an important technique for energy efficient driving. As is well known, the coasting section is the only section without effort; reducing energy consumption could be simply understood as making the coasting section as long as

possible. In actual operations, the reiteration of acceleration-coasting-acceleration-coasting is normally used to reduce the energy costs in the time permitted.

#### *A.1.2.4 Braking Section*

Theoretically, the braking section can be understood and calculated as the acceleration section backwards. In most train simulators, however, constant braking acceleration rates are usually used because braking is a short process compared with acceleration and the results are very similar. Therefore, constant brake acceleration rate $b$ is used in this calculation, the braking time $t_b$ and distance $d_b$ is:

$$t_b(v_o, v_t) = \frac{v_o - v_t}{b} \tag{A-7}$$

$$d_b(v_o, v_t) = \frac{v_o^2 - v_t^2}{2b} \tag{A-8}$$

where $v_o$ is the initial speed of this braking section and $v_t$ is the terminal speed.

## A.2 Right of Way Signalling Time Calculation

Before estimating the train journey times one by one according to the order of passing a junction area in the simulation when sequence is known, it is essential to introduce the time superposition method first. This time superposition method is used when more than one train could pass the junction area at the same time. By using this method, the right of way signal time for each train can be given by the time one of the front trains leaves the junction area, and the results of the superposition method will show which train is the front train. Figure A-2 shows an example junction and Figure A-3 presents the related right of way signal time of the sequence 1-2-3-4-5-6-7-8-9-10-11-12; the white block is the right of way signal time for each train; the black block is the train running time of passing the junction area. The outputs of using the

superposition method are finding the exact front train that the current train should follow.

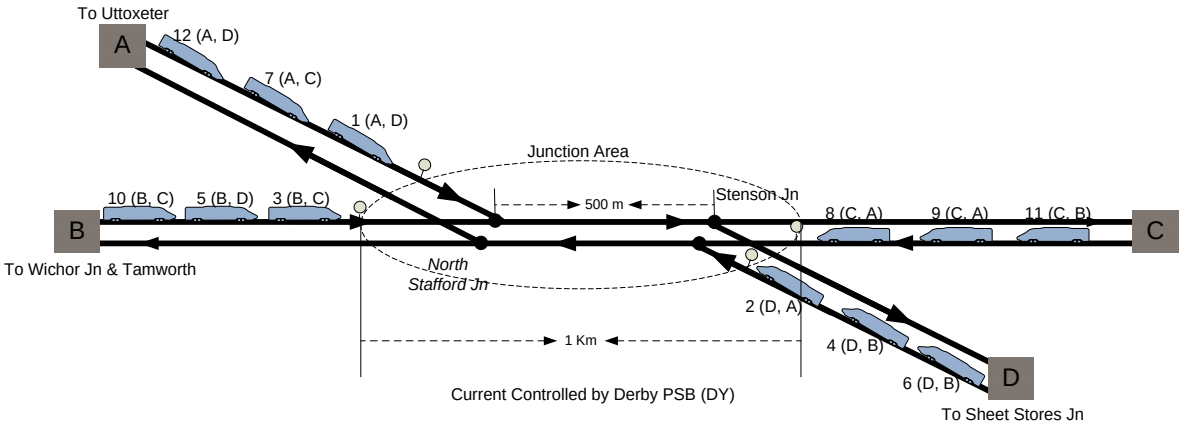


Figure A-2: Example junction (Fan)

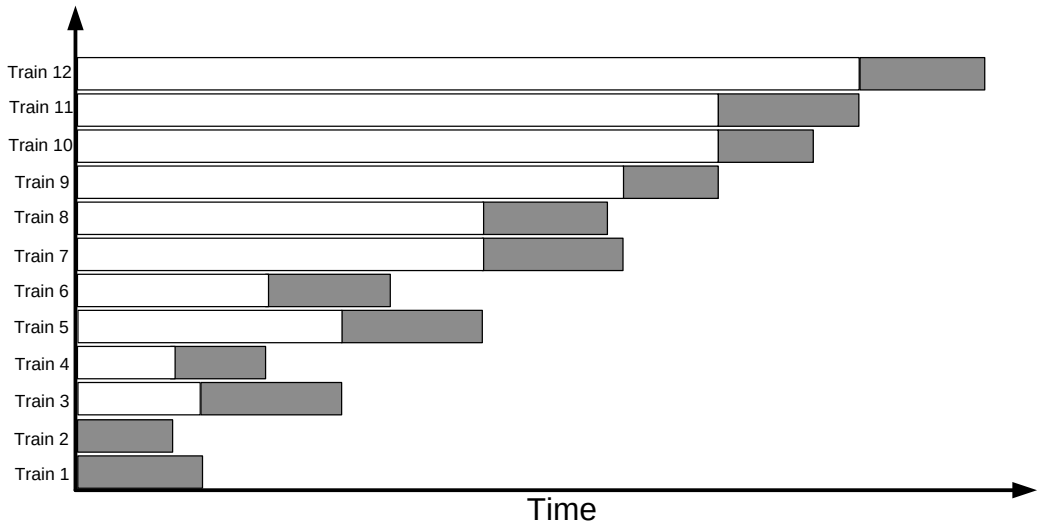


Figure A-3: Related right of way signal time of the example junction (Fan)

The aim of this time superposition technique is to create a conflict-free function and a comparison function. The conflict-free function $cff_i$ (where $i$ is the train number) gives all conflict-free trains in a set (For example, if train 1 could run with train 2, 4 or 6 at the same time as Figure A-3 shows: $cff_1 = \{2, 4, 6\}$). The comparison function checks trains that could run at the same time with previous trains from the

second to the last, as ordered by the conflict-free function. If the train has a conflict with a previous train or the previous train has already been matched with previous trains, this train will be compared with the next. This procedure will repeat one by one until all trains are checked in the sequence.

For example, the results of sequence 1-2-3-4-5-6-7-8-9-10-11-12 are as shown in Figure A-4. The vertical bar isolates the trains which are conflict-free, in other words, this sequence is the exactly same as the sequence of 2-1-4-3-6-5-8-7-9-11-10-12. The arrows indicate the train that is followed. In this sequence, because train 4 is conflict-free with train 1, which is from the opposite direction, train 4 follows train 2 (namely, the right of way signal waiting time for train 4 is only dependent on the leaving junction area time of train 2). Train 3, however, follows both train 1 and train 2 (it needs to wait for both trains to leave the junction area) since train 3 has conflict with train 2, this is why the arrow points to the middle of train 1 and train 2. This result is saved as $f_1 = 0, f_2 = 0, f_3 = 1, 2, f_4 = 2, \cdots\cdots, f_{12} = 10, 11$ for later use.

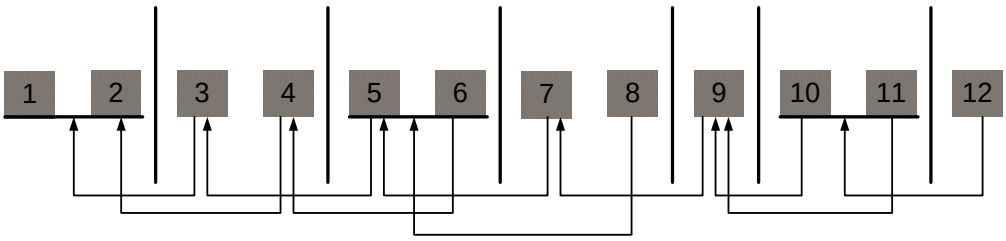


Figure A-4: Conflict-free check results (Fan)

## A.3 Equations of Running Time Estimation

Based on these equations, the entire journey time is found out in different situations. If there is no conflict in the junction area (the train does not need to wait for a right of way signal), the trains will brake to the junction area speed limit and accelerate to the maximum speed, then brake for the station, which is shown in Figure A-5.

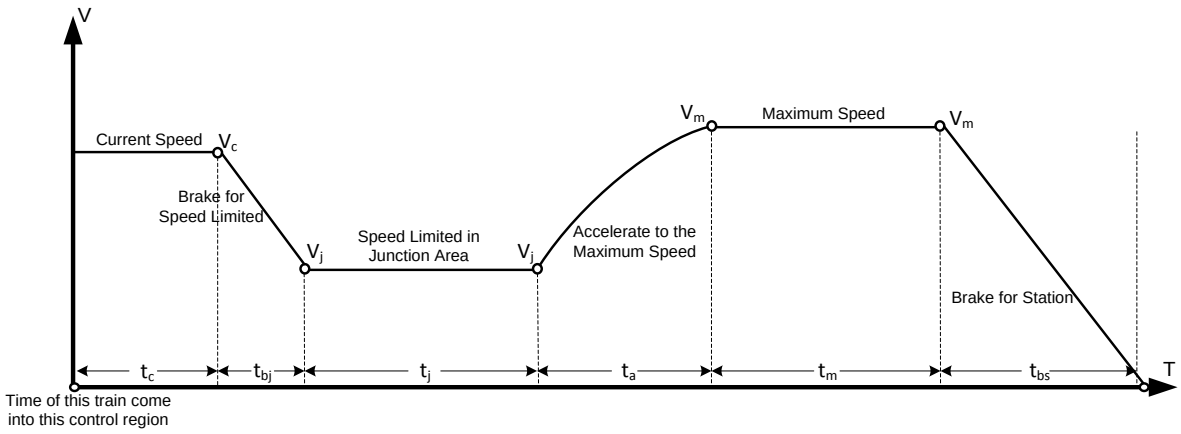


Figure A-5: Traction effort and speed characteristic for propulsion (Fan)

Where the length of the train is $l_t$, the current speed $v_c$, junction area limited speed $v_j$, maximum speed $v_m$, distance to the junction area $d_{oj}$, distance of the junction area $d_j$ and distance from junction area to the station $d_s$ are known, the total journey time from this control region to the next station is:

$$T = t_c + t_{bj} + t_j + t_a + t_m + t_{bs} \tag{A-9}$$

$$T = \frac{d_{oj} - \frac{v_c^2 - v_j^2}{2b}}{v_c} + \frac{v_c - v_j}{b} + \frac{d_j + l_t}{v_j} + t_a(v_j, v_m) + \frac{d_s - d_a(v_j, v_m) - d_b(v_m, 0) - l_t}{v_m} + t_b(v_m, 0) \tag{A-10}$$

When it is not conflict-free at the junction area (the train needs to wait for right of way signal before junction area), trains are normally stopped before the signal and accelerate from 0 again.

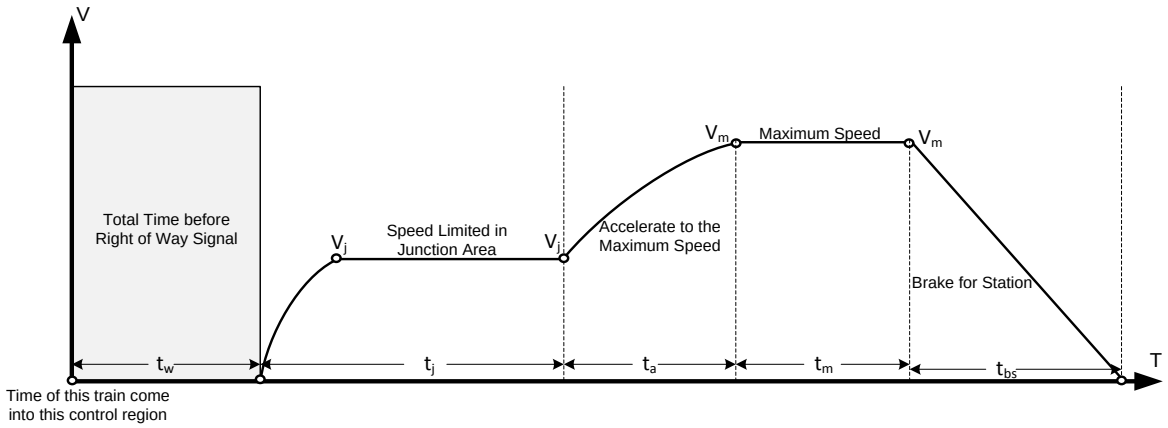


Figure A-6*:* Traction effort and speed characteristic for propulsion (Fan)

Since the right of way signal waiting time $t_w$ is known from the front trains, it is not impossible to find out the running trajectory before the junction area. The journey time in this situation is:

$$T = t_w + t_j + t_a + t_m + t_{bs} \tag{A-11}$$

$$T = t_w + t_a(0, v_j) + \frac{d_j - d_a(0,v_j) + l_t}{v_m} + t_a(v_j, v_m) + \frac{d_s - d_a(v_j,v_m) - d_b(v_m,0) - l_t}{v_m} + t_b(v_m, 0) \tag{A-12}$$

From the automatic train protection (ATP) point of view, the safe distance $d_m$ can be found by the braking distance and the front train length $l_t^f$ (the superscript f means front train):

$$d_m = \frac{v_m^2 - 0}{2b} + l_t^f : +0.12\ km \tag{A-13}$$

The headway $t_h$ is:

$$t_h = \frac{d_m}{v_m^f} \tag{A-14}$$

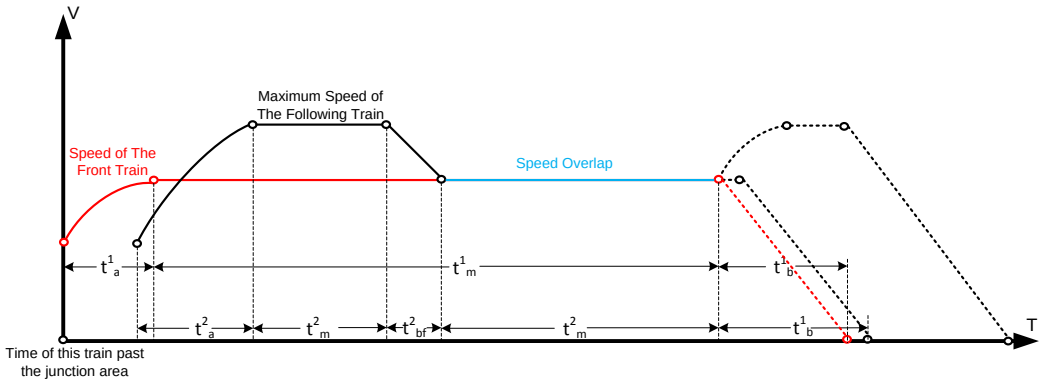


Figure A-7: Traction effort and speed characteristic for propulsion (Fan)

When the following train could run faster than the front train, the following train will be speeded up to follow the front train by ATP. The ATP system works (another journey time calculation equation is used) when the critical function $T + t_h < T^f$ , the journey time for the following train is:

$$T = T^f + \frac{d_m}{v_m} - t_b^f(v_m, 0) + t_b(v_m, 0) \qquad \text{(A-15)}$$

In Figure A-7, the red line is the speed profile of a front train; the black line is the speed profile of following train; the blue line shows that they run as the same speed while maintaining a safe distance.

By using this simulator, every journey time from the control region to the destination can be calculated when the sequence is given.

# Appendix B: Railway Traffic Controller

The Railway Traffic Controller (RTC) is the American standard software which is used to plan and control the traffic. All of the train rescheduling results are checked with the RTC.

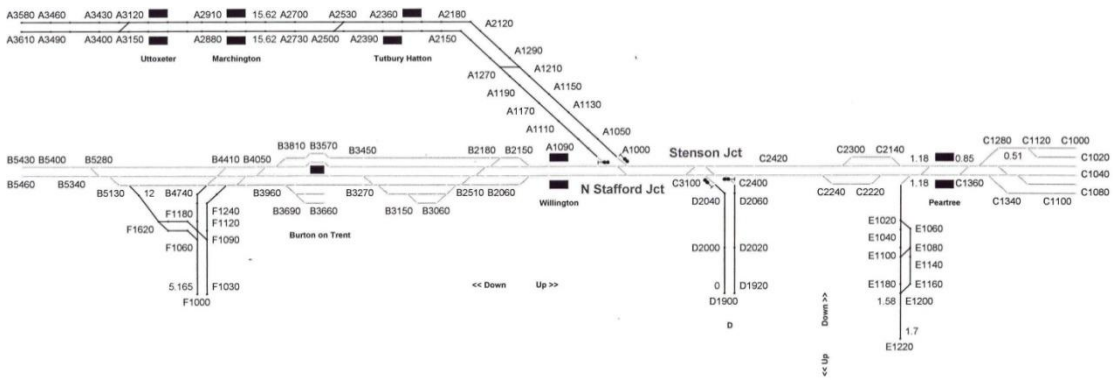


Figure B-1: RTC interface of the case study diagram (Fan)

As Figure B-1 shows, the considered control area is built in the RTC according to real track characteristics, and the parameters of each train are the same as a practical vehicle. The rescheduling solutions of both OSJR and OMJR are tested in the RTC, and the simulated travel time of each train and the corresponding delay cost are almost the same as the simulator for the examples considered in this thesis.

# Appendix C: Railway Track Diagram of the Multi-junction Case Study

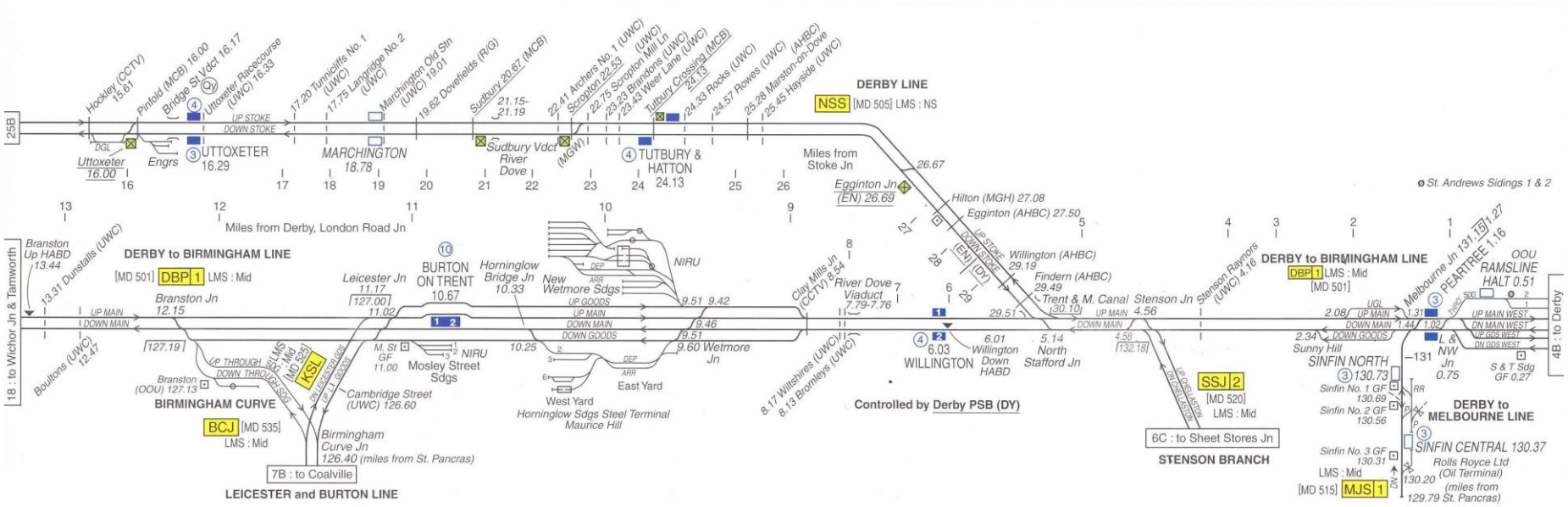


Figure C-1: Railway track diagram of the multi-junction case study (Jacobs, 2005)

The distances of single junction area, junction area to junction area, junction area to station are shown in Figure C-2. In the multi-junction case study, the destinations of considered trains are Lichfield Trent Valley Station, Uttoxeter Racecourse Station, Birmingham New Street Station, Nottingham Station and Derby Station.

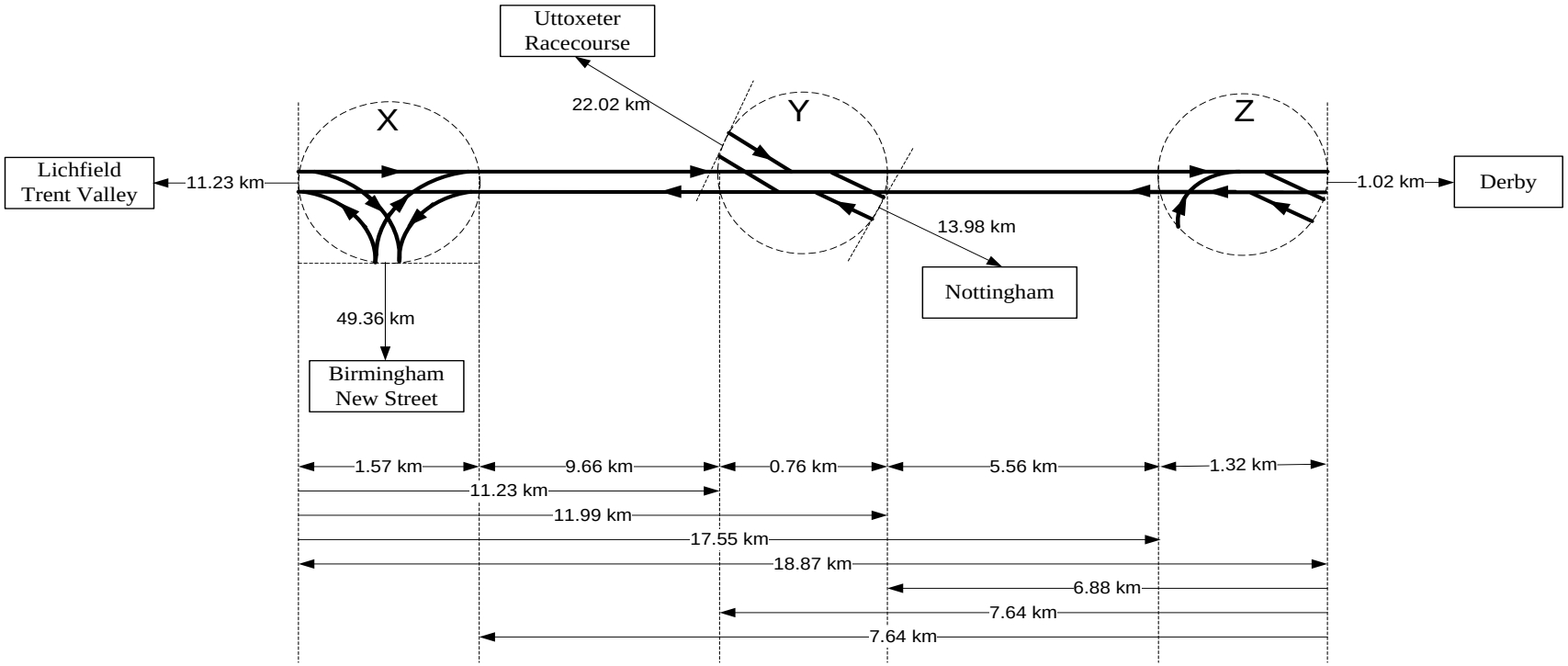


Figure C-2: Distances between junction areas and next station (Fan)

# Appendix D: Publications during PhD Research

[1] FAN, B., ROBERTS, C. & WESTON, P. 2011. A hybrid algorithm for optimal junction traffic control. *In:* HANSEN, I. A. (ed.) *4th International Association of Railway Operations Research.* Rome.

[2] FAN, B., ROBERTS, C. & WESTON, P. 2011. A comparison of algorithms for minimising delay costs in disturbed *Railway traffic scenarios.Railway Transport planning & management*. - accepted and revising