UNIVERSITY OF SOUTHAMPTON
FACULTY OF BUSINESS AND LAW
School of Management

Train Scheduling with Application to the UK Rail Network

by

Banafsheh Khosravi

Thesis for the Degree of Doctor of Philosophy

May 2013

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF BUSINESS AND LAW

SCHOOL OF MANAGEMENT

Doctor of Philosophy

Train Scheduling with Application to the UK Rail Network

by Banafsheh Khosravi

Nowadays, transforming the railway industry for better performance and making the best usage of the current capacity are the key issues in many countries. Operational research methods and in particular scheduling techniques have a substantial potential to offer algorithmic solutions to improve railway operation and control. This thesis looks at train scheduling and rescheduling problems in a microscopic level with regard to the track topology. All of the timetable components are fixed and we aim to minimize delay by considering a tardiness objective function and only allowing changes to the order and to the starting times of trains on blocks. Various operational and safety constraints should be considered. We have achieved further developments in the field including generalizations to the existing models in order to obtain a generic model that includes important additional constraints. We make use of the analogy between the train scheduling problem and job shop scheduling problem. The model is customized to the UK railway network and signaling system. Introduced solution methods are inspired by the successful results of the shifting bottleneck to solve the job shop scheduling problems. Several solution methods such as mathematical programming and different variants of the shifting bottleneck are investigated. The proposed methods are implemented on a real-world case study based on London Bridge area in the South East of the UK. It is a dense network of interconnected lines and complicated with regard to stations and junctions structure. Computational experiments show the efficiency and limitations of the mathematical programming model and one variant of the proposed shifting bottleneck algorithms.

This study also addresses train routing and rerouting problems in a mesoscopic level regarding relaxing some of the detailed constraints. The aim is to make the best usage of routing options in the network to minimize delay propagation. In addition to train routes, train entry times and orders on track segment are defined. Hence, the routing and scheduling decisions are combined in the solutions arising from this problem. Train routing and rerouting problems are formulated as modified job shop problems to include the main safety and operational constraints. Novel shifting bottleneck algorithms are provided to solve the problem. Computational results are reported on the same case study based on London Bridge area and the results show the efficiency of one variant of the developed shifting bottleneck algorithms in terms of solution quality and runtime.

Contents

List of Figures

List of Tables

DECLARATION OF AUTHORSHIP

I, Banafsheh Khosravi, declare that the thesis entitled Train Scheduling with Application to the UK Rail Network and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:

  1. Khosravi, B., Bennell, J. A., and Potts, C. N. (2011). Train Scheduling in the UK using a Job Shop Scheduling Approach. In *Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2011), Phoenix, Arizona, USA.*
  2. Khosravi, B., Bennell, J. A., and Potts, C. N. (2011). A Modified Shifting Bottleneck Procedure for Train Scheduling in the UK. In *Workshop on Mathematical Optimization of Railway-Systems (CPAIOR2011), Berlin, Germany.*
  3. Khosravi, B., Bennell, J. A., and Potts, C. N. (2012). Train Scheduling and Rescheduling in the UK with a Modified Shifting Bottleneck Procedure. In *Proceedings of the 12th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS 2012), Ljubljana, Slovenia.*
  4. Khosravi, B., Bennell, J. A., and Potts, C. N. (2012). Modified Shifting Bottleneck Algorithms for Train Scheduling and Rescheduling in the UK. In *Proceedings of the Vehicle Routing and Logistics Optimization (VeRoLog 2012), Bologna, Italy.*

Signed: ....

Date: ....

Acknowledgements

I would like to thank a number of people who helped me throughout my PhD studies. First of all, I am extremely grateful to my supervisors Professor Julia A. Bennell and Professor Chris N. Potts for firstly giving me the opportunity to study at the University of Southampton and then for their valuable guidance and encouraging support. I greatly appreciate numerous fruitful discussions and the trust and the freedom during the past few years. They gave me fundamental technical advice and the general insight on the workings of academic research. I would like to thank my academic advisor Professor Tolga Bektas for brilliant suggestions and help in the first couple years of my studies.

I am also grateful to my examiners Doctor Patrick Beullens and Doctor Natasha Shakhlevich for all the detailed constructive comments and feedbacks that helped to improve this thesis.

I am also indebted to Professor John Preston and Doctor John Armstrong from Transportation Research Group (TRG) at the University of Southampton. This thesis would not have been the same without their inputs on rail transport. I had the great pleasure of collaborating with them on a railway project in initial steps.

I would further like to thank School of Management and former Faculty of Law, Arts and Social Sciences (LASS) of the University of Southampton and the LANCS Initiative which partially funded and supported this research.

I owe my gratitude to Ms. Rosalind Van Dyke and Mr. Kevin Goodman of Southeastern train operating company in the UK for their technical advice in our visits to the company and providing the data for Kent area which is used in the computational experiments.

I am grateful to all my friends who were supportive and made my time as a PhD student enjoyable. Last but not least, I would like to thank my husband Mohammad and my family for their patience, encouragement and persistent support.

*To my husband and my parents for their endless love,
support and encouragement.*

List of Abbreviations

ASG	Active Schedule Generation
ATC	Apparent Tardiness Cost
ECT	Earliest Completion Time
EST	Earliest Starting Time
FCFS	First-Come First-Served
FIFO	First In First
GA	Genetic Algorithm
MILP	Mixed Integer Linear Programming
MJSS	Modified Job Shop Scheduling
MBJSS	Modified Blocking Job Shop Scheduling
MBPM-JSS	Modified Blocking Parallel Machine Job Shop Scheduling
PM	Parallel Machine
SA	Simulated Annealing
SB	Shifting Bottelneck
SM	Single Machine
TS	Tabu Search

Chapter 1

Introduction

Continuous improvement of reliability, capacity utilization and resilience is a key issue across all transportation systems and many countries invest in technology and processes to improve the current railway system. Effective transportation of passengers and freight is critical to the economic prosperity of a country. In addition, there are several aspects involved in the railway system which adds to its importance. Nowadays, there is an increasing demand for train traffic although the level of congestion is already high in several parts of the rail networks. Moreover, railway transport is a matter of public concern and customer experience is a major aspect of the system. There are also concerns about energy use and environmental issues. Moreover, construction of new railway infrastructure is very expensive. Acquisition and maintenance of the rolling stock are also very costly.

Although the railway system is very complex and capital-intensive, there are huge opportunities to improve the system performance for ensuring the best usage of the available resources. The capacity can be utilized more efficiently to improve customer satisfaction and increase the revenue through effective routing and scheduling, which is the focus of this thesis.

1.1 Motivation

In a railway network, train movements are confined to the tracks and the infrastructure is shared by several trains over long distances. Although railway transportation has been around for more than two centuries, operating trains on a railway network has become more challenging than before. Large investments are required to improve the railway network infrastructure, equipment and staff to satisfy the ever-increasing demand of the traffic for both passenger and freight. However, utilisation of the railway network can be improved hugely by generating better plans for railway operations.

Meeting the ever-increasing demand for additional rail capacity is a key issue for many train companies. There are two ways of providing the additional capacity for passengers and freight users. One way is to construct new sections of track and another is through the release of capacity on the current rail network. Whereas first option is very costly, the latter is linked to train scheduling and routing which reduces the loss of capacity of the network through better scheduling and routing decisions.

Management of train operations in real-time is also very important. The issue becomes more crucial when the network is highly congested and trains are highly interconnected. A disturbance in this case can easily propagate through the network and affect several services. Train rescheduling and rerouting can help with real-time decisions in response to disruptions. The aim is to minimize delay propagation through the network with fast and effective scheduling and routing capability.

The railway industry has a major role in the UK's transport system. Network Rail is the train infrastructure manager in the UK who is the owner and operator of most of the rail infrastructure in the UK. Network rail calculates a measure called *Public Performance Measure (PPM)* to show the percentage of train which arrive at their destination on-time (NetworkRail, 2013). A train is defined on-time in the UK if it arrives at the destination within five minutes of the planned arrival time for London and South East or regional service, or within 10 minutes for long distance services. Therefore, PPM is a measure to calculate the percentage of trains which are neither cancelled nor late. So a criticism is that PPM can underestimate the importance of cancellations as a 6-minute late train and a cancelled train score the same. Network Rail announces the national PPM, which includes all train operating companies as 93% for the first period of 2013-14 from 1 April to 27 April 2013. Compared to the other countries, the measure of punctuality in the UK can be improved further by unlocking sufficient capacity to provide better services.

Another issue which can be pointed out in relation to the capacity in the UK network is about train operating companies (TOC's). After the privatisation of British Railways and railway deregulation in the UK, there are several TOC's who operate on the same part of the network. TOC's work under the supervision of the Network Rail and have different penalties for service delays according to their contracts with Network Rail. Minimizing the effect of disturbances in such a complicated situation becomes more critical.

The application of operational research methodologies, in particular scheduling, along with advances in technology, can provide better services for customers and increased revenue for the rail industry.

1.2 Research objective

This thesis aims to build on existing models for train scheduling and rescheduling problems with the goal of obtaining a generic model that includes important operational and safety constraints. Also, the model needs to present special characteristics of the UK railway network. In order to make the best usage of the existing capacity in the tactical level planning, train scheduling defines train orders and timing on track segments in order to minimize delay propagation in the network. Train rescheduling is defined in the same context, but its aim is to respond to a disruption in an operational level. Due to the complexity of the train scheduling and rescheduling problems, it is usually hard to find optimal solutions for these problems. This study intends to formulate train scheduling and rescheduling problems as modified job shop scheduling problems in order to make use of the algorithmic advances found in the literature to solve the problem. The performance of the proposed solution methods is evaluated through a case study from the UK rail network.

This thesis also addresses the train routing and rerouting problem which focuses on offering better routing decisions in order to minimize delay propagation. In addition to train routes, their order and timing on track segments are defined. Train routing can be preformed at both tactical and operational levels to suggest routes in an advance plan and in the case of a disruption, respectively. We aim to offer a generic model with main operational and safety constraints which can be customised to the UK railway network. We make use of the analogy between the train routing problem and job shop scheduling problem to formulate the problem. Algorithms are proposed based on job shop approaches to offer near optimal solutions in a short period of time.

1.3 Contribution

In recent years, many railway projects have emerged as a result of collaborations between the rail industry and academia which address a high level of details in optimization models. However, many studies in the literature simplify the network details and model the train scheduling problems such that less detail about the topology of the network is included. There is a pitfall of oversimplification and losing critical details which affects the precision of the decisions.

Modeling train scheduling and routing problems leads to large formulations. Some studies in the literature need a significant amount of preprocessing and sometimes they are not flexible with regard to a change in the network such as adding a new link. Therefore, an adaptable problem formulation can have major benefits particularly for real-time planning.

Another shortcoming in the literature is about formulating and solving train scheduling and routing problems in separate phases. Integrating scheduling and routing can have additional benefits in terms of solution quality and computational time. Moreover, the literature mostly looks at one junction at a time in the routing phase. Looking at more than one subsequent junctions can have obvious benefits on routing decisions.

There are many optimization approaches for train scheduling and routing problem which are proposed for railway networks in Europe and America. However, less work has been done in terms of optimization algorithms in the UK network.

Although there is currently an increasing trend towards realism in railway optimization problems, more work is still needed to fill the gap between theory and application.

In view of the above discussions, the main contributions of this thesis with regard to *train scheduling* can be given as below.

- A flexible formulation is proposed for train scheduling as a modified job shop problem in order to minimize delay. It looks at the detailed layout of the tracks. Schedules arising from solving this model offer precise timing and order of the trains on track sections which conforms to a microscopic planing level in terms of network topology.

- A generic mathematical programming model is suggested for train scheduling and rescheduling which formulates the main operational and safety constraints and it can represent particular characteristics of the UK rail network. Experimental results on our case study show the computational advantage of the proposed model for tackling smaller disruptions in terms of the solution quality and runtime.

- A novel algorithm using job shop scheduling concepts is proposed and tested on our case study. The efficiency of the method appears during bigger disruptions where the other approaches struggle even to offer feasible solutions in some cases.

- A novel heuristic is suggested to avoid infeasibility of the solution in order to serve as a baseline for our comparisons. It appears that simply finding a feasible solution is nontrivial as the developed FCFS algorithm frequently runs into infeasibility.

Here, we list the main contributions of the *train routing* in the following.

- The train routing and scheduling problem is formulated with a flexible model based on modified job shop problem to minimize delay propagation. It looks at the mesoscopic planning level. Hence, the model can be used for the tactical and operational planning.
- A generic mathematical programming problem which can be also adapted to the UK network is offered. It incorporates main operational and safety constraints.
- A novel algorithm based on job shop scheduling approaches has been developed. Computational results show the efficiency of the algorithm in handling different types of disruption. Results are reported for our case study in the UK network.
- A combined approach for train scheduling and routing is offered in this study which has substantial advantages over solving the problems in separate phases both in terms of solution quality and computational time.
- A real-world case study is developed for a complicated network in the UK. It provides a detailed representation of particular characteristics of the UK rail network. Development of the case study includes challenging steps for data collection, refinement and implementation with regard to both train scheduling and routing problems.

1.4 Thesis Overview

The remainder of this thesis is organized as follows. Some well-known solution methods for combinatorial optimization are introduced in Chapter 2. In Chapter 3, the terminology and concepts of train planning and control are explained. The purpose and scope of the train scheduling and routing problems are also defined in this chapter. Then, a real-world case study in the UK rail network is introduced. Chapter 4 gives an overview of the railway planning studies. We also offer a literature review of job shop scheduling in general and in railway studies. The train scheduling and rescheduling problems are addressed in Chapter 5 which contains the development of our proposed model. We adapt the shifting bottleneck solution approach for the particular job shop problems that arise in train scheduling and rescheduling. The performance of the proposed methods on a real-world case study based on London and South East area of the UK is also reported in this chapter. Chapter 6 offers the study on train routing and rerouting problems. After formulating the problems as a modified job shop problem, we suggest a modified shifting bottleneck to solve it. Then, computational experiments for the proposed methods are provided for the same case study. Finally, Chapter 7 presents some concluding remarks and suggestions for future work.

Chapter 2

General combinatorial optimization approaches

Operational Research or Operations Research (OR) was initiated during World War II in Britain. After the war, the ideas in military operations were expanded to civilian sectors and helped with advances in efficiency and productivity in industry and commerce. Management Science (MS) is another term which is interchangeable with OR. Nowadays, OR and MS are widespread in the world and they are dominant and indispensable tools in decision making and management.

In this context, optimization tries to find the best possible solution through a mathematical model. The mathematical models are the cornerstones of OR and solving a model provides a foundation to reach a decision. A mathematical optimization model consists of an objective function and some constraints. There are different techniques in the literature in order to minimize or maximize some evaluation functions.

Broadly speaking, a *Combinatorial Optimization Problem* (COP) is any optimization problem with a finite number of feasible solutions. If we have a discrete search space X and a function $f : X \mapsto \mathbb{R}$, the COP problem is to find $x^* = \underset{x \in X}{\text{argmax}} f$, where x is a vector of decision variables and f is the objective function for a maximization problem. Same argument stands for a minimization problem with relevant changes. Thus, COP can be defined more specifically as an optimization problem which has a discrete search space and an objective function of discrete variables. Combinatorial optimization is about studying effective search algorithms to solve COPs by examining the solution space effectively. In combinatorial optimization problems, we are usually concerned with finding the best selection, arrangement, sequence, etc with regard to a specific objective function (Papadimitriou and Steiglitz, 1982).

Some of the well-known COPs can be listed as travelling salesman problem (TSP), minimum spanning tree problem, shortest path problem, facility location, knapsack problem, assignment problem, vehicle routing, matching, set covering, network flow and machine scheduling. There are different methods to solve COPs such as integer programming, branch-and-bound, dynamic programming, local search and metaheuristics which are discussed in the following sections.

2.1 Problem Complexity

Complexity theory is developed by logicians and computer scientists based on a mathematical framework. Complexity theory addresses the computational complexity of the algorithms with regard to required resources such as time and storage to solve the problem. Assume that a computational problem can be presented as a function h which maps an input x in a given domain to $h(x)$ in a given range. We are interested in an algorithm $h(x)$ for an input x to solve the computational problem.

Measuring the performance of an algorithm is one of the main issues of complexity theory. As it is commonly assumed in any text on computational complexity, in order to define a computational step, a *Turing machine* is used as a standard mathematical model of a problem. If we define the input length $|x|$ as the length of the an encoding of x, the efficiency of an algorithm for a given problem is measured by an upper bound $T(n)$ on the number of steps that the algorithm takes on an input x with $|x| = n$ (Brucker, 2007). As it is hard to calculate the precise form of T, the asymptotic order is used. If there exist constants $c > 0$ and a nonnegative integer n_0 such that $T(n) \leq cg(n)$ for all integers $n \geq n_0$, we say that $T(n) \in O(g(n))$.

A search problem can be defined as a problem with a solution which can be checked effectively for correctness. It means if there is an instance Ins with a proposed solution Sol, we can find a *Polynomial* algorithm to determine whether the solution Sol is an actual solution for instance Ins.

According to complexity theory, problems can be classified to main classes of P and NP. NP stands for *Nondeterministic Polynomial* which determines that there is an idealised algorithm to find and verify a solution and the algorithm can guess correctly at each stage. As the problem is nondeterministic, there are different choices for computation. Actually, the computation becomes like a search tree. Therefore, not all paths result in an actual solution. If a correct choice is made in each stage, we can reach a correct solution in polynomial time. NP is the so-called class of all search problems. According to the discussion above, if in all cases we can find a path deterministically to a solution, the problem is in class P which denotes polynomial. Thus, P is the class of the problems that are solved in *Polynomial* time.

In complexity theory, there is a distinction between *optimization* problems and *decision* problems. A problem is a *decision problem* when the output range is $\{yes, no\}$; Therefore, the decision problems are also called as yes-no problems (Pinedo, 2008). More formal definitions of classes of P and NP are given in the following. If there is a polynomial p such that $T(|x|) \in O(p|x|)$ for all inputs x of a problem, the problem is *polynomially* solvable. The class of all decision problems which are polynomially solvable is shown by P (Brucker, 2007). A problem is called *pseudopolynomially* solvable if there is a pseudopolynomial algorithm which solves the problem (Brucker, 2007).

For a problem to be in class NP, it is not required that every instance can be solved by a polynomial algorithm. We simply require that ,if $|x|$ is a “yes” input in the decision problem in class NP, it has a certificate y such that $|y|$ is bounded by a polynomial in $|x|$ and a polynomial time algorithm exists which verifies that y is a valid certificate for x (Papadimitriou and Steiglitz, 1982; Brucker, 2007).

It should be noticed that P is a subset of NP. If we have a decision problem solvable in polynomial time which calculates the answer $h(x) \in \{yes, no\}$ for each input x, the answer $h(x)$ can be used as a certificate and this certificate can be verified by the algorithm. Therefore, we can conclude that $P \subseteq NP$ which means that P is also in NP (Papadimitriou and Steiglitz, 1982; Brucker, 2007). It is generally speculated that $P \neq NP$, but there is no proof known for it. It is one of the major open problems of the modern mathematics and a $\$1,000,000$ prize is offered to solve it.

Another fundamental concept in complexity theory is the *problem reduction.* It is often the case that one combinatorial problem is a special case of another problem or more general than the other one. When an algorithm works well for one, it can often work well for the other one with some modifications. If we have two decision problems Q and Q', it is said that Q reduces to Q', denoted as $Q \propto Q'$, if there is a polynomial time computable function g that convert input of Q to Q' such that x is a yes input of Q if and only if $g(x)$ is a yes input of Q' (Papadimitriou and Steiglitz, 1982; Brucker, 2007).

In a renowned theorem, Cook (1971) introduced a proof for a class of hardest search problems which are called *NP-complete.* NP-complete problems are a very important subset of class NP and if all other search problems are reduced to it, the search problem is NP-complete. More formally, we call a decision problem Q', NP-complete if $Q' \in NP$ and all the other decision problems $Q \in NP$, transform to our problem, that is $Q \propto Q'$. To highlight the importance of the NP-complete problems in complexity theory, it should be mentioned that if any NP-complete problem Q can be solved in polynomial time, all problems in NP are also solved in polynomial time and we can conclude $P = NP$. This is one of the most important theoretical questions in mathematical logic and combinatorial optimization.

A problem is *NP-hard* if the entire class of NP problems are polynomially reduced to it (Garey and Johnson, 1979). We call an optimization problem NP-hard if the corresponding decision problem is NP-complete (Brucker, 2007). It should be noted that when a scheduling problem is solved, we deal with optimization problems instead of decision problems.

Finding the solution to the NP-hard problems in polynomial time is an issue. So we may need to settle for approximation methods or implement search methods to find the best possible solution. Also, it becomes very demanding in terms of time and computational effort to find an optimal or near-optimal solution. This is specially the case when real-world optimizations are involved. Therefore, it is important that the search method can find a sufficiently good solution in many applications.

2.2 Mathematical programming

Nowadays with the advances in technology and faster computers, solution based on mathematical programming for practical problems are more prevalent. Mathematical programming has a wide range of applications in different areas such as scheduling and resource allocation.

The mathematical model includes an objective function, decision variables and some constraints and the aim is to find values of the decision variables in order to minimize or maximize an objective function among all the values of the decision variables that satisfy the constraints. In the following, three classes of mathematical programming are presented as linear programming in Subsection 2.2.1, and integer programming and mixed integer linear programming in Subsection 2.2.2.

2.2.1 Linear programming

Linear programming is a mathematical programming approach and it is one of the widely used OR techniques. In a linear program, the objective function and constraints are strictly linear. More precisely, an instance of *Linear Program* (LP) can be defined as a maximization (or minimization) of a linear function over a polyhedron:

$$\text{Maximize (Minimize)} \quad CX$$

$$\text{subject to}$$

$$AX \leq b$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and the variables $X \in \mathbb{R}^n$. It should be noted that the feasible region in an LP is a convex space and the objective function is a convex function. Thus, according to the optimization theory, the optimal solution can be found at an extreme point of the feasible region as long as the variables are of any real non-negative values and probably bounded above (Burke and Kendall, 2005).

Duality is an important concept of the linear programming. For a maximization problem of the form

$$\begin{aligned} \text{Maximize} \quad & CX \\ \text{subject to} \quad & \\ & AX \leq b \\ & X \geq 0 \end{aligned}$$

which is called the *primal* problem, where all constraints are $\leq$ constraints and all variables are non-negative, there is a *dual* problem defined as

$$\begin{aligned} \text{Maximize} \quad & b^T Y \\ \text{subject to} \quad & \\ & A^T Y \leq C^T \\ & Y \geq 0 \end{aligned}$$

The optimal solution value of the primal and dual problems are the same. If both primal and dual problems are formulated in standard form with $s_1, s_2, ..., s_m$ and $e_1, e_2, ..., e_n$ as the slack variables of primal and dual problems respectively, according to the *theorem of complementary slackness*, if $X = x_1, x_2, ..., x_n$ is a feasible primal solution and $Y = y_1, y_2 ..., y_n$ is a feasible dual solution, X is primal optimal and Y dual optimal if and only if $s_i y_i = 0, \forall i = 1, 2, ..., m$ and $e_j x_j = 0, \forall j = 1, 2, ..., n$. This important property is quite useful in developing LP-based solution approaches.

In general, solution approaches for LP problems can be categorized in two classes of simplex-based and interior point methods. These techniques are developed based on the concept of searching the extreme points of the feasible region of the primal or dual problems until the optimality conditions are satisfied. Variants of the simplex-based methods have emerged after the seminal work of Dantzig (1951) and used in optimization software packages. The only disadvantage of these methods is that they have poor worst-case time performance.

Due to this deficiency, the interior point methods are developed where they search for solutions through a path in the interior points of the feasible region and the optimal solution is found when they reach the boundary. The study of interior point methods trace back to Karmarkar's algorithm by Karmarkar (1984) which shows that it is possible to create an algorithm with polynomial complexity for linear programming problems. Although the ellipsoid method by Kachiyan (1979) was already developed as the first polynomial time algorithm, it was too slow to be interesting in terms of practical use.

Linear programming can model real situations and find optimal solutions to very huge problems with thousands of variables and constraints, and millions of variables and constraints with the new computers. However, there is often a level of approximation involved in the modeling phase which helps with solving the model.

Compared to the other approaches, linear programming is quite a strong tool in terms of the modeling and solvability (Taha, 2002). Linear programming is the foundation of other analytical OR models such as integer, nonlinear and stochastic programming. Linear programming has a wide range of practical interest in different disciplines such as industry, transportation, health care and military.

2.2.2 Integer programming and mixed integer linear programming

Integer programming is another mathematical programming approach where variables are assumed to be integer. In other words, integer programming is similar to linear programming with more restrictions. At first, this requirement may seem very straightforward, but it can extensively add to the difficulty of the problem to solve. Wolsey and Nemhauser (1998) and Schrijver (1998) are excellent references addressing theoretical concepts of integer programming.

However, integer programming is a powerful optimization tool to model many problems. A key issue in modeling problems as an integer programming is how the formulation is done. Similar formulations can lead to different computational results. While one model may lead to a fast or even optimal solution, the other one may take a long time to be solved. There are some key issues to model and solve better integer programming. Generally, finding formulations with strong linear relaxations, considering more variables and constraints and avoiding symmetry results in more successful integer programming models (Burke and Kendall, 2005).

By dropping the integrality constraints of an *integer program* (IP), an associated linear program is created which is called the *linear relaxation* of the problem. The linear relaxation of the IP has several properties in relation to the associated IP. These properties can be quite useful to solve the IP as it provides a bound on the optimal solution of the IP. The optimal value of the linear relaxation is less (greater) than or equal to the optimal value of the IP if IP is a minimization (maximization) problem. If the linear relaxation is infeasible, IP is also infeasible. If all the variables in an optimal solution of a linear relaxation are integer, that solution is optimal for the IP.

One of the most common approaches to solve integer programming problems is *branch-and-bound* which is discussed in detail in Subsection 2.3.1. It is not always the case that branch-and-bound can solve the problem quickly and there are a number of techniques to improve the computation time. One of the techniques is to find a good formulation with strong relaxation. In an empirical viewpoint, a good formulation is the one for which branch-and-bound or another integer programming technique can find the optimal solution quickly (Burke and Kendall, 2005). The most important aspect for a successful formulation is to find a formulations with a linear relaxation which does not differ a lot from its associated IP.

As noted before, a strong linear relaxation improves the strength of the formulation. By adding certain types of constraints, we can cut off the optimal solution x^* which can generate improved relaxations around the linear relaxation optimum.

For instance, consider an IP which has constraints of form

$$\sum_j a_j x_j \leq b$$

where all x_j are required to be binary and a_j are nonnegative. Consider a subset C of the variables such that

$$\sum_{j \in C} a_j > b.$$

We can deduce the following *cover constraint*

$$\sum_{j \in C} x_j \leq |C| - 1$$

which is not violated by any feasible integer solution and cuts off a fractional solutions if C is minimal. Although we like to add such constraints to our model, the number of these constraints can be exponential in n which makes it impractical to include all of them. We select and generate only the constraints which are needed so that we can benefit from the strength of the improved relaxation without huge number of constraints.

One of the most well-known constraints that can be applied to any IP is the Gomory Cut (Gomory, 1958). Cutting planes offer the possibility of improving relaxation values and have been exploited to make branch-and-bound more effective. Crowder et al. (1983) showed that cutting planes can greatly strengthen relaxations and improve the branch-and-bound method.

Similar to the constraints, we can improve the IP formulation by adding many variables. Just as we can handle an exponential number of constraints by generating them as needed, we can create variables as they are needed through *variable generation*. If we consider a formulation with a huge number of variables, we can start solving this problem by a smaller number of variables at first. Thus, we solve the linear relaxation to get the dual values associated with each constraints. Then, we can determine if adding one or more variables to the formulation can result in a better solution. If the inclusion of the variable(s) does not improve the solution, we solve the linear relaxation and repeat the same process.

Generally, models with a large number of variables provide tight formulations. It would be very time consuming to check all variables if the number of variables is high. It is necessary to develop a variable generation routine to find improving variables. Using appropriate branching rules with variable generation for solving the subproblem at each node in a branch-and-bound method is called *branch and price*. Barnhart et al. (1998) offers an excellent survey of this approach.

Another cause for the failure of the integer programming models is symmetry and branch-and-bound can become very inefficient for solving models with symmetries. Depending on the problem, there are different ways to remove the symmetries in a formulation. We can break the symmetry through adding constraints, fixing variables or modifying the formulation.

A *pure integer program* is a formulation in which all variables are integer, otherwise the problem is *mixed integer program* (MIP) or Mixed Integer Linear Program (MILP) (Papadimitriou and Steiglitz, 1982). A formulation is called a MIP or MILP problem when it involves some variables which are constrained to be integer and some which are not integer.

According to Wolsey and Nemhauser (1998), a MILP problem can be written generically as

$$\begin{aligned} \text{Maximize} \quad & CX + HY \\ \text{subject to} \quad & \\ & AX + GY \leq b \end{aligned}$$

where $X \in \mathbb{Z}_+^n$ and $Y \in \mathbb{R}_+^p$. C, H, A, G and b are matrices which has integral coefficients with dimensions mentioned in the following. C is $1 \times n$, H is $1 \times p$, A is $m \times n$, G is $m \times p$ and b is $m \times 1$, where p and n are any nonnegative integers with $p + n \geq 1$ and m is any positive integer.

According to the above formulation, pure integer programming is a special case of MILP problem when $p = 0$ and linear programming is the special case of MILP problem when $n = 0$. An instance of a linear or pure integer program is also an instance of a MILP. Therefore, an algorithm which can solve all instances of MILP, can solve all instances of special cases of pure integer and linear programming.

Thus, it can be obviously concluded that MIP is at least as hard as pure integer and linear programming (Wolsey and Nemhauser, 1998).

Although most of the books on operations research and combinatorial optimization deal with MILP models, there are a few references like Plastria (2002), Williams (1990) and Paschos (2010a) which cover a systematic approach for MILP modeling. Some basic MILP formulations for well-known combinatorial optimization can be listed as the knapsack problem, the set covering/set partitioning problem, the minimum cost flow problem, the maximum flow problem, the transportation problem, the assignment problem and the shortest path problem.

There are some general techniques to enable us to develop linear models when there are nonlinear expressions or there are logic conditions between real/integer variables and/or between binary variables. We refer the reader to Paschos (2010a) for more details about modeling tricks to deal with the following situations.

- Min-max, max-min, min-abs models
- logic conditions such as one-to many conditions, fixed charge constraints, big-M constraints, either-or constraints

For an insightful study on how to define tight representations of logical conditions by using linear constraints, we refer to Yan and Hooker (1999).

According to Paschos (2010a), a MILP model has a good formulation if its linear programming relaxation is sufficiently tight. This means that optimal solution of the linear programming relaxation is sufficiently close to the optimal solution of the MILP. It is also advised that in order to develop a good MILP model, all structural properties of the problem should be taken into account.

2.3 Exact algorithms

Exact algorithms are methods which guarantee to compute optimal solutions. Integer programming has already been described in Section 2.2. We discuss two main classical exact algorithms, branch-and-bound in Subsection 2.3.1 and *dynamic programming* Subsection 2.3.2.

2.3.1 Branch and bound

Branch-and-bound algorithm, which was proposed first by Land and Doig (1960), is an algorithm to find optimal solution for a problem. Branch and bound algorithm is based on a systematic enumeration of all feasible points of a combinatorial optimization problem which solution has a tree structure (Papadimitriou and Steiglitz, 1982). Compared to a complete enumeration, the branch-and-bound search is more effective as it discards unattractive alternatives by evaluating them with local information.

Branch-and-bound tries to construct a proof that a solution is optimal through successive partitioning of the solution space. We assume that the combinatorial problem to be solved is a minimization problem. Two components of the branch-and-bound algorithm in a general context to develop a tree are as below.

- *Branching*: A set of solutions shown by a node is partitioned into mutually exclusive subsets, where each subset is shown as a child of the parent node.
- *Lower bounding*: An algorithm is implemented for calculating a lower bound for each solution of a given subset.

Assume X to be a set of feasible solutions and $f : X \to \mathbb{R}$ to be a criterion function. Define X^* as the set of optimal solutions such that

$$X^* = \{x^* | x^* \in X, f(x^*) = \min\{f(x) | x \in X\}\} \tag{2.1}$$

Lenstra (1977) suggests a branch-and-bound procedure to find an element of X^* in the following.

- The best solution x^* found so far, during the execution of the procedure, provides an upper bound $f(x^*)$ on the optimal solution value.
- Let $Y \subset X$. A branching rule b is defined which associates a family $b(Y)$ of subsets to Y such that $\bigcup\limits_{Y' \in b(Y)} Y' \cap X^* = Y \cap X^*$, where subsets Y' are the children of the parent subset Y. The branching rule is defined on a class X with $x \in X$ and $b(Y) \subset X$ for any $Y \in X$.

- A bounding $lb : X \rightarrow \mathbb{R}$ is to offer a lower bound $lb(Y) \leq f(x)$ for all $x \in Y \in X$. Eliminate Y if $lb(Y) \geq f(x^*)$.
- The logical variable $\xi : X \rightarrow true, false$ determines if during the evaluation of Y (for instance while calculating $lb(Y)$) a feasible solution $x(Y)$ is produced which has to be examined. x^* is improved if $f(x^*) > f(x(Y))$.
- A search strategy selects a subsets from the generated subsets which have not been neither eliminated nor resulted in branching so far.

In summary, the branch-and-bound prunes some branches of the tree which seem to result in sub-optimal results. The evaluation of a partial solution is through the use of upper and lower bounds. A problem is repeatedly partitioned into a set of smaller subproblems where the nodes at each level correspond to partial solutions. In a minimization problem, search starts from a root node and when the lower bound of a node is worse than the upper bound which is found so far, the node is pruned.

There are different choices in how to implement a branch-and-bound algorithm for a given problem. Lenstra (1977) determines three general search procedures given as below

- Frontier search: A breadth-first search is implemented where a subset with minimal lower bound is chosen for evaluation.
- Newest active node search: A depth-first search is employed where children of a parent subset are evaluated in an arbitrary order.
- Restricted flooding: A depth-first search is carried out when the children are selected in nondecreasing order of lower bounds.

In general, many problem-dependent adjustments may be included in a branch-and-bound algorithm. For instance, Y can be eliminated during the calculation of $lb(Y)$ or by using some elimination criteria based on dominance rules or feasibility considerations (Lenstra, 1977).

With regard to many possibilities to organize the branching, Brucker (2007) describes firstly the possible trade-offs between choosing a relatively tight lower bound with higher computation time, and calculating not so tight lower bounds which requires less computation time. Secondly, he mentions about the same trade-off which exists in selecting the dominance relation. Thirdly, he explains about the choice of which node to branch from at each branching step. Brucker (2007) and Papadimitriou and Steiglitz (1982) determine usual alternatives for branching at each node as least-lower-bound-next, last-in-first-out or first-in-first-out.

Brucker (2007) and Papadimitriou and Steiglitz (1982) mention that most of the time the branch-and-bound algorithm is stopped before optimality is reached due to the design or necessity. Therefore, we end up with a complete solution with cost U and the lowest lower bound LB of any live node providing a lower bound on the optimal cost. Thus, we are in the ratio of $(U - LB)/LB$ of optimal solution. Let OPT determine the optimal solution, then $(U - OPT)/OPT \leq (U - LB)/LB$, that is $(U - LB)/LB$ is an upper bound for the performance ratio of the heuristic that is obtained by terminating the branch-and-bound algorithm before it reaches the optimal solution.

It should be clear now that the branch-and-bound algorithm is not a specific algorithm, but a wide class and its performance hugely depends on the branching strategies and the quality of the bounds. Design of a branch-and-bound algorithm depends on the problem and the data. In order to develop a good branch-and-bound, some computational experiments are needed. The main drawback of branch-and-bound algorithm is the long computational time and in a large size problem, the computation time may become prohibitive. Therefore, it is necessary to develop heuristics which lead to reasonable results in a reasonably short time (Pinedo, 2008). Lawler (1966) offer a good review of branch-and-bound studies up to 1966.

2.3.2 Dynamic programming

Dynamic programming is similar to branch-and-bound in the sense that it is an implicit enumeration of all the feasible points of a problem. Dynamic programming

(DP) was firstly developed by Richard Bellman in the 1940's (Bellman, 2003). DP is a technique where the solution process is decomposed into solving smaller subproblems. This break down makes it easier to solve the smaller problems. Actually DP looks at all the possibilities at each stage of the solution. In each stage, we need to decide how to get to the next stage through a number of states.

Defining stages and states are the main tasks to solve a problem by DP. Although determining stages and states depends on the problem, but generally stages are associated with time periods such as planning horizons and years, and states correspond to the attributes of an entity like the amount of production or capacity of a resource.

DP solves a problem recursively and the optimal solution of a subproblem is the input for the next subproblem. Thus, a recursive function needs to be formulated to connect the current stage with its states to the previous stages and its states.

Paschos (2010a) introduces a formal framework of the dynamic programming in the following. Assume that the combinatorial optimization problem we need to solve, can be broken down into $n+1$ periods $0, ..., n$ where n is an integer $n \in \mathbb{N}$. Let S_i, defined for each period i, be a set of all states at the end of a period i. Suppose $S = \bigcup_{i=0}^{n} S_i$ denotes the set of all states. Further assumptions are defined as below.

- $S_i \cap S_j = \emptyset$ if $i \neq j$. It is not a restrictive assumption because if this is not the case, the states can be redefined by adding the index of the corresponding period.
- There is a unique initial state s_0 and a unique final state s_n. It is not a restrictive assumption as we can add dummy initial and/or a final state.

DP aims to take a decision at each period such that the decisions are optimal for certain criteria at the end of the process. Let X denote a set of all possible decisions. Considering that not all decisions can be taken in each state, there is a corresponding subset (which is included in set X) representing possible decisions in each state. Therefore, we define a set $A \subset S \times X : (s, x) \in A$ to show that the decision x can be taken in state s.

A decision taken at the period i gets us from one state to another one. A transition function $t : A \to S$ is considered, where $t(s, x)$ denotes the state that we are in after taking decision x in state s. This means a decision $(s, x) \in A$ from a state $s \in S_i$ at the end of period i takes us to a state $t(s, x) \in S_{i+1}$ at the end of period $i + 1$.

If final stage of the problem is analyzed in the first stage of the approach, the algorithm represents a backward DP. Similarly, a forward DP solves the the first stages of the problem in the first stage of the approach.

The main characteristic of dynamic programming is called *principle of optimality* and it means that for a given state, an optimal decision for the remaining stages should not depend on previously chosen stages. Papadimitriou and Steiglitz (1982) offer a formal definition of the principle of optimality in the following. Suppose we need to make a sequence of n decisions D_1, D_2,...,D_n, in order to solve a combinatorial optimization problem. If the sequence of decisions is optimal, the last k decisions D_{n-k+1}, D_{n-k+2},...,D_n should be optimal. This means that the completion of an optimal sequence of decisions should be optimal.

It can be seen from the discussions above, DP is a general idea that can be applied to different problems in different ways as there are various ways of breaking the problem into stages and formulating a recurrence relation. Some of the NP-hard problems in the literature have been solved pseudopolynomially using DP (Brucker, 2007). Denardo (1982), Ross (1983) and Bertsekas (1987) are some of the books that have been written on dynamic programming.

This approach can provide satisfactory results if the problem is not too large. The main disadvantage of the DP is due to the fact that the number of subproblems depends not only on stages, but also on the states, which may result in running out of memory.

2.4 Heuristics

A heuristic is a method to obtain high quality solutions at a reasonable computation cost which does not guarantee the optimality and possibly feasibility

(Rayward-Smith et al., 1996). According to (Brucker, 2007), we can consider any approach without formal guarantee of performance as a heuristic which is useful in practice when there is no better method. They are the method of choice for NP-hard problems as they provide robust solution approaches for problems of practical interest in a controllable time.

There are two main classes of *constructive* and *perturbative* heuristics. Whereas constructive heuristics build a solution from scratch, perturbative heuristics start with an initial complete solution and afterwards try to iteratively improve it. Different variations of the heuristics include local search, simulated annealing, tabu search and genetic algorithm (Burke and Kendall, 2005). In the following sections, we will address some of the mentioned heuristics.

2.5 Local search heuristics

In general, local search methods provide only feasible solutions which are not guaranteed to be optimal (Brucker, 2007). So the result and performance of a local search algorithm cannot be assured. However, local search method is widely used to solve many NP-hard optimization problems approximately (Paschos, 2010b). Local search methods can solve a wide range of hard combinatorial problems. Local search can be a very effective heuristic for computationally complex problems and it is often in fact, the best available Papadimitriou and Steiglitz (1982).

Brucker (2007) and Papadimitriou and Steiglitz (1982) summarize the general local search algorithm in the following. If an instance (F, c) of an optimization problem is given, where F is the feasible set and c is the cost mapping, a neighborhood $N : F \to 2^F$ is chosen such that it is searched at point $t \in F$ for improvement according to the following subroutine

$$improve(t) = \begin{cases} \text{any } s \in N(t) \text{ with } c(s) \leq c(t), & \text{if such an } s \text{ exists} \\ 'no', & \text{otherwise} \end{cases} \tag{2.2}$$

A local search starts from an *initial solution* $t \in F$ and explores the solution space in order to improve the solution by searching the *neighborhood* of the current solution for a better solution. If the new solution is accepted, we move to that solution by applying some *transformations* by using subroutine *improve*(t). Then, we need to consider the neighbors of this solution and the process is repeated until no better solution can be found. Thus, the algorithm stops when it reaches a solution which is not strictly better than itself. Such a solution is called *local optima* Paschos (2010b).

Papadimitriou and Steiglitz (1982) represent the local search algorithm framework in the following.

Algorithm 1 Local Search

```
$t :=$ some initial starting point in $F$;
**while** $improve(t) \neq 'no'$ **do**
  $t := improve(t)$;
**end while**
**return** $t$
```

To obtain a an initial solution, it is sometimes practical to run local search algorithm from different starting points and select the best result. The definition of the neighborhood and the moves (transformations) are dependent on the problem structure. According to Papadimitriou and Steiglitz (1982), very little theory is available as a guide for defining the neighborhood and a search method for it. Therefore, choices are mostly based on intuition. A larger neighborhood can provide better local optima, but it will take longer computation time to search it. Designing an effective local search is an art and questions about it are usually answered empirically.

The simplest form of the local search is the *descent* which performs a series of moves as discussed above. The main drawback of descent local search is that it can be trapped at a local optimum.

Iterated descent local search is capable of finding near-optimal solutions for a wide range of complex combinatorial problems and it is more applicable in comparison with descent search. Iterated descent search has a perturbation operator in addition to the descent search which helps the solution not to get stuck in a local optimum. This operator which is called a *kick move* changes the local optimum to a new starting point for the descent search.

The complexity of a problem which includes finding a locally optimal solution is an open problem which means that we do not know if it is possible to find a solution in polynomial time. Johnson et al. (1988) introduce the complexity class *Polynomial-time Local Search (PLS)*. Paschos (2010b) have mentioned that many local search problems are proved to be complete for the class PLS with regard to an appropriate reduction. Therefore, the class PLS represents the complexity of local search problems in a similar way to class NP which shows the complexity of combinatorial optimization problems.

Aarts et al. (1994) offer a detailed computational study for local search algorithms applied on job shop scheduling problems. Books by Aarts and Lenstra (1997) and Hoos and Stutzle (2005) offer a general overview of local search. Local search is the basis for many metaheuristics such as Simulated Annealing and Tabu Search (Paschos, 2010b). In addition, it can be used jointly with other metaheuristics like Genetic Algorithm. In the next section, metaheuristic methods such as Simulated Annealing, Tabu Search and Genetic Algorithm are discussed.

2.6 Metaheuristics

Metaheuristics is a term employed for a special class of the heuristic methods which are beyond the heuristics in terms of the strategy and procedure and they can guide and modify underlying heuristics. They are also called modern heuristics and they have received a lot of interest among search methodologies. Metaheuristics are more complicated than heuristics and some of the well-known metaheuristics in the literature are simulated annealing, genetic algorithm and tabu search.

2.6.1 Simulated annealing

Simulated annealing (SA) is a stochastic heuristic which is developed based on the analogy between annealing process and solving COPs. It is a method which tries to avoid being trapped in a local minimum by allowing moves to inferior solutions with the help of a randomized scheme (Rayward-Smith et al., 1996).

Annealing is a thermal process in physics to reach an stable state with low level of energy for solid crystallisation which was introduced by Metropolis et al. (1953). SA is formally proposed by Kirkpatrick et al. (1983) who showed the analogy between the Metropolis algorithm and solving an optimization problem.

In order to accomplish the annealing process, the material is heated to a temperature that all its molecules can move freely. Then it is cooled so slowly that material freezes to a completely ordered crystal. Thus, the system is in the state of minimum energy. The same logic is used to solve optimization problems.

Let c be the cost function defined on the solution space X. Assume $N(x)$ to be a neighborhood function for $x \in X$. Glover and Kochenberger (2003) state the common steps during the conventional SA algorithm for a minimization problem as follows. SA starts with an initial solution $x \in X$ and a neighboring solution $x' \in N(x)$ is then generated which leads to a change in the cost function which is denoted by $c(x') - c(x)$. If the move decreases the cost and therefore improves the solution (improving step), the move is accepted and the decision variables are modified accordingly. If the move increases the cost and thus makes the solution worse (non-improving step), a random number R is generated by a uniform distribution on the interval $[0, 1]$. The random number R is compared with the probability of accepting the move and the move to x' is accepted if $\exp(-(c(x') - c(x))/t_k) < R$, where t_k is a control parameter at iteration k such that $t_k > 0$ for all k and $\lim_{k\to+\infty} t_k = 0$. If the move is accepted, variable values are updated; otherwise, it is rejected. This procedure is repeated until an equilibrium state is obtained. Now the temperature is reduced and the same process is repeated until a stopping criteria is met. Eglese (1990) outlines the SA algorithm in Algorithm 2.

We can conclude that the SA algorithm has $M_0 + M_1 + ... + M_k$ total iterations, where k is associated with the value of t_k at which the stopping criteria is met. Also, if $M_k = 1$ for all k, the temperature changes in each iteration.

van Laarhoven and Aarts (1987) determine in their study how to define function repetition schedule M_k and the stopping criteria for practical applications. A brief review and discussion of the theory is given in (Dowsland, 1993).

Algorithm 2 Simulated Annealing

```
Choose an initial solution $x \in X$
Choose the temperature change counter $k = 0$
Choose a temperature cooling schedule, $t_k$
Choose an initial temperature $T = t_0 \geq 0$
Choose a repetition schedule $M_k$ which determines the number of iterations at
each temperature $t_k$
repeat
  Set repetition counter $m = 0$
  repeat
    Generate a solution $x' \in N(x)$
    if $c(x') - c(x) \leq 0$ then
      $w := w'$
    end if
    if $c(x') - c(x) > 0$ then
      $w := w'$ with probability $\exp(-(c(x') - c(x))/t_k)$
      $m := m + 1$
    end if
  until $m = M_k$
  $k := k + 1$
until Stopping criterion is met
```

A principal characteristic of SA should be pointed out here. The moves which decrease the cost or improve the cost function are called downhill moves whereas the moves resulting to an increase in the cost or a worse cost function are named as uphill moves. As it is indicated before, SA algorithm not only permits downhill moves but also allows uphill moves according to a probability function calculation. Therefore, it is unlikely that SA algorithm becomes trapped in a local optimum. It can be concluded that SA is an appropriate technique to solve optimization problems with a hidden optimal solution among many local optimum points.

2.6.2 Tabu Search

Tabu search (TS) is a deterministic search procedure introduced by Glover (1989) for solving COPs. TS is designed to guide the neighborhood search to escape the trap of local optimality. TS is an extension of classical local search and the basic idea of TS is to continue the local search when it reaches a local optimum by allowing non-improving moves (Glover and Kochenberger, 2003).

It uses a flexible short-term memory which allows the thorough exploitation of the search information and prevents reversing the current moves. In addition, it uses a longer-term memory to allow interesting moves that direct the search to diversify the search towards less explored areas. Generally, tabu search has strategic constraining and freeing conditions which are substantiated in form of a *tabu list* and *aspiration criteria* (Glover and Kochenberger, 2003).

One of the distinctive elements of TS is to create a subset which is called tabu list that contains elements which are called *tabu moves*. These moves contain historical information of the search process and they are used to create the tabu list. A move becomes a member of the tabu list if it has been made in the recent past and proved either to be unproductive by a historical list of moves or by a set of tabu conditions (e.g. constraints that need to be satisfied).

Aspiration criteria is a tool to cancel some tabus in order to permit attractive moves which have been prohibited by the tabu list. The simple and most commonly used form of aspiration criteria is to allow a tabu move if it leads to a solution which is better than the best solution currently found (Rayward-Smith et al., 1996).

Let x be the current solution and x^* be the best-known solution. Assume $c(x)$ to be the cost function of x, and $N(x)$ to be the neighborhood of x. We define $\bar{N}(x)$ as the admissible subset of $N(x)$ that is not tabu or satisfies the aspiration criterion. A general framework of the TS algorithm is offered by (Brucker, 2007) in Algorithm 3.

Algorithm 3 Tabu Search

```
Choose an initial solution $x \in X$
$best := c(x)$
$x^* := x$
$TabuList = \emptyset$
**repeat**
    Generate a solution $\bar{x} \in \bar{N}(x)$
    Update the tabu list
    $x := \bar{x}$
    **if** $c(x) < best$ **then**
        $x^* := x$ and $best := c(x)$
    **end if**
**until** Stopping criterion is met
```

Various stopping criteria can be employed such as stopping after a fixed number of iterations, or after some number of iterations without an improvement in the objective function value, or when objective function reaches a pre-defined threshold value. In addition, there are different methods to update the tabu list and to generate a solution $\bar{x} \in \bar{N}(x)$. Therefore, the tabu list and aspiration criteria can constrain and free the search, and make tabu search more sophisticated than a descent algorithm.

Tabu search has been applied on various problems. We refer the reader to Nowicki and Smutnicki (1996) and DellAmico and Trubian (1993) for application of tabu search to the job shop scheduling problem.

Simple TS can sometimes solve difficult problems successfully, but often additional elements are required in order to make the search strategy more effective. Tabu search can use additional memory functions with varying time spans for *intensifying* and *diversifying* the search. In each iteration of the algorithm, we choose the best move defined in terms of the objective function. Finally, it should be noted that tabu search is more aggressive compared to the gradual descent of simulated annealing.

2.6.3 Genetic algorithm

Genetic algorithm (GA) is a stochastic metaheuristic which is developed based on the natural evolution concept. GA is first developed by Holland (1975). The algorithm starts with a set of solutions which is called population. A population is composed of individuals represented by chromosomes. In order to get better solutions, the current population is exploited to create a new generation including survivors from the previous generation and the children or offsprings. To form a new generation, individuals are evaluated and selected according to their fitness (Rayward-Smith et al., 1996).

Therefore, GA requires a representation of the solution which is usually consisted of binary strings of 0 and 1. Moreover, a fitness function is needed to evaluate the quality of the solutions. Fitness function is usually defined based on the problem. After the problem is represented and the relevant fitness function is developed, GA initializes the population and iteratively selects suitable individuals and improves them through GA operators called *mutation* and *crossover*.

Initialization is usually done by randomly generating initial solutions, or generating solutions in sequence in a way that diversity is maximized, or generating high quality solutions by another heuristic. Selection is carried out based on the filtering solutions with regard to the fitness function. There are certain selection methods to rate the fitness of each solution such as *Roulette Wheel* selection, *Rank* selection, *Elitism*, etc.

Crossover is a GA operator that takes certain number of parents and combines their genes in order to create a new offspring. There are different ways to do a crossover. the simplest form is to randomly choose a crossover point and copy the genes before and after this point from the parents' chromosomes. *Mutation* is another GA operator which is performed after crossover. Mutation modifies an individual by changing their genes. In a binary coding, we can simply change a few random genes from 0 to 1 and 1 to 0. The whole process is repeated until a termination condition is satisfied.

(Glover and Kochenberger, 2003) offer a simple template for a genetic algorithm in Algorithm 4.

Algorithm 4 Genetic algorithm

```
Choose an initial population of chromosomes
while termination condition is not satisfied do
  repeat
    if crossover condition is satisfied then
      Select parent chromosomes
      Choose crossover parameters
      Perform crossover
    end if
    if mutation condition is satisfied then
      Select parent chromosomes
      Choose mutation points
      Perform mutation
    end if
    Evaluate fitness of offspring
  until sufficient offspring is created
  Select new population
end while
```

According to (Rayward-Smith et al., 1996), GA can be efficiently applied on a problem if certain criteria are satisfied. Firstly, an appropriate fitness function should be developed to evaluate the the quality of the conceivable solution. The fitness function should provide a fine-grained set of fitness value to enable us to choose the partially correct solutions compared to less desirable ones . Another criterion for efficient search is that a suitable encoding of the solution should be available. If encoding is such that the valuable portion of the string is always or frequently lost, the progress of the GA can be severely hampered.

Having been applied in a wide range of problems, genetic algorithm has proved its value. Goldberg (1989) and Davis (1991) are the standard texts on the operation of the algorithm. We also refer the reader to Fogarty et al. (1995), Reeves (1995) and Cartwright and Cattell (1995) for application of the GA on different problems.

Chapter 3

Problem definition

Railway system consists of the four main components: (1) infrastructure, (2) rolling stock, (3) schedule and (4) the operating rules and safety regulations, which are linked together to provide the railway services to passengers and to perform freight transportation. Infrastructure includes tracks, stations, power supply, safety and signaling equipments and telecommunication systems. Rolling stock includes cars and locomotive. Pachl (2009) categorizes infrastructure and rolling stock as hardware part of the railway system and there is a software part which consists of schedule and operating rules and safety regulations.

Operating trains on the rail network has a major difference from railroad transport in that the driver cannot steer the train and train movement is limited to the tracks. This simple rule affects the management of train movements. Trains can only pass each other or change route where the infrastructure allows them to do it. Moreover, trains cannot pass the maintenance activities on tracks. So it can be seen how train movement is dependent on the track layout and signaling system. This unique characteristic of railway transportation make the railway operation management more difficult.

Given that the train movements are dependent on the tracks and signaling system, there should be some means to define this relation. Railway traffic management links the traffic flow and the infrastructure together in order to produce a schedule which determines not only how long a process takes but also what resources it uses. The notion of producing a schedule by linking traffic flow and infrastructure provides the basis for the next sections which start with some discussions on terminology and concepts of train planning and control in Section 3.1. Then, we explain the purpose and scope of the train scheduling and rescheduling in Section 3.2. Later, what train routing and rerouting problems address with regard to their aim and scope is clarified in Section 3.3. Finally, we show a real-world case study that we use in this thesis in Section 3.6.

3.1 Train planning and control terminology and concepts

Different terminologies are used in different countries, or even in different regions of a country, which can lead to misunderstanding between international researchers and practitioners. In particular, there is a big difference between British and North American terms used for railway network and operations. In order to illustrate how the railway system works, we need to define some basic terms and concepts. This will help us to lay a common ground to explain train scheduling and routing processes in this study.

3.1.1 Network topology and traffic

Railway scheduling decisions are made complicated by two main factors; size of the problem which depends on the complexity of the track topology and infrastructure, and trains traffic. Firstly, we compare different approaches towards modeling network topology and their influence on the problem complexity, and then we discuss how train traffic can affect the problem complexity.

Depending on the level of detail about the track topology and train dynamics, train scheduling and routing problems can be classified as microscopic, mesoscopic or macroscopic problems (Caimi, 2009). Each approach has some advantages over the other one and selecting an approach depends on the characteristics and goals of the planning problem. The macroscopic approach is to handle globally an aggregated topology with a simplified safety model while the microscopic solves a scheduling problem on an infrastructure with more local details about the topology and operations. The mesoscopic models are useful when all microscopic details are not available and some standard assumptions are made for the missing data. Obviously, the microscopic planning produces reliable results with high quality. On the other hand, macroscopic planning models an aggregated structure more roughly.

When dealing with a long term plan, it makes much more sense to consider less details and use a macroscopic model. On the contrary, high accuracy is more meaningful and manageable for a short term planning. Most of the mathematical programming models consider macroscopic models as the size of the problem becomes very huge when the level of details are high. Conversely, the simulation tools which are usually used in railway companies consider microscopic models. Many studies in the literature try to create a macro timetable without considering the feasibility at the micro level. A fewer number of studies has been carried out to generate a micro timetable on a complete or partial network (Caimi, 2009).

There are a few approaches in the literature which consider a top-down approach and solve a macroscopic model of the general railway system and then consider the details of the railway system in a microscopic model (See Klemenz and Schultz (2007), Caimi (2009) and Schlechte (2012)). It should be mentioned that generating the timetable manually is still the most prevalent planning method in practice in railway system in the world (Schlechte, 2012). In our experience, the manual construction of the timetable also happens in the UK. The feasibility of the manually generated schedule is then checked by using microscopic models developed in the railway network simulation tools such as OpenTrack and RailSys which are nowadays available in most of the train companies.

In summary, it seems to be more reasonable to deal with operational scheduling and rescheduling in micro level. Micro scheduling can consider details of the incident precisely and exploit several properties of different areas of the infrastructure. This study investigates train scheduling and rescheduling problems at the micro level including detailed information about the tracks and train movements in Chapter 5. However, less details are desirable in tactical level planning as it only adds to the problem complexity. Thus, we consider the train routing and rerouting problems in macroscopic level in Chapter 6 (See Section 4.1 for more information about different levels of railway planning process).

Train traffic is heterogenous in many countries including passenger and freight trains. Passenger trains may have different priorities with regular commuters, long-distance passengers with connections and freight trains may carry different cargo such as express freight, bulk goods. So different types of trains have different preferences, speeds and destinations. On the other hand, trains are highly interdependent and interlinked and any disturbance can jeopardise the whole plan compromising of trains, crew and rolling stock. It is obvious that a more utilised network, or a more complicated infrastructure, can be effected to a bigger extent by a disruption. Therefore, planning in a congested area with a complicated infrastructure and mixed traffic is a challenging task.

3.1.2 Blocks and signals

Tracks are defined as the roadways of the railway. Tracks are also referred to as lines in railway operations. Tracks are categorized to three classes. The first one is *main tracks* or *running lines* which are used for regular train movements. When the track passes through the station, it is called *platform*. The second class is called *loop* where trains pass and overtake on the tracks. The last term is a *siding* which is used for train shunting.

A locomotive moving with cars on a track has a *running* movement whereas *shunting* movements are for coupling and decoupling cars and locomotives. A *junction* is a node where a train can change from one line to another. A *yard* is the arrangement of sidings for assembling, classifying and storing cars. An assembly of facilities including lines and signaling system at a terminus or on a intermediate line is called *Terminal*, where there are track groups for the train arrival and departure, yard facilities for cars to be sorted or stored and sometimes maintenance facilities for rolling stock.

The movement of a train on the network is controlled for safety reasons by lineside signals which divide the network into track sections called *blocks*. In this study, we also introduce a new concept called *track section* which is a set of aggregated blocks. More formal definition of track section is given in Section 6.2. Under a *fixed block system*, there are fixed block sections protected by signals. Until the previous train clears the signal, the train cannot enter the block. Passage of a train on fixed blocks from its origin to its destination follows a so-called *blocking time theory* and can be illustrated in a time-over-distance diagram called *blocking time stairway*. The blocking time stairway is used in fixed block systems to define the minimum time interval between two following trains in each block section. More details about blocking time theory railway operation planning on the fixed block system is given in Subsections 3.1.3 and 3.1.4

On the contrary, in a *moving block system*, a safe zone is calculated by computers around each moving train and the other trains are not allowed to enter this zone. For this, knowing the precise location and speed of the train is required. Information are directly sent to and received from the trains, and no lineside signal is necessary. As a result, trains can follow each other at exact braking distance. Comparing it with the fixed block system, the length of the block is reduced to zero so there is no blocking time, but all the other components of the blocking time theory exist. So the stair-way shape of the blocking time diagram is transformed to a continuous time channel. This system can increase the capacity of the line if trains have the same speed so the capacity improvement is quite limited on lines with mixed traffic. For this reason, the technology is not mature enough to be applied to a wide extent and signaling operations with fixed block system which is the most common signaling system. For a more comprehensive and extensive study of signaling systems and principles of railway operations, the reader is referred to Pachl (2009).

Various signaling systems are used in different countries. These systems can be categorized in two main classes of *speed signaling* and *route signaling*. The signal in route signaling indicates the train route to follow while speed signaling defines the speed limit that the train can run. In route signaling, the driver knows the speed not to be exceeded in a route. Speed signaling is applied in most of the modern signaling systems. Modern European signaling systems use additional speed or route indicators and the signal itself lets the driver know about the occupation of the following blocks.

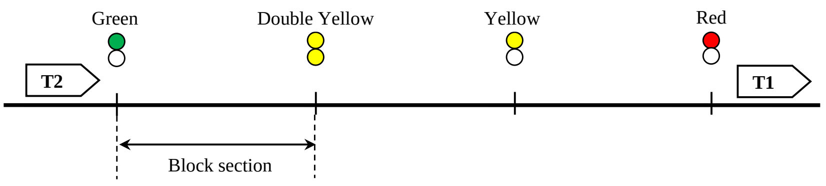

FIGURE 3.1: Four-aspect signaling system

In general, there are two blocking principles named as *one-block signaling* and *multiple-block signaling*. If the signal can only show the state of the one block section ahead, the signaling system is a one-block signaling. Therefore, the system has only two aspects including red for stop (danger) and green for clear. Accordingly, a train can enter a block if the signal is green; otherwise, it should stop. In contrast, more than one block state is identified in multiple-blocking system. *Two-block signaling* is very common and it comprises of three aspects of red for stop, yellow for approach (caution) and green for clear. Similarly, there is *three-block signaling* with four signal aspects of red, yellow, double yellow and green which is more common in British signaling.

In this study, we consider *four-aspect signaling* which is common for the main lines of the UK network, as shown in Figure 3.1: red for stop (danger), yellow for approach (caution), double yellow for advance approach (preliminary caution) and green for clear. Each aspect gives information for four blocks ahead, thus enabling the train driver to adjust the speed and to keep sufficient separation between trains to allow safe braking. It should be noted that we have assumed that trains run with a fixed speed to exactly keep them three blocks away from each other.

3.1.3 Blocking time theory

The *blocking time theory* is about the concept of authorising the train movement in a way that a block exclusively assigned to a train at a time. So the *blocking time* is the time interval that a block is occupied by a train and therefore blocked for the other trains. This time does not only include the time the train travels between the block signals, but also the following time intervals including the time to clear a signal, a certain time for the driver to view the clear aspect, the approach time between signal with approach aspect and the clear signal at the entrance of the block, the clearing time to clear the block, and the release time to unlock the block. There is no approach time if the train has a scheduled stop at the signal at the entrance of the block section (Pachl, 2009).

The blocking time stairway includes consecutive blocking times of a train over fixed blocks. So it is the time-over-distance diagram of the blocking times of a train which passes several blocks on its route. Thus, operational use of a line by a train is shown by a blocking time. This diagram is also helpful to define the minimum headway of two trains. Therefore, the train movements on blocks which are safeguarded by signals can be formally interpreted by a blocking time stairway which helps us to determine the minimum safety interval between following trains (Pachl, 2009).

3.1.4 Timetable components

The timetable should determine several parameters for each train including the day on which train runs, train route, arrival and departure times at each station and the maximum speed. The main components of a timetable are the scheduled running time, dwell time, headway and their buffer times which are explained in the following.

Scheduled running times: Each train needs a minimum specified time to travel between two points which can be two signals, two stations or two *timing points*. A Timing point refers to a location which can be a station or junction which is used to break up the whole journey into series of passing times. It is also used as a reference point in the route to schedule the train to reach there as planned. The first way to calculate the running time is to analyze train dynamics and track topology with some simulation tools and solve corresponding differential equations. The other way is to determine minimum running time by some test runs. Pachl (2009) classifies two types of recovery time to be added to the shortest possible running time. A *running buffer time* is added to the pure running time to absorb the running time variations due to changes in train dynamics and weather conditions. The first type of the buffer called *regular recovery time*, is added as a percentage of the pure running time. Regular recovery time is about 3 to 7% on European railway and 6 to 8% for north American passenger trains. The other type of buffer is *special recovery time* to compensate for maintenance or construction work.

Dwell time: If there is a scheduled stop at a station, the train needs a minimum time for the passengers or freight to board/load and alight/unload. There are various models to calculate the dwell time precisely considering several factors such as passenger demand pattern, capacity and layout of the rolling stock, entrance design, and accessing points in the station. However, planners generally do not take boarding and alighting times into consideration and simply use pre-defined values to calculate it with regard to the station and train classification (Luethi, 2009). The dwell time buffer is used to reduce the effect of changing demand and changing dwell time. It is calculated based on principles and variations according to the historical data. Connection times as well as dwell time constraints affect the train departure time in bigger stations. Dwell time buffer reduces the possibility of delay propagation similar to running time buffer.

Headway: Safety considerations impose a separation time which is the minimum time between two consecutive trains travelling on the same block. Headway is dictated by signals and it is helpful to reduce the effect of a train delay on the other ones which generally minimizes the delay propagation in the network. On a fixed block system, the minimum headway depends on blocking time. Nowadays, planners still use fixed headway between two trains and the blocking time theory is rarely employed (Luethi, 2009). The fixed headway includes the buffer time, but it means that the scheduling decisions are dependent on the planner's experience and historical data which may not necessarily result in the maximum utilisation of the infrastructure.

3.2 Train scheduling and rescheduling purpose and scope

Train scheduling or *timetabling* is the process of creating a plan from scratch which makes the optimum use of the infrastructure by coordinating the train paths and it provides the timetable for the passenger information. The schedule also makes the train traffic predictable and it is the essential input for scheduling rolling stock and crew and traffic control. Scheduling can be done in tactical level and operational level. Train scheduling can be performed at a tactical level, which can take up to a year. Tactical scheduling is usually performed in Europe where a master schedule is created to determine the train timetable and routes ahead of time to be executed in real-time (D'Ariano, 2008). Passenger trains should be scheduled in tactical level. As the freight operators are facing changing demands they schedule trains in operational level and a few hours before train departure. Operational scheduling is usually applied in the North America. In the Europe, extra trains are added to the timetable in the operational level when it is required to run more freight trains. In summary, the aim of train scheduling is to make the best usage of the existing capacity by allocating trains to the blocks (See Section 4.1 for different levels of railway planning process).

The *Time-distance diagram* or *traffic diagram*, which is a common language of the railway transportation, is an essential tool for traffic planning and the basis of control of current operations . The North American diagram is called *stringline graph* and it is only used for capacity analysis in the early stages of the planning (Hansen and Pachl, 2008). It consists of a time axis and a station axis which can be either horizontal or vertical. Train movements are shown as train paths with the train names indicated on them. The intersection of the train path and a parallel line to the time axis at each station represents the time period spent in that station.

When trains are operated according to a plan, disruptions can cause deviations to that plan due to various causes such as train delays, accidents, track maintenance, no-shows for crew, and weather conditions. *Train rescheduling* responds to disruptions in an operational level, where a new schedule is required in a matter of minutes or seconds. Rescheduling is to respond to disturbances which arise during actual operation in order to identify and resolve conflicts. The new schedule should take into account all the factors and circumstances around the event that has happened and produce a conflict-free timetable with a provision of traffic control (Hansen and Pachl, 2008).

When trains are delayed, they generate a knock-on effect due to the their high interdependency. Therefore, the main purpose of the train rescheduling is to minimize the consequences of the disruption on other trains. In order to have an efficient rescheduling goal, one can minimize the overall delays of trains, minimize the delay of trains based on their priorities or minimize the time to return to the original timetable as close as possible. Therefore, the objective function in our model is to minimize the delay propagation.

In real-life operations, a dispatcher resolves the conflict and reschedules trains in different ways. Some of possible modifications are using different routes, extending dwell or running times, introducing additional stops and cancelling trains in a complete or partial route. Still, many dispatcher perform the disruption management process manually, maybe with the support of some tools which differ in various countries. The dispatcher gets the information about the actual train movement from the equipments such as track circuits. Then, he makes rescheduling decisions with an attempt to keep the effect of the disturbance to a minimum using his experience and some representation tools such as time-distance diagrams. Finally, the decisions should be relayed to the signallers if a train order or route is changed.

In summary, according to the safety principles, only one train can travel on a block at a time and a *conflict* occurs when more than one train are assigned to a block. Another issue is the *deadlock* that arises when certain trains are currently positioned in a way that none can move further without causing a collision. A deadlock happens usually in complicated networks with bidirectional tracks. Thus, being conflict-free and deadlock-free are essential characteristics of a feasible schedule. The above-mentioned operational and safety issues are crucial to determine a schedule and they are treated as constraints in our model as it is discussed in Chapter 5.

3.3 Train routing and rerouting purpose and scope

Similar to train scheduling and rescheduling, train routing and rerouting can be performed in different levels of railway planning process. Routing problem aims at deciding a feasible route for each train. Other planning decisions can also be made at the same time such as determining the type or frequency of the train route, arrival and departure time of the trains at specific points of the route. Generally, a train route defines the usage of the infrastructure for a train movement in the network in time.

Lusby et al. (2011) define routing as a planning phase following timetabling process for an aggregated network. Checking the compatibility of the timetable to the detailed topology of the tracks and finding a feasible route through nodes is called a *train routing* problem. If a node is a station, the problem is called a *train platforming* problem. In this special case, selection of a route leads to choosing a platform. In general, this process is performed in tactical planning. However, one may apply train routing at a strategic level to explore several questions about construction or modifications of the infrastructure and resources with regard to the routing decisions.

During day to day operations, disruptions can block some routes and can cause conflicts among train routes or even deadlocks in the network. Dispatchers need to be able to recover the plan by defining new routes in order to restore the feasibility of the plan and to minimize the knock-on effect of the delay. Responding to disruptions and redefining the routes to recover the plan in the operational level is known as the *train rerouting* problem. In Chapter 6, we look at the routing problem and devise methods which are applicable in tactical and operational routing.

3.4 Formal problem definition

Train scheduling problem can be formulated as a job shop scheduling problem. According to safety rule, only one train can traverse on a block at a time. This is similar to job shop scheduling problem where each machine can process one job at a time. Using this analogy, trains are considered as jobs and blocks are known as machines. In addition, a train traversing a block or dwelling on a block is similar to a job being processed on a machine operations. So an operation is analogous to a train running or dwelling on a block. The analogy between train scheduling problem and job shop scheduling problem is summarised in Table 3.1.

Table 3.1: Analogy between train scheduling and job shop scheduling problems

Train scheduling	Job shop scheduling
Train	Job
Block	Machine
Train traversal or stop on a block	Operation

The classical job shop scheduling problem should be modified in order to consider safety and operational constraints of the train scheduling problem. The objective function is to minimize the total delay of trains with priorities which is equivalent to minimize the total weighted tardiness. It should be noted that it is unlikely to get a solution with early trains due to tight due dates. Thus, earliness is not considered in the objective function. Also, if a train is early, it is kept at a station to reach its departure time in the timetable. So it is reasonable to consider only train tardiness in the objective function formulation.

Furthermore, one should notice that tardiness is considered only for destinations although there are intermediate stops in the case study for which the tardiness is not taken into account. Calculating tardiness only at the destination seems to be a reasonable approximation as the tardiness on intermediate stops are small. We look at a partial network in our case study which is about 15 km long and stations are close to each other. Therefore, it is more practical to reflect delays at the destinations.

There are three constraints in the train scheduling problem which are called running/dwell time constraints, headway constraints and signaling constraints. Whereas all mentioned constraints are formulated for both following and opposite trains, signaling constraints are only considered for following trains. Signaling constraints help with maintaining the necessary separation between two following train according to the network signaling system. In our case study, it is assumed that following trains need to be three blocks away all the time so that they can run on the green signal all the time with a fixed speed. One can easily see that in case of a disruption, this is not possible by only considering the headway constraints. Because trains can get closer than three blocks during a disruption if the signaling constraints are omitted. More about how signaling constraints are formulated can be found in Subsection 5.2.3.

These modifications are elaborated in Section 5.2. Given predetermined routes for each train, a schedule defines the entry times and the order of the trains on each block. A summary of inputs and outputs for the train scheduling problem is given in Table 3.2 and Table 3.3 .

Table 3.2: Train scheduling inputs

Train scheduling	Job shop scheduling
Set of trains	Set of jobs (27 jobs)
Set of blocks	Set of machines (135 machines)
Departure of a train from its origin	Release time of a job
Arrival time of a train at its destination	Due date of a job
Priority of a train	Weight of a job (=1, equal weights)
Train running or dwell time	Operation of a job
Train headway	Delay between starting time of two consecutive operations of a job
Following trains signaling separation time	Sum of operation time of a job on a machine and its immediate (and second) successor operation in two-aspect signaling (four-aspect signaling) for following jobs

Table 3.3: Train scheduling outputs

Train scheduling	Job shop scheduling
Starting time of a train on a block	Starting time of a job on a machine
Train delay	Tardiness of a job

Train routing problem is similar to train scheduling problem in terms of operational characteristics. Therefore, we can use the analogy between train routing problem and job shop scheduling to formulate the problem. As mentioned before, trains are analogous to jobs and the traversal or stopping of a train on a track section is considered as an operation. However, there are a few differences between train scheduling and train routing problem formulation. The main dissimilarity is in the definition of the machine. A block is considered as a machine in the job shop scheduling problem, whereas a track section is considered as a stage in train routing problem. A stage includes single or multiple tracks which are analogous to single or parallel machines, respectively. Train routing problem and job shop scheduling problem have the following analogy which is shown in Table 3.4.

In order to formulate safety and operational constraints of the train routing problem, the classical job shop scheduling problem should be modified. Similar to train

Table 3.4: Analogy between train routing and job shop scheduling problems

Train routing	Job shop scheduling
Train	Job
Track section	Stage
Single track	Single machine
Multiple tracks	Multiple machines
Train traversal or stop on a track section	Operation

scheduling problem, we aim to minimize total delay of trains with priorities which is translated to the total weighted tardiness. It should be noted that earliness is not considered in the objective function and delays are calculated at the destinations according to the same reasons mentioned before for the train scheduling problem. The new constraints which are added for train routing problem include running/dwell time constraints and headway constraints. It should be noted that signaling constraints are not required to be considered in train routing problem as aggregated blocks are considered in the train routing problem. Necessary modifications are discussed in detail in Section 6.2. Train routing problem aims to improve the schedule by defining more efficient train routes and sequences on track sections. Inputs and outputs for the train routing problem are given in Table 3.5 and Table 3.6.

Table 3.5: Train routing inputs

Train routing	Job shop scheduling
Set of trains	Set of jobs (27 jobs)
Set of track sections	Set of stages (18 stages)
Set of of single and parallel tracks	Set of single and parallel machines
Departure of a train from its origin	Release time of a job
Arrival time of a train at its destination	Due date of a job
Priority of a train	Weight of a job (=1, equal weights)
Train running or dwell time	Operation of a job
Train headway	Delay between starting time of two consecutive operations of a job

Table 3.6: Train routing outputs

Train routing	Job shop scheduling
Starting time of a train on a track of a track section	Starting time of a job on a machine of a stage
Train delay	Tardiness of a job

In general, the running or dwell time is similar for all trains in both train scheduling and routing problems. This special property holds for the case study which is used in this research. This means that all operations on the same machine have equal length. However, we deviate from this assumption when we have disruptions. Thus, the algorithms that we exploit to solve train scheduling and routing problems consider the general problem formulation for the case that operation times are different for the same machine. More specifications about the case study for the train scheduling and routing problems are mentioned in the following section.

3.5 Our case study

The experimental evaluation of the suggested scheduling and routing techniques is based on a real-world case study. We report the effectiveness of scheduling and routing methods on a bottleneck area in the South East of the UK where the network has a complicated structure including several junctions and stations. The so-called Kent area was chosen due to the complexity of infrastructure and the dense traffic on interconnected lines for passengers in and out of London, East Sussex and the Channel Tunnel. Figure 3.2 shows a map of the Kent area.

NetworkRail (2010), the train infrastructure managing organisation in the UK, discusses significant issues of the Kent route. NetworkRail has forecasted a growth of 30 percent in demand from 2008 to 2022. There is strong growth in the number of off-peak commuters as well as the peak period commuters. A provisional growth is also expected in freight based on the potential new terminals planned in the area. According to NetworkRail's report, critical sections mainly in the London area operate near capacity for several hours in a day. Many of the capacity and performance issues are due to the constraints in the London Bridge area.

The London Bridge area is a critical corridor with known capacity and performance issues, which are made more complex by the addition of a new high speed line HS1. Therefore, we focus on the partial network of the London bridge area which is about 15 km long and it includes busy stations like London Charing Cross, London Waterloo, London Cannon Street, New Cross and Deptford, and a total

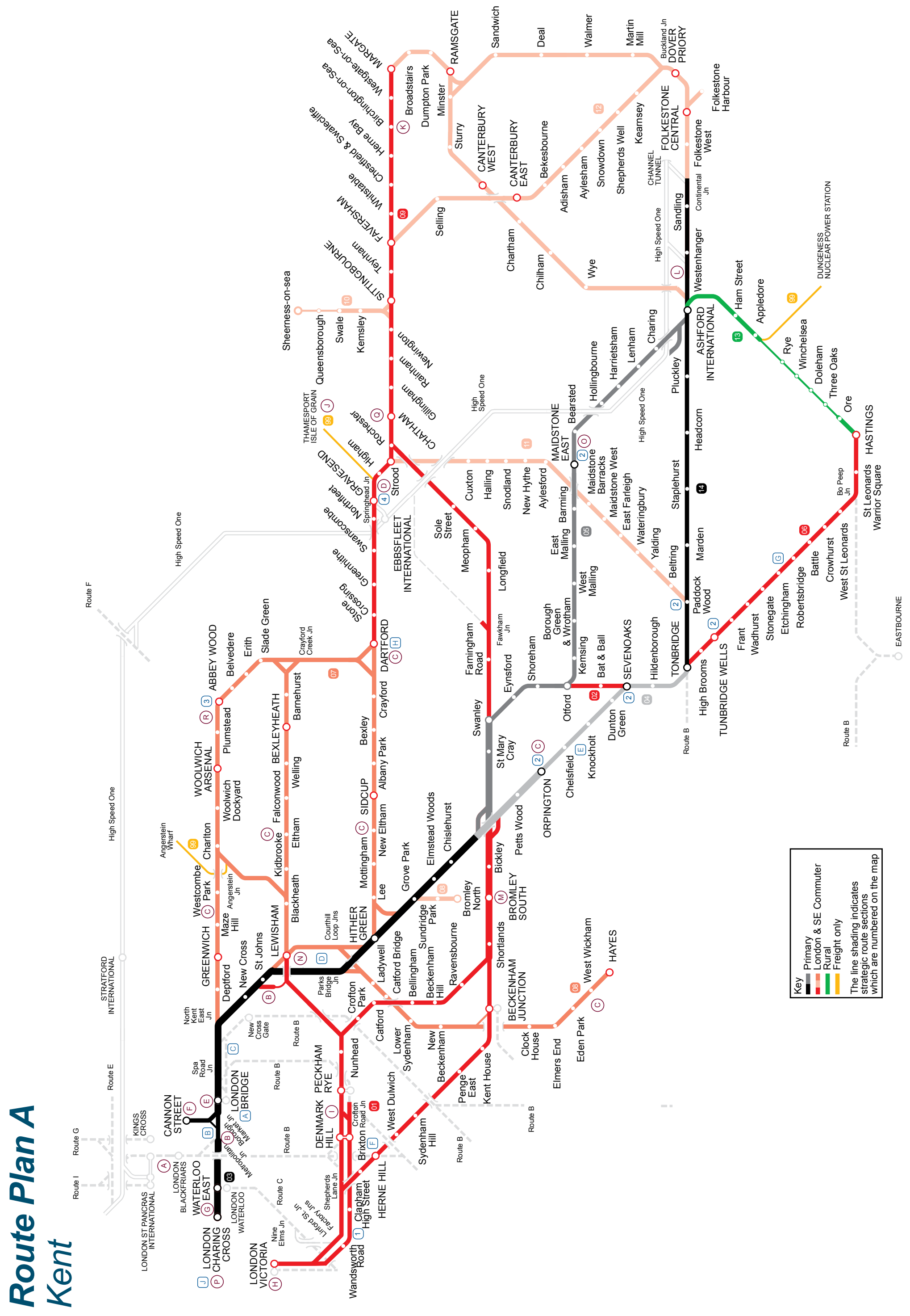

FIGURE 3.2: Map of the Kent area. Adapted from NetworkRail (2010)

of 28 platforms. The network includes 135 blocks with unidirectional and bidirectional traffic. Passenger trains start their journey from either Charing Cross or Cannon Street and travel through 75 blocks down the network in order to leave the network, or they enter the network and travel through 76 blocks up the network and terminate at one of the mentioned stations.

Our experimental data focus on the off-peak services because there is an on-going strong growth in off-peak period commuters. The timetable cycles every 30 minutes for the passenger trains and it includes 27 trains. The train timetables, running times and track diagrams are provided by the primary train operator for this region of the UK, the Southeastern train operating company.

3.6 Data collection and refinement

This section describes the procedure to collect, refine and implement the train operation and control data in this case study. There are several challenges in each step which are addressed in the following.

Confidentiality of data

Accessing rail data is one of the major challenges as it is strict confidentiality rules. Information is not easily shared cross industry and rarely shared outside of industry circle. Although NetworkRail supervises TOC's (Train Operating Companies) and they work closely to offer the best service with high performance, TOC's need to compete with each other to run their services on the shared network. Under these circumstances, information is highly valuable.

Locating the data

Due to privatization of the British Railways and deregulation in the UK, train operation and control are decentralized. Due to this fact, further knowledge of the rail industry in the UK was necessary to locate proper data. The organizational structure is directly involved in train operation and control. Hence, some general background about the rail industry was required. This requirement is addressed by some studies about railway engineering in general. It should be noted that railway engineering is a multi-faceted discipline and it includes a wide range of disciplines such as civil engineering, electrical engineering and mechanical engineering. In order to gain enough knowledge about railway operation and control in the UK, extensive official documents from NetworkRail and TOC's were studied to learn more about the industries structure, the actors, and rules and regulations.

Initiating contacts with industry and railway experts
Some contacts were initiated with Southeastern train operating company and the Transport Research Group (TRG) in the School of Civil Engineering, University of Southampton. We made visits to the company to become familiar with train planning process and disruption management. After some negotiation with both groups, the data about train dynamics and signaling information were received. Finally, data from Southeastern was selected as the case study with a more complete set of data. Meetings with TRG members was also beneficial.

Data cleaning
One of the objectives of the study is to provide a precise representation of the train operations and network topology in the UK network. In order to achieve the required details in scheduling and routing, planning in the microscopic and mesoscopic level is desirable. As the amount of detailed data in the microscopic level is very extensive, data cleaning and refinement became a long and time-consuming process. Another factor which added to the difficulty of the process was that we had to extract the corresponding data for the partial network from massive data for the whole network.

Terminology and technicalities
In order to extract relevant data, some knowledge of technicalities and abbreviations in the UK rail industry was necessary. Train operation and control data are coded and classified so that the huge amount of data can be used and maintained efficiently. To extract the relevant data for the case study, we had to decode them in order to refine the relevant data for the case study.

Timetable components
Running time and headway values are extracted from huge databases of the company. Network topology is modelled based on the track diagrams. Track diagrams represent the detailed layout of the tracks and signaling system which are very useful in the train planning process. However, the amount of technical details

make them hard to read. Some knowledge of the signs and abbreviations is required to assist the reader in searching the maps and refining the relevant data as not all data is of interest in this case study. It took some time to shape up a refined diagram of the London bridge area. Figure 3.3 shows the configuration of the London Bridge area which is used in the train scheduling and disruption management in microscopic level.

In order to apply train routing and disruption management in mesoscopic level, blocks are integrated according to timing points on the network which are called TIPLOCs (See Section 6.2 for more details about TIPLOCs and integration of blocks). The configuration of the London Bridge area with regard to possible routing options is represented in Figure 3.4.

After developing the track diagrams, we need to adjust running time and headway values with regard to the track diagrams. In addition, information of train services and their frequencies are extracted from public passenger timetables. No detail is available about the exact routes to define fixed routes for train scheduling and rescheduling. Therefore, information about platforms on National Rail Enquiries (2013) website is used to define routes. The platform information which are in line with the track diagrams are used in order to define the most reasonable train routes. It should be noted that National Rail Enquiry is owned by the Association of Train Operating Companies (ATOC) and their webpage provides online journey planning, timetable and ticketing service.

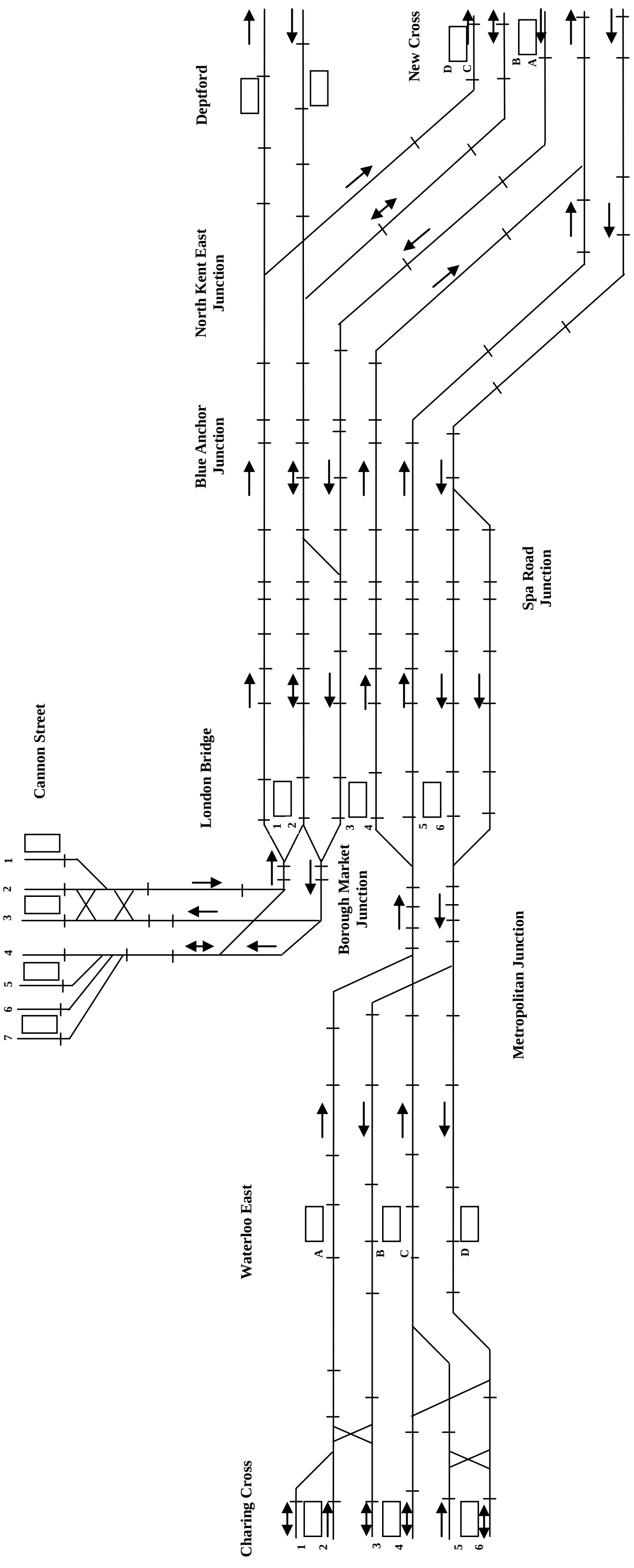

FIGURE 3.3: London Bridge diagram developed for scheduling problem

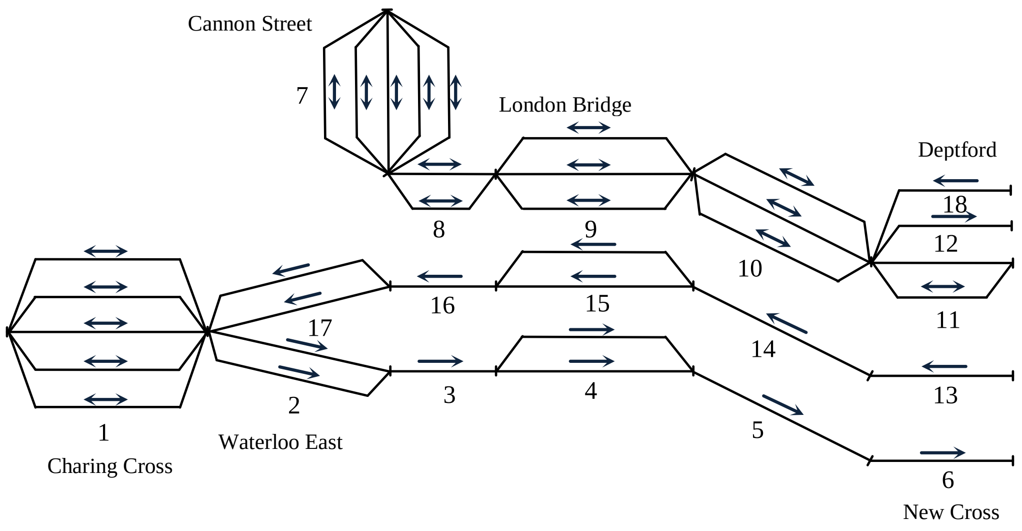

Figure 3.4: London Bridge diagram developed for routing problem

Chapter 4

Literature Review and Background

The railway is an important mode of transportation in many countries with ever increasing demand in passenger and freight services. Being a capital intensive industry, railway management should make the best usage of the existing capacity. Providing additional capacity for passenger and freight is a key issue for many train companies. To fulfill the increasing demand for additional capacity, there are two solutions; one is to construct new tracks and another is to release capacity through better planning decisions. While former solution is very expensive, the latter is through better scheduling decisions to avoid loss of capacity on the current tracks. So train scheduling in the offline mode and rescheduling in online mode help with railway traffic management to fully utilise the current railway capacity.

4.1 Railway planning process

The railway planning is a complex process which is made tractable for railway companies by implementing hierarchical decision-making approach (Lusby et al., 2011; Bussieck, 1997). It also conforms to a strategic/tactical/operational classification of Assad (1980) as shown in Figure 4.1. Huisman et al. (2005) categorize railway resource planning into strategic, tactical, operational and short-term. Based on the organization of the passenger operator NS Reizigers in the Netherlands, they use a new convention of dividing the operational level into two horizons. First one is two-month scheduling which is the called operational and the second is daily modifications referred to as short-term.

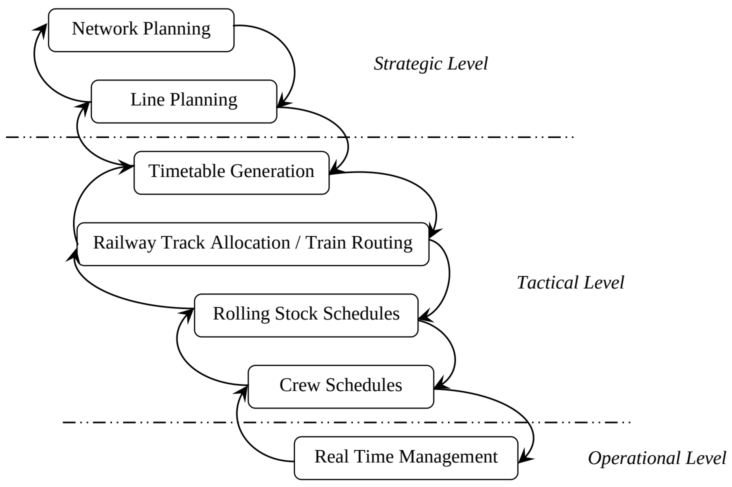

FIGURE 4.1: The railway planning process

Strategic level is about resource acquisition and it is a lengthy process which may take 5 to 15 years. Strategic decisions are usually related to the planning the desired service level and providing the required capacity considering resources including infrastructure, rolling stock and crew. The main issues addressed in the strategic level are network planning and line planning which are discussed in Sections 4.1.1 and 4.1.2, respectively.

Tactical level concerns resource allocation and it may take 1 to 5 years. One of the main tactical stages is generating the basic timetable given that the infrastructure is fixed which is discussed in Section 4.1.3. Usually the timetable needs some adjustments to be compatible with the detailed layout of the network. Thus, train routing or railway track allocation is the next stage in the tactical planning (See Section 4.1.5). Afterwards, certain number and type of the rolling stock should be assigned to have trains with a certain length. It is a big investment as the rolling stock is very expensive and its maintenance and power supply is also involved in the planning. A balance should be found between the costs and satisfying passengers' seat demand. Shunting problem arises when scheduling the rolling stock to move the train units to a shunting or parking yard to wait until they are needed for operation according to the timetable. To cover the timetable, proper crew should also be assigned to the train services. Crew Planning is a complicated problem due to the size and the operational constraints of the problem. Rolling stock scheduling and crew scheduling are not discussed in this thesis as each of them are an entire research direction by themselves.

Whereas the first two levels of planning include planning horizons up to a few years, the operational level is of specific interest with planning decisions from 24 hours up to a year. Resource consumption is the main focus in operational planning. On daily basis, disruption management is necessary to avoid disruptions due to weather conditions, crew no shows, maintenance and etc. So real-time management role is to monitor the planned timetable and respond to the disruptions in order to get back to the original plan. Train rescheduling which is discussed in Section 4.1.6 is a typical problem to solve in real-time management. The other common problems are rolling stock and crew rescheduling which are in the scope of this thesis. The major subproblems of timetable adjustment, rolling stock rescheduling and crew rescheduling are conventionally solved as sequential separate tasks.

As it can be seen in Figure 4.1 the planning task is divided into separate tasks. According to Bussieck (1997), the tasks are not only solved in a top-down manner, but also it may be required to go back to the preceding tasks or look at succeeding tasks when solving the current task. Although this approach has the advantage of providing manageable problems, it may not result in an overall optimal plan. Since optimal plan of a task is the input of a subsequent plan.

We refer the reader to Assad (1980), Bussieck (1997), Cordeau et al. (1998), Crainic (2003), Huisman et al. (2005), Ahuja et al. (2005), Caprara et al. (2007), Lusby (2010) and Cacchiani and Toth (2012) for reviews on railway optimization. Bussieck (1997) offers a hierarchical classification of railway planning with a variety of examples from line planning and scheduling to rolling stock circulation. Assad (1980), Cordeau et al. (1998) and Crainic (2003) provide surveys on freight transportation looking at rail yard models and locomotive and car management. Whereas Ahuja et al. (2005) focus on planning and scheduling in the US railroad freight industry, Huisman et al. (2005), and Caprara et al. (2007) give an overview of passenger transportation planning problems in the Europe. Lusby (2010) summarizes more recent works on train timetabling and routing and Cacchiani and Toth (2012) survey is suggested for more recent works on train timetabling.

In this chapter, we classify the railway traffic studies with respect to the time perspective of the track allocation problem which can be online or offline planning and the network complexity of the railway infrastructure which can be a single track or a general railway network. Hence, main contributions in strategic level including network planning are discussed in Section 4.1.1 and line planning in Section 4.1.2. Then, there is tactical decisions where we survey train timetabling or scheduling on single track networks in Section 4.1.3 and proceed to cover studies on more general networks in Section 4.1.4. Section 4.1.5 reviews work on train scheduling where routing decisions are also involved. The topic of Section 4.1.6 is train rescheduling during disturbances in operational level.

4.1.1 Network planning

Network planning is a strategic planning phase which is about construction, and/or modification of the infrastructure. Newton et al. (1998) model the railroad blocking problem as a mixed integer programming problem. Yards are presented by nodes whereas blocks are shown by arcs. They solve the problem with solution methods such as branch and bound and column generations embedded in a strategic decision support system. The same formulation is implemented in the study by Barnhart et al. (2000). A decomposition approach based on Lagrangian relaxation methods is developed which is not suitable for larger problems in application because of the computational time.

Ahuja et al. (2004) formulate the railroad blocking problem as an integer programming problem which is large to be solved by the current software packages. Thus, they develop a dedicated algorithm to solve commercial environment problems to near optimality in few hours. The algorithm makes use of a novel large-scale neighborhood search approach introduced by them. Gorman (1998) solves a train routing and sequencing problem with a novel decomposition approach. The problem is formulated as mathematical programming with binary variables for each train service. Computational tests report on freight railroad in the US to analyse strategic scenarios for operations by the operators.

4.1.2 Line planning

Bussieck (1997) has done a significant work on line planning. They consider the use of mathematical programming methods in public rail transport in view point of an integrated decision making process with three different levels of strategic, tactical and operational planning. Albrecht (2009) consider solving line planning and timetabling problem simultaneously in a two-level approach for suburban railways.

4.1.3 Train scheduling on single track networks

Train scheduling which is also called train pathing, or train dispatching problem or train timetabling is to determine the timing and ordering plan based on the fixed routes for trains by taking into account optimization objectives. The pioneering publication of Szpigel (1973) exploits the job shop scheduling problem to formulate a train scheduling on a single track network in Brazil and solves it by using a branch and bound algorithm. He reports computational experiments for five track sections and ten trains. An enumeration strategy for all potential conflicts to create an optimal meet-pass plan is presented by Sauder and Westerman (1983). The objective is to minimize total weighted tardiness for a single track railway.

Jovanovic and Harker (1991) develop a decision support model called SCAN (Schedule Analysis) system for the tactical scheduling of freight trains. The study considers both combinatorial optimization and simulation approaches for a single track network. They consider instances including 24 train lines and about 100 trains. Kraay et al. (1991) present a mixed integer nonlinear program for train pacing problem. To minimize fuel consumption and delay, the train speed profile is determined. A branch and bound procedure with cutting planes is employed to determine a meet-pass plan.

Carey and Lockwood (1995) propose a mixed integer program for a single line with traffic in one direction. A heuristic approach is developed train pathing problem on a network similar to Szpigel (1973). Carey (1994a,b) extends the work to more complicated networks with multiple lines and platforms and bidirectional traffic on single lines. A set packing integer programming model is introduced by Brannlund et al. (1998) for a bidirectional single track connecting 17 stations in Sweden. They solve the model with Lagrangian relation techniques successfully for 30 trains.

The studies by Oliveira and Smith (2000) and Oliveira (2001) are based on modeling the train scheduling problem as job shop scheduling problem with extra constraints. They suggest constraint programming techniques to minimize total delay along a single track network. Dorfman and Medanic (2004) develop a greedy travel advance strategy using a discrete event model for scheduling trains on a single line which can avoid deadlocks and handle perturbations well. They extend it for double track networks and train variable priorities.

Zhou and Zhong (2007) propose a generalized resource-constrained project scheduling formulation for a single track train timetabling problem. To minimize the total travel time considering a set of operational and safety constraints, a branch and bound algorithm is developed which sequentially adds precedence relation constrains to resolve conflicts between trains. The longest path algorithm solves each subproblem solution for earliest start time of each train on each segment.

4.1.4 Train timetabling and periodicity on general railway networks

As stated by Caprara et al. (2007), after the line planning problem determining the train routes, and types and frequency of trains on each route, the actual timetable of the each train on a certain part of the network is defined which is referred to as train timetabling problem (TTP). TTP can be categorized as cyclic and noncyclic timetables. When the network traffic has a high density, noncyclic timetables help with the capacity pressure to allocate optimally the required train paths requested by train operators.

In practice, the train operating companies propose their preferred timetables to the infrastructure manager and bid for the tracks. Then, the infrastructure manager modifies the collected timetables to consider operational constraints such as safety margins between trains operated on the same track. Infrastructure manager returns back to the operating companies with a modified timetable. These modifications should be kept as few as possible. The process is repeated if the train operating company does not accept the proposal and consequently puts a new proposal.

Based on a graph theory representation, Caprara et al. (2002) solve a timetabling problem with arc-based multicommodity flow formulation with track capacity and operational constraints. Heuristics based on Lagrangian relaxation is developed for a network including one direction single tracks which connect two major stations called as corridors. Real life instances in Italy with 73 stations and 500 trains are tested.

In a follow-up study, Caprara et al. (2006) add more practical constraints like station capacities, maintenances and fixed timetables for certain trains. Computational tests are performed on a railway network including up to 49 stations and 221 trains. Train timetabling on a corridor is considered by Cacchiani et al. (2008) with similar time-space network in Caprara et al. (2002) and Caprara et al. (2006). Path based formulation, which is a variation of the model in previous studies, is proposed and solved by a column generation approach. The test instances as many as 102 station and 221 trains from Italian railways are considered.

Borndorfer et al. (2005) consider an auction based approach for a similar formulation to Caprara et al. (2002). An arc-based multicommodity flow formulation with additional packing constraints formulates the optimal track allocation problem, named OPRTRA. CPLEX is employed to run the computational test on a subnetwork of long-distance railway in Germany for the Hanover-Kassel-Fulda area with 946 train requests of known origin and destination stations. Extra cases were also tested on the generated variations of the original case.

The optimal track allocation problem is followed up with a new viewpoint by Borndorfer and Schlechte (2007b). Instead of packing constraints for resolving the conflicts, they use additional configuration variables to develop a new integer programming formulation. Their LP-relaxation provides the basis for a column generation algorithm. They report computational results on the same instance from the Hanover-Kassel-Fulda area in Germany including up to 570 trains. Borndorfer and Schlechte (2007a) compare the same formulation of Borndorfer and Schlechte (2007b) with the standard formulation that rules out conflicts with packing constraints in more detail. Producing the same LP-bound, the LP-relaxation of the new formulation based on additional configuration variables is proved to be solved in polynomial time. Mesoscopic data on the same long distance railway area in the Germany with up to 570 trains are tested.

In a noncyclic timetabling problem, cyclic timetables are optimally assigned to the train routes where the timetable periods are similar. Serafini and Ukovich (1989) introduce Periodic Event Scheduling Problem (PESP) which includes defining the schedule for periodic events and therefore represents a macroscopic view of the railway network. In their study, they propose a branch and bound approach which sequentially satisfies the constraints to create a timetable.

Typical constraints in PESP, as mentioned in a study by Peeters (2003), refer to train connections, trip time between stations and headway between consecutive trains. Peeters (2003) formulates PESP by an integer programming model and shows the effectiveness of modeling cyclic railway timetabling by several practical situations in the Dutch railway network. Schrijver and Steenbeek (1994) employ a constraint programming approach with a local search heuristic to compute feasible timetables for Dutch railways.

Odijk (1996) describe a mathematical model for constructing periodic railway timetables. His cutting plane algorithm is based on constraint generation which is tested by a real-life instance considering 6 platforms and 12 stopping trains. Liebchen and Mohring (2008) suggest the potential power of the PESP to combine decisions of network planning, line planning and vehicle scheduling with the periodic timetabling. They show that one can model the extensions due to the mentioned integration as mixed integer programming and additionally extensions do not change the constraint types.

Wong et al. (2008) try to minimize the interchange waiting times of all passengers using a mixed integer programming model. They construct accurate timetables by precise adjustment of the timetable components like run times, dwell times, turnaround times and headways. Their heuristic has shown improvements on current practice of Mass Transit Railway (MTR) system in Hong Kong running six railway lines with many platform interchanges at stations.

4.1.5 Train scheduling and routing

Train routing is the problem of routing trains through railway junctions which includes assigning each train in a proposed timetable a conflict free path through the junction considering operational constraints. When routing is carried out in a station to assign the platforms to trains, the problem is called train platforming.

The computational complexity of different variations of train routing through stations is studied by Kroon et al. (1997). They prove that the train routing problem is NP-complete if there are at least three routing options for each train. Carey and Carville (2003) model the routing and scheduling trains through stations as a mathematical programming problem to minimize weighted combination of the costs and schedule deviations. Heuristic techniques are proposed to consider train operator's interest.

Caimi et al. (2005) formulate the problem as node packing model. A local search is developed to increase the time slot that a delayed train finds its assigned route still available. Carey and Crawford (2007) extend the original problem of Carey (1994) for routing and scheduling on multiple lines in opposite directions and several stations. They introduce a mixed integer programming model (MILP) to consider coast evaluation of different routes. The developed heuristic is flexible in such a way that it can be customised to help the train planner with finding and resolving conflicts.

Lusby et al. (2006) introduce a set packing model to formulate the train routing in a complex network with multiple lines. A branch and bound problem is suggested to be implemented to a small and simple example of a railway junction for illustration. Caimi et al. (2007) define compensation and condensation zones to decompose the railway network in order to create better timetables. Condensation zones are close to the stations and compensation zones connect them. Their focus is mostly on condensation zones where the traffic is dense. The problem is formulated as a conflict graph model which is solved by a fixed-point iteration heuristic.

4.1.6 Train rescheduling

Cai and Goh (1994) formulate the train scheduling problem on a single track network as an integer programming model. The approach is to respond to real-time scheduling in practice with a quick good feasible solution. They show that presented greedy heuristic finds a feasible solution very quickly, but optimal solution is unlikely to be found in polynomial time because the problem is NP-complete.

Cai et al. (1998) study an extension of the previous work by Cai and Goh (1994) with regard to rapid greedy heuristic for train scheduling for a major Asian railway. The new algorithm can take in real-world constraints easily whereas it may not be straightforward to define their mathematical formulation.

Sahin (1999) considers the conflict resolution among trains on a single track railway. He formulates the problem as a job shop scheduling problem with an objective to minimize knock on effect of the delays. A look-ahead algorithm is developed to detect and resolve conflicts in merging or crossing points.

Tornquist and Persson (2005) look at different scenarios of rescheduling to minimize the total delay due to disturbances. They introduce a two-level algorithm which is solved iteratively with the upper level solving the train meeting and overtaking by simulated annealing and tabu search and lower level defining the start and end times for each train on each section. They perform computational experiments for real-world problems in Swedish railway.

Mascis and Pacciarelli (2002) formulate the problem as a job shop scheduling problem with blocking constraints and use alternative graph for problem representation. Priority rules are used to evaluate rerouting trains and ordering is handled by a greedy heuristic to minimize maximum delay.

Tornquist and Persson (2007) propose a MILP model for dispatching trains in a railway network with several crossing and merging points. They solve the model with a commercial software package. They suggest different dispatching strategies to reduce the search space which is shown in computational tests to be fruitful to improve the quality of the solutions.

Rodriguez (2007) proposes a computerised system to solve the real-time conflict resolution problem by a constraint programming approach. The focus of the study is on the routing and scheduling train in junctions. He shows that a truncated branch and bound algorithm is capable of finding good solutions in a reasonable computation time for real-time disruptions.

4.2 Job shops

Various types of machine scheduling problems have been studied in the literature and job shop problems are one of the most difficult scheduling problems (Leung, 2004). In this section, we discuss machine scheduling, job shop models and methodologies in general, and job shop models in railway.

4.2.1 Machine scheduling

A rich variety of COPs with different combinatorial characteristics is offered in machine scheduling. Scheduling is a decision making process to allocate limited resources to tasks over time in order to optimize a given objective function. One could define resources and tasks in different ways in an organization. The resources can be a machine in a workshop or railway tracks in a rail network. Also, tasks can be defined as an operation in a workshop and traversal of a train on tracks in a rail network. Moreover, the objective function can be customised to the system's feature. Minimizing completion time of the last task in the workshop and minimizing the total delay of all trains in the rail network are two examples of adapted objective functions for a system.

With respect to uncertainty, scheduling problems are categorised into deterministic and stochastic scheduling problems. Deterministic scheduling assumes that problem parameters are fixed and each value is exactly known. Deterministic machine scheduling has received a lot of attention in the literature.

Since mid-1950s, thousands of scheduling problems and models have emerged and studied (Potts and Strusevich, 2009). Scheduling is a mature field with various theories and techniques which can be applied in different problem contexts. Obviously, there is still a wide scope to tackle interesting and challenging problems in the field.

Single machine scheduling is a scheduling model which is of great importance. Although it represents a special case and seems to be very simple, it can give us a good insight about the other environments with more complex configurations (Pinedo, 2008). Various single machine models are analyzed in the past five decades. Problems differ from each other in number of jobs, job characteristics and objective function(s). A wide variety of approaches are applied to solve single machine problems.

In parallel machine scheduling problems, there is a bank of parallel machines and a special case is when there are M identical machines in parallel. In this case, a job needs to be processed once and it can be processed on any of parallel machines. In more specific problems, a job can only be processed on a subset of the parallel machines. Parallel machine scheduling is the generalisation of single machine problem and it can be employed in decomposition of multi-stage systems.

Pinedo (2008) considers scheduling of parallel machines as a process with two steps. Firstly, it should be determined which job is assigned to which machine. Then, the sequence of the jobs allocated to each machine is defined. Various solution methods have been employed in the literature for different variants of the parallel scheduling problems.

Job shop scheduling models deal with multiple operations. There are M machines and each job has to be processed according to its predetermined route on these machines. If a job visits specific machines more than once, the job is said to have *recirculation*. The classical job shop problem minimizes makespan and it has no recirculation. Different solution methods such as branch-and-bound and constraint programming have been used to solve the job shop scheduling problem. Shifting bottleneck (SB) is an elaborated heuristic specially designed for job shop scheduling which is an adaptable approach to different constraints. (Pinedo, 2008, p.189) states "One of the most successful heuristic procedures developed for $J||C_{\max}$ is the Shifting Bottleneck heuristic."

Over the last 5 decades, different variations of the scheduling problems have emerged and a significant amount of research studies has been addressed. A three-field notation $\alpha|\beta|\gamma$ has been introduced by Graham et al. (1979) to classify and illustrate scheduling problems according to their structure. Under this notation, α defines the machine environment, β specifies job characteristics and γ indicates the optimality criterion. Assuming that we have I jobs and M machines, we list some of the notation which is used in this thesis in the following.

Job characteristics

p_{jm}: Processing time of job j on machine m.
r_j: Release date of job j.
d_j: Due date of job j.
w_j: Weight of job j.
prec: Precedence constraints.

Machine environment

1: Single machine.
P: Identical machines in parallel.
J: Job shop.

Optimality criteria

C_j: completion time of job j.
T_j: tardiness, where $T_i = \max\{0, C_j - d_j\}$.
$C_{\max}$: Makespan.
$\sum w_j T_j$: Total weighted tardiness.

In the following section, an overview of job shop models and approaches is given.

4.2.2 Job shop models and methodologies in general

In a job shop with M machines, a job visits machines in a specified order. Some machines can be visited more than once by a job whereas some machines may not be visited by a job at all. The classical job shop scheduling problem has an objective function to minimize makespan. Job shop scheduling has received a lot of attention in the literature. A few job shop scheduling problems can be solved in polynomial time. The job scheduling problem with two machine and at most two operations for each job can be solved polynomially by Jackson (1955) and Johnson (1954). Also, Brucker (1988) solves job shop problem with two jobs in polynomial time. Roy and Sussman (1964) suggest the so-called *disjunctive graph* for minimizing makespan in a job shop problem which is a useful presentation and has been used in many studies in the literature.

Branch-and-bound techniques have been widely used for minimizing the makespan in a job shop. There are two main branching rule *disjunctive arc branching* proposed by Nemeti (1964) and *active schedule generation branching* introduced by Brooks and White (1965). We refer the reader to Pinson (1995) for an overview of branch-and-bound problems applied to job shop problems. The well-known 10 jobs 10 machines 10 operations by Fisher and Thompson (1963) remained unsolved for 25 years and led to many sophisticated branch-and-bound techniques in attempts to solve the problem. McMahon and Florian (1975) introduce one of the first successful branch-and-bound methods based on one-machine decomposition application and it was the best exact method for a long time. Finally, Carlier and Pinson (1989) solve the famous 10 jobs 10 machines problem. Schrage (1970), Charlton and Death (1970) and Bratley et al. (1973) are the other early studies focused on branch-and-bound. More recent successful exact methods are listed as Applegate and Cook (1991), Carlier and Pinson (1994) and Brucker et al. (1994b).

Many heuristic methods such as simulated annealing, tabu search, genetic algorithm have been also suggested in the literature. Yamada et al. (1994) use backtracking in a simulated annealing algorithm. Studies by Laguna and Glover (1993), Taillard (1994), and Nowicki and Smutnicki (1996) are some studies on tabu search methods. Dorndorf and Pesch (1995) and Smith (1992) and Della Croce et al. (1995) have developed genetic algorithms.

A few studies look at job shop scheduling problem with total weighted tardiness as their objective functions. Heuristics based on priority rules are implemented by Vepsalainen and Morton (1987). Pinedo and Singer (1999) propose a shifting bottleneck heuristic for a job shop problem to minimize total weighted tardiness. Later, Kreipl (2000) suggests a large step random walk to minimize total weighted tardiness in a job shop problem. De Bontridder (2005) develops a tabu search to minimize total weighted tardiness in a job shop with generalized precedence relationships. A genetic algorithm is presented by Essafi et al. (2008) which uses an iterated local search to improve the quality of chromosomes. Bulbul (2011) applies a hybrid shifting bottleneck heuristic which replaces the re-optimization step by a tabu search. A broader overview of the scheduling problems is provided by Potts and Strusevich (2009) which discuss the main topics of scheduling research in the past fifty years and highlight the main contributions to shape the field.

4.2.2.1 Job shop problems with shifting bottleneck procedures

One of the most well-known heuristics is shifting bottleneck procedure by Adams et al. (1988). It is regarded as a decomposition approach due to the fact that it decomposes a multiple machine problem to single machine problems and uses Calier's algorithm (Carlier, 1982) to solve its optimally. More refined version of the Carlier algorithm is introduced in studies by Dauzre-Prs and Lasserre (1993), Dauzre-Prs and Lasserre (1994) and Balas et al. (1995) who also embed it into a modified version of the shifting bottleneck.

Balas et al. (1995) introduce delayed precedence constraints in the single machine problem arising in the shifting bottleneck procedure. Balas and Vazacopoulos (1998) have improved the shifting bottleneck procedure by embedding a guided local search into it. Although the developed shifting bottleneck is computationally more expensive, it can improve the quality of the solutions significantly. Balas et al. (2008) deal with the a variant of job shop problem in the presence of release dates, deadlines and sequence-dependent setup times. They modify the shifting bottleneck to solve the single machine problem as a Travelling Salesman Problem with time windows. The variant of the job shop problem which is discussed in Balas et al. (2008) study is very common in semiconductor industry (Ovacik and Uzsoy, 1997).

Ovacik and Uzsoy (1997) present the application of shifting bottleneck and the other decomposition methods to large scale job shop problems with various objective functions such as the makespan and the maximum lateness. Ivens and Lambrecht (1996) extend the disjunctive graph formulation and the shifting bottleneck procedure to deal with real-life applications by introducing due dates, release dates, assembly structures, split structures, overlapping operations, setup times, transportation times, parallel machines and beginning inventory in the job shop scheduling problem. Schutten (1998) also extends the shifting bottleneck procedure to accommodate practical features such as transportation times, simultaneous resource requirements, setup times, and other important characteristics.

Pinedo and Singer (1999) develop a disjunctive graph formulation and employ shifting bottleneck for minimizing the total weighted tardiness in a job shop. The study by Pinedo and Singer (1999) has been later extended by Mason et al. (2002) in order to formulate a disjunctive graph and propose a modified shifting bottleneck heuristic for a semiconductor wafer fabrication facility. The problem is a flexible job shop with sequence-dependent setups, different arrival times of jobs, and re-entrant or re-circulating product flow through a number of different tool groups consisting of multiple identical machines in a given work center.

Mason et al. (2005) offer a complex scheduling problem for the semiconductor wafer fabrication facility which is compromised of batching machines, parallel machines, machines with sequence dependent set ups and recirculating product flow. A mixed integer program to minimize total weighted tardiness and a modified shifting bottleneck heuristic are developed.

Pfund et al. (2008) also formulate a complex job shop model for semiconductor wafer fabrication process. They employ a desirability function for a multi-criteria optimization problem of makespan, cycle time and total weighted tardiness which is implemented in two different levels of subproblem solution procedure and the machine criticality measure level. They build on study by Mason et al. (2002) which considers minimization of the total weighted tardiness and offer a modified shifting bottleneck procedure for the multi-criteria optimization using the mentioned desirability function.

Monch and Drieel (2005) develop a modified shifting bottleneck for a complex job shop problem for semiconductor wafer facilities which contains parallel batching machines, machines with sequence-dependent set up times and reentrant process flows. Their proposed shifting bottleneck heuristic minimizes total weighted tardiness in a two-layer hierarchical approach which decomposes the overall scheduling problem. The upper layer considers the aggregated model which determines start dates and due dates for the jobs in each work area defined as a set of parallel machine groups. The lower layer employs determined start dates and due dates in the shifting bottleneck heuristics applied on jobs in each single work area. They assess the performance of the heuristic in comparison with a simulation model of a dynamic job shop environment.

Similar to Monch and Drieel (2005), Monch et al. (2007) minimize the total weighted tardiness, but they extend the previous study where shifting bottleneck decomposes the overall scheduling into scheduling problems for single tool groups. In the study by Monch and Drieel (2005), only subproblem solution procedures have been developed based on dispatching rules but in the extended study, Monch et al. (2007) apply more sophisticated subproblem solution procedures like genetic algorithms for parallel machine scheduling. In comparisons with the shifting bottleneck procedure, simulation experiments in a dynamic job shop environment indicate that using genetic algorithm result in improved results compared to subproblem solution procedures based on dispatching rules.

Bulbul (2011) minimizes total weighted tardiness in a job shop scheduling problem. He suggests a hybrid shifting bottleneck-tabu search algorithm to solve the problem. A tabu search algorithm is embedded into shifting bottleneck heuristic which replaces the re-optimization step. Therefore, the shifting bottleneck heuristic has a long-term memory which is helpful in diversifying the local search. The tabu search is applied to both feasible full schedules and to partial schedules where some machines are assumed to have infinite capacity. Computational performance of the algorithm is tested on benchmark instances from the literature.

Liu and Kozan (2009) formulate train scheduling problem as a blocking parallel machine job shop scheduling problem. In order to model the problem, they adapt the alternative graph by Mascis and Pacciarelli (2002) which is an extension of the classical disjunctive graph. The problem is solved by an extended shifting bottleneck procedure without considering blocking conditions. A constructive heuristic algorithm called Feasibility satisfaction procedure (FSP) algorithm is developed to find feasible solutions for the blocking parallel machine job shop scheduling problem. Suggested algorithm is implemented on real-world data from Queensland Rail for freight trains. Some sensitivity analysis are performed to consider train length, upgrading track sections, increasing train speed and changing the bottleneck sections which shows the proposed method is promising for solving real-life train scheduling problems.

Liu and Kozan (2012) develop a hybrid shifting bottleneck procedure combined with Tabu search to solve the parallel machine job shop scheduling problem. The conventional shifting bottleneck has been improved with respect to four novel features which are given in the following. Firstly, a new algorithm called topological-sequence algorithm is developed to decompose the parallel machine job shop problem into a set of single and parallel machine scheduling subproblems. Secondly, a modified Carlier algorithm is developed to solve the single machine subproblem. Thirdly, to solve the parallel machine subproblems, the Jackson rule is extended. Finally, a Tabu search algorithm is embedded into the shifting bottleneck heuristic in order to optimize the single and parallel machine subproblems.

4.2.2.2 Job shop problems with alternative arcs

The concept of an alternative graph was first introduced by Mascis and Pacciarelli (2000). Mascis and Pacciarelli (2000) and Mascis and Pacciarelli (2002) look at different types of job shop problems such as the ideal (classical) job shop, the blocking job shop with and without swaps, and the no-wait job shop. They formulated these problems with an alternative graph which is a generalization of disjunctive graph. They develop three fast dispatching heuristics for the mentioned job shop problems. Computational experiments are reported for a large number of benchmark problems. They also develop a branch and bound algorithm and tested the performance of it on eighteen smaller (10×10) instances.

Meloni et al. (2004) deal with ideal, blocking and no-wait job shop which is formulated as an alternative graph. They present a rollout metaheuristic to solve this problem through a constructive procedure which extends iteratively a partial schedule shown by a partial selection of alternative arcs to a complete schedule. At each extension step, a scoring function candidate evaluates all candidate arcs and the arc with the best score is added to the partial selection. Numerical results show promising results for the eighteen (10×10) benchmark instances which were solved optimally by Mascis and Pacciarelli (2002), representing improvements compared to dispatching methods in Mascis and Pacciarelli (2002).

A recent stream of research on train scheduling and routing studies focused on the alternative graph formulation of Mascis and Pacciarelli (2002). Flamini and Pacciarelli (2008) address a train scheduling problem in real-time looking at routing trains through an underground rail terminus. They model a bicriteria job shop scheduling problem in which punctuality and regularity of train service are optimized. Optimizing earliness/tardiness and train headways are the two objective functions in the lexicographical order. A heuristic is developed which solves the problem in two steps. Firstly, a feasible solution with regard to the first objective function is built. Then, second objective function is optimized by deteriorating the first objective function.

Another study which formulates the train scheduling problem by using alternative graph is D'Ariano et al. (2007). They model real-time train traffic control which is faced by railway infrastructure managers when train operations are perturbed. A new conflict-free timetable of recomputed arrival and departure times should be produced so that the deviation form the original plan is minimized. The problem is modeled as a huge job shop scheduling problem for trains with fixed routes. A branch and bound algorithm is proposed which uses implications rules to speed up the computation. Computational experiments report on the Dutch railway around Schipol International airport and show optimal or near-optimal solutions can be found in a short computation time.

In a followup paper by D'Ariano et al. (2008), the implementation of a real-time traffic management system, called ROMA (Railway traffic Optimization by Means of Alternative graphs) is described which supports controllers in managing disturbances in daily basis. The branch and bound algorithm is implemented for sequencing train movements and a local search optimizes train rerouting. The compound approach of the branch and bound and local search algorithm is implemented iteratively to compute optimal sequences for given train route and then improving this solution by rerouting some trains. Computational experiments are carried on practical size instances of the Dutch dispatching area between Utrecht and Den Bosch. Instances include different types of disturbances with multiple delayed trains and different blocked tracks in the network. The results show promising potential of ROMA that can significantly reduce delays by rerouting and rescheduling train movements.

Another study based on the alternative graph formulation is the research by Corman et al. (2009) to analyze a policy, called green wave (GW), which is a one of the several new traffic policies in the management of complex railway networks in respond to the expected growth of transport demand in the next years. According to this policy, trains wait at the stations in order to avoid speed profile modifications in open corridors which is expected to be very effective when corridors are the bottlenecks of the network. Two models are introduced to formulate the train scheduling problem with the GW policy and with an alternative policy which is letting trains change their speed profile in open corridors. Computational experiment is carried out for two practical dispatching areas of the Dutch railway network, one is the line from Utrecht to Den Bosch and the other is Dutch railway around Schipol International airport.

Corman et al. (2010) improve on the study by D'Ariano et al. (2008) by including effective rescheduling algorithms and a tabu search for local rerouting strategies. Problem formulation is based on the alternative graph by Mascis and Pacciarelli (2002). A fast heuristic and a truncated branch and bound are implemented alternatively to produce train schedules in a short computation time. In addition, they develop new routing neighborhood structures for more effective routings. The computational experiments are reported on the same Dutch dispatching area between Utrecht and Den Bosch including complex disturbances with multiple delayed trains and blocked tracks. Tabu Search results are compared with branch and bound of D'Ariano et al. (2007) and local search algorithm of D'Ariano et al. (2008). The new tabu search algorithms find optimal solutions for small instances. For large instances, the new algorithm achieves significantly better results within a shorter computation time compared to previous version of ROMA.

Liu and Kozan (2009) use also the alternative graph for the formulation of the train scheduling problem. They model the problem as a blocking parallel machine job shop scheduling (BPMJSS) problem. An improved shifting bottleneck procedure is developed to solve the proposed BPMJSS problem (See Subsection 4.2.2.1 for more details of their study).

Liu and Kozan (2011) investigate train scheduling problem with priorities and model it as a No-Wait Blocking Parallel-Machine Job-Shop Scheduling (NWBP-MJSS) problem. They formulate the NWBPMJSS problem by integer programming and analyse it based on an alternative graph model. A novel constructive algorithm is suggested to create train timetables by defining the order of trains. The constructive algorithm includes several recursively implemented sub-algorithms called Best-Starting-Time-Determination Procedure, Blocking-Time-Determination Procedure, Conflict-Checking Procedure, Conflict-Eliminating Procedure, Tune-Up Procedure, and Fine-Tune Procedure. Solving these subalgorithms recursively can guarantee feasibility by satisfying the blocking, no-wait, deadlock-free, and conflict-free constraints. In order to find the good train schedule efficiently, a two-stage hybrid heuristic algorithm is developed by combining the constructive algorithm and a local-search heuristic called the Best Insertion Heuristic. The computational experiments represent that the proposed approach is promising because it can serve as a generic and fundamental toolbox for identifying, analysing, modeling, and solving real-life scheduling problems.

Groeflin and Klinkert (2009) propose a generalization of the blocking job shop, called Generalized Blocking Job Shop, which considers transfer operations between machines and sequence-dependent setup times. The developed disjunctive graph is transformed into a more compact formulation via alternative graphs by Mascis and Pacciarelli (2000). As there is no easy mechanism to generate feasible neighbor solutions like classical job shop problems, they develop a neighborhood for local search by defining concept of closure and a key result on short cycles. A tabu search is devised based on this neighborhood. Computational experiments show that the tabu search algorithm improves most of the benchmark results in the literature.

4.2.2.3 Job shop problems with parallel machines

Job shop problems with parallel machines have been studied in the literature with different names such as parallel machine job shop scheduling, generalized, flexible or complex job shop scheduling problem.

Sadeh et al. (1995) address a job shop scheduling problem where some operations have to be scheduled within time windows such as earliest/latest possible start time windows. In order to solve this well-known NP-complete Constraint Satisfaction Problem (CSP), they combine a new look-back scheme with consistency enforcing techniques and variable/value ordering heuristics which helps the search procedure recover from so-called dead-end search states.

Dauzre-Prs and Paulli (1997) investigate an important extension of the classical job-shop scheduling (JSS) problem where the same operation can be performed on more than one machine such that the makespan of a set of jobs is minimized. By defining an extended version of the disjunctive graph model, they can define a neighborhood structure for the problem. An integrated tabu search (TS) algorithm where there is no distinction between re-assigning or re-sequencing an operation is devised and computational experiments are provided.

Chen and Luh (2003) introduce a new Lagrangian relaxation approach for the parallel-machine job shop scheduling problem where operation precedence constraints rather than machine capacity constraints are relaxed. Then, they decompose the relaxed problem into single or parallel machine scheduling subproblems which are NP-complete in general. The so-called surrogate subgradient method is used to solve the dual subproblems which allows approximate optimisation of the subproblems. Computational experiments show that the machine decomposition-based LR approach is efficient, particularly for large problems with long time horizons.

Alvarez-Valdes et al. (2005) address the design and implementation of a scheduling system in a glass factory. They model the problem as a flexible JSS problem with additional special characteristics. They minimize a non-regular criterion defined by the user and based on due dates. It can be devised to determine delivery dates for new customer orders by taking into account current machine workloads, or to schedule a set of orders by trying to meet given customer due dates. The method can produce approximate solutions in very short computation times.

Xia and Wu (2005) consider a multi-objective flexible job shop scheduling problem. As it is quite difficult to achieve an optimal solution to this problem with traditional optimization approaches, they propose a particle swarm optimization algorithm which is an evolutionary computation technique. Their approach has high search efficiency by combining local search (by self experience) and global search (by neighbouring experience). They develop a hybrid approach consisting of a Simulated Annealing (SA) algorithm and the particle swarm optimisation algorithm. The computational results show the viability and effectiveness of the proposed hybrid approach for the multi-objective flexible job shop scheduling problem.

Fattahi et al. (2007) propose a mathematical model and heuristic approaches for flexible job shop scheduling problems. A mathematical model is used to obtain optimal solution for small size problems. In order to solve the real size problems with more than two jobs, two types of heuristic approaches are developed, hierarchical approaches and integrated approaches. Hierarchical approaches consider the assignment of operations to machines and sequencing of operations on machines separately; Whereas, assignment and sequencing are not differentiated in integrated approaches. The numerical results validate the quality of the suggested algorithms and show that the hierarchical algorithms have better performance than the integrated ones.

Gao et al. (2008) consider the flexible job shop scheduling problem with three objectives; min makespan, min maximal machine workload and min total workload. A hybrid genetic algorithm (GA) for the problem is suggested which uses two vectors to represent solutions. To adapt to the special chromosome structure and the characteristics of the problem, advanced crossover and mutation operators are used. Individuals of GA are first improved by a variable neighborhood descent (VND) which includes two local search procedures. The performance of the proposed GA approach is tested by extensive computational experiments on 181 benchmark problems.

Job shop problems with parallel machines are also implemented in the railway studies. We refer the reader to 4.2.2.1 for more details on the studies by Ivens and Lambrecht (1996), Mason et al. (2002), Mason et al. (2005), Pfund et al. (2008), Monch and Drieel (2005), Monch et al. (2007), Liu and Kozan (2009) and Liu and Kozan (2012). The study by Liu and Kozan (2011) is also a relevant study which is mentioned in Subsection 4.2.2.2.

4.2.3 Job shop models in railway

After the pioneering publication of Szpigel (1973), formulating the train scheduling problem as a job shop scheduling problem offered a promising new research direction. However, there have been several job shop scheduling approaches such as mathematical programming techniques by Szpigel (1973) and Sahin (1999), constraint programming approaches by Oliveira and Smith (2000), Oliveira (2001) and Rodriguez (2007), and the alternative graph formulation by D'Ariano et al. (2007), D'Ariano et al. (2008), Corman et al. (2009), Corman et al. (2010) Liu and Kozan (2009) and Liu and Kozan (2011).

Szpigel (1973) formulates a train scheduling problem on a single track network in eastern Brazil. The problem is formulated as a job shop scheduling problem with additional constraints. He defines the the best crossing and overtaking locations for given routes and departure times of the trains on the mentioned network. A branch and bound algorithm is introduced and computational experiments are reported on for five track sections and ten trains.

Sahin (1999) considers the inter-train conflict resolutions on a single track railway. A job shop scheduling formulation is developed for rescheduling trains by modifying the current meet/pass plans in conflicting situations. The objective function is to minimize the knock-on effect of the delays. A heuristic is developed to resolve the conflicts in the order that they appear. The look-ahead property of the algorithm helps to compute the arrival time of all the other trains if a train is delayed. Thus, the algorithm chooses the train resulting in the least amount of delay.

Oliveira and Smith (2000) and Oliveira (2001) study train scheduling problem along a single track network for a disruption recovery purpose. They use the same formulation as Szpigel (1973) excluding ordering constraints in order to minimize delay. Delays are only introduced at the beginning of the train trip. Conflicts are resolved in chronological order through a constraint programming approach. Some practical constraints are included such as forcing two trains residing in the same station for a certain amount of dwell time, and allowing the same train to do multiple itineraries. They solve real-life problems gathered form literature to carry out computational experiments.

Rodriguez (2007) uses also the similarities between job shop scheduling and the problem of scheduling and routing trains through a junction. They propose that the model can be integrated to a decision support system which is used by operators in order to help them with rerouting and reordering trains to avoid conflicts. Test instances are based on the Pierrefitte-Gonesse railway junction in Paris and include between 6 and 24 trains. Experimental results show a significant improvement in performance in a reasonable amount of time.

D'Ariano et al. (2007) study the train scheduling in real-time as a huge job shop scheduling problem with no-store constraints. In case of a disruption, a new conflict-free timetable should be created such that the deviation from the original plan is minimized. The alternative graph formulation is used to model the problem. A branch and bound algorithm is developed which uses some implication rules to make the computation faster. The experimental results are based on the Dutch railway around Schipol International airport and the truncated version of the algorithm can find optimal or near-optimal solutions in a short computation time.

D'Ariano et al. (2008) is a followup study which focuses on the implementation of a real-time traffic management system, called ROMA (Railway traffic Optimization by Means of Alternative graphs) to help controllers in handling disruptions. The problem is modeled as a job shop scheduling with additional constraints. A compound problem of sequencing and routing trains are solved iteratively. The branch and bound is utilized to find the optimal sequence of the trains for given routes and a heuristic is implemented to improve the solution by locally rerouting some trains. Computational experiments based on the Dutch dispatching area between Utrecht and Den Bosch includes instances with different types of disturbances in terms of train delays and blocked tracks. The system shows high potential to be viable as a support tool to effectively improve punctuality.

Corman et al. (2009) is another study based on the job shop scheduling formulation and the alternative graph formulation of it. This study analyses the concept of the green wave (GW) policy which is a traffic policy in the management of complex railway networks. It allows trains to wait at the stations so that they do not change their running times in open corridors. GW policy is compared with an alternate policy which allows speed profile changes in open corridors through some computational experiments on two dispatching areas of the Dutch railway network, the line from Utrecht to Den Bosch and Dutch railway around Schipol International airport. The GW policy is shown to be the most effective in instances with short corridors, small differences in train speeds and spare capacity available at the stations.

Corman et al. (2010) offer some improvements on the study by D'Ariano et al. (2008) through advanced strategies to solve the compound train rerouting and rescheduling problem as a job shop problem. A novel tabu search with effective neighborhood structures and search strategies is devised to minimize consecutive delays. The new tabu search algorithm shows promising performance improvements compared to previous version of the ROMA in terms of solution quality and computation time. Test instances are from the same dispatching area between Utrecht and Den Bosch considering complex disturbances with multiple delayed trains and blocked tracks.

The train scheduling problem is formulated as a blocking parallel machine job shop scheduling problem in the study by Liu and Kozan (2009). They propose an extended shifting bottleneck procedure which does not consider blocking conditions. The feasible solutions for the blocking parallel machine job shop scheduling problem are found by a constructive heuristic algorithm called Feasibility Satisfaction Procedure (FSP) algorithm. The suggested method is promising to be implemented on real-life train scheduling problems. computational experiments are represented for freight trains in Queensland Rail. Sensitivity analysis are also performed to consider train length, upgrading track sections, increasing train speed and changing the bottleneck sections.

Train scheduling problem with priorities is modeled as a No-Wait Blocking Parallel-Machine Job-Shop Scheduling (NWBPMJSS) problem by Liu and Kozan (2011). An integer programming model and an alternative graph formulation are offered for the NWBPMJSS problem. They propose an innovative generic constructive algorithm to create the feasible train timetable by recursively implementing several subalgorithms called Best-Starting-Time-Determination Procedure, Blocking-Time-Determination Procedure, Conflict-Checking Procedure, Conflict-Eliminating Procedure, Tune-Up Procedure, and Fine-Tune Procedure. Applying these subalgorithms can guarantee feasibility by satisfying the blocking, no-wait, deadlock-free, and conflict-free constraints. A two-stage hybrid heuristic algorithm is developed by combining the constructive algorithm and a local-search heuristic called the Best Insertion Heuristic helps to find good train schedule efficiently. Extensive computational experiments show that the proposed method is promising as it can be viable as a generic and fundamental toolbox for identifying, analysing, modeling, and solving real-life scheduling problems.

In summary, there are two main lines of research with regard to the complexity of the railway infrastructure. In the first category, Szpigel (1973), Sahin (1999), Oliveira and Smith (2000) and Oliveira (2001) address a single line railway with single and multiple track segments. More realistic networks are considered in the second category of studies. Rodriguez (2007) schedules trains in a terminal station, whereas D'Ariano et al. (2007), D'Ariano et al. (2008), Corman et al. (2009) and Corman et al. (2010) provide solutions for a dispatching area of a railway network with passengers and freight. Further, Liu and Kozan (2009) and Liu and Kozan (2011) investigate a case study of a railway network for the transport of coal.

4.3 Complexity of scheduling problems

In this section, the complexity aspects of the problems and subproblems which appear in this study is discussed. The train scheduling and rescheduling problem which appears in Chapter 5 is formulated as a modified blocking job shop scheduling problem with additional constraints. In the case study, the processing times of jobs are the same on the same machine if there is no disruption. Otherwise, processing time of jobs associated to disrupted trains differ form the other jobs operated on the same machine (the other trains running on the same block). Each job has a non-negative release time and due date. The objective function is to minimize the total weighted tardiness for the proposed generic problem; However, weights of jobs are considered to be equal to one in the case study.

According to Garey and Johnson (1979) and Ullman (1975) job shop problem is known to be NP-hard. Thus, the train scheduling problem suggested in this study is NP-hard. Timkovsky (1985) proves that $J2|r_j, p_{ij} = 1| \sum C_j$ is NP-hard if all operations are no-wait. Therefore, the proposed train scheduling problem is NP-hard, even if there are equal-length operations on each machine.

More results about the particular cases of job-shop problems with blocking and no-wait are listed in the following. If all operations are ideal, Kubiak et al. (1996) show that the problem is polynomially solvable. The three machine flow shop problem where all operations to be operated on the second machine have zero processing time, is strongly NP-hard in the blocking case according to Papadimitriou and Kanellakis (1980). French (1982) proves the same three machine flow shop problem is polynomially solvable in the ideal case.

The train scheduling and rescheduling problem is solved by a novel modified shifting bottleneck procedure in Subsection 5.3.5. The shifting bottleneck decomposes the problem into subproblems. Generically, a single machine problem $1|prec; r_j| \sum w_j T_j$ can be formulated and then solved. If arbitrary processing times are assumed, there is no hope to find an algorithm to solve the problem in polynomial time. Because it is a generalization of $1|| \sum T_j$ which is shown by Du and Leung (1990) to be NP-hard.

In the case study, operation time of the jobs on a machines is the same when there is no disruption; This means that we have equal processing times $p_{jm} = p_m$ in the problem. Moreover, we have assumed that all jobs have equal weights. Therefore, the objective function can be defined as $\sum T_j$ with all weights equal to one. $1|r_j; p_j = p| \sum T_j$ can be solved in polynomial time according to Baptiste

(2000). The presence of alternative constraints in the proposed problem, makes the problem $1|prec; p_j = p; r_j|\sum T_j$ very difficult to solve. According to Brucker and Knust (1994), this problem is a generalization of $1|chains; p_j = 1|\sum T_j$ which is proved to be strongly NP-hard by Leung and Young (1990).

Therefore, the subproblem $1|prec; p_j = p; r_j|\sum T_j$ in our case study is strongly NP-hard. It should be noted that the proposed modified shifting bottleneck procedure in Subsection 5.3.5, is developed for the generic subproblem $1|prec; r_j|\sum w_j T_j$ which is strongly NP-hard according to the discussion mentioned above.

It is worthwhile to note that precedence constraints appear in the $1|prec; p_j = p; r_j|\sum T_j$ subproblem. It may seem at first that in the case that headways are large, alternative arcs can be redundant and there is no precednece constraints in the graph. Alternative arcs only exist for following trains and they become active when there is a disruption to keep following trains 3 blocks far from each other. This is due to the fact that during the disruption running times are longer and it takes longer to traverse a block. In this case, headway is smaller than the sum of the operations times of three consecutive blocks. Alternative arcs are only redundant when there is no disruption. There are another set of arcs which are called static implications which are only fixed for the following trains running on common blocks. After fixing the static implication arcs, the precedence constraints are imposed among the operations which are performed on the same machine. We fix the arcs among the operations corresponding to both following and opposite trains running on common blocks and call these arcs simple implied arcs. Reader is referred to Chapter 5 for more information about alternative arcs, static implication arcs and simple implied arcs.

Train routing and rerouting problem is formulated as a modified parallel machine job shop scheduling problem which is presented in Chapter 6. An novel extended shifting bottleneck is proposed to solve the train routing and rerotuing problem in Subsection 6.3.2. The shifting bottleneck decomposes the train routing and rerotuing problem into several single machine (SM) and parallel machine (PM) subproblems.

The arising SM problem is minimizing total weighted tardiness on a single machine with release times $1|r_j|\sum w_j T_j$. This problem is NP-hard as it is the generalization of $1||\sum w_jT_j$ which is proven to be strongly NP-hard by Lawler (1977). However, in the case study, we have assumed similar priority for all trains. That is, the weights are all set equal to 1. Moreover, the processing times are the same on a single machine. Therefore, the arising model in our case study is $1|p_j = p; r_j|\sum T_j$ which can be solved in polynomial time (Baptiste, 2000). It should be noted that in what follows, the solution method is represented for the generic problem $1|r_j|\sum w_jT_j$.

The PM subproblem is an instance of $P|r_j|\sum w_jT_j$ which is a generalization of $1||\sum w_jT_j$. Lawler (1977) has shown that $1||\sum w_jT_j$ is strongly NP-hard. Hence, $P|r_j|\sum w_jT_j$ is NP-hard. In the case study, trains have equal priorities and equal running times on track sections. So the model in the case study is minimizing total tardiness on identical parallel machines with common processing times and release times $P|p_j = p; r_j|\sum T_j$. This problem can be solved in polynomial time (Brucker and Kravchenko, 2005). However, we propose a solution method for the generic problem $P|r_j|\sum w_jT_j$ in our study due to main two reasons. Firstly, in our case study, we look at disruptions and develop a suitable method to deal with disruptions. In all test instances other than the timetable, we solve the problem where the processing times are not equal for the jobs processed on the same machine. Therefore, we generally solve the problem $P|r_j|\sum T_j$ which is NP-hard; Because, it is the generalization of the $1|r_j|L_{\max}$ which is shown by Lenstra (1977) to be NP-hard. Secondly, a solution for the generic problem is developed so that it can be viable to solve more general problems.

Chapter 5

Train scheduling and disruption management

Today, the railway network system is a major part of the transportation infrastructure in many countries. Due to the increasing volume of railway traffic and the high expense of construction or modification of infrastructure, train scheduling is employed to ensure the best usage of the existing capacity. There has been increasing interest in the train scheduling problem recently and job shop scheduling has been exploited in some studies after the seminal publication of Szpigel (1973).

This chapter addresses scheduling and disruption management which is intended to optimize train traffic on a rail network with a detailed topology. Our novel approach to model and solve the problem makes use of job shop scheduling concepts. This chapter is organized as follows. We introduce the train scheduling and rescheduling in the case of delays briefly in Section 5.1. A mathematical formulation for the problem is suggested in Section 5.2, which shows computational advantages for real-world instances. We also extend a graph representation of job shop scheduling problem to formulate additional operational and safety constraints for train scheduling problem. Consequently, a sophisticated algorithm is offered in Section 5.3 to solve train scheduling problem based on a special heuristic for job shop scheduling problem. Different variations of the heuristic are developed and analyzed through computational results in Section 5.4 which shows their strength and viability for practical use. Finally, we conclude the chapter with a summary and final remarks in Section 5.5.

This chapter summarizes the study and it is an edited version of the papers which has been partially presented in the following conferences Khosravi et al. (2010), Khosravi et al. (2011a), Khosravi et al. (2011b), Khosravi et al. (2012a), Khosravi et al. (2012b).

5.1 Introduction

This study is focused on train scheduling problem at the micro level including detailed information about the tracks and train movements. As it is already discussed in Chapter 3, we base our experiments on a bottleneck area in the South East of the UK where the network is very congested and has a complicated structure with regard to junctions and stations.

The aim is to define entry times and the order of trains on blocks for predetermined routes to get from origin to destination. Hence, a schedule is created by timing and ordering trains at junctions and stations in order to improve the utilization of the existing infrastructure and capacity. In order to do this we use the analogy between train scheduling and job shop scheduling.

In the case of disruptions, which cannot be avoided in operational level, the same scheduling technique can be applied for real-time traffic management when the solution method is fast enough. The produced schedule must be conflict-free, where a conflict is defined as two or more trains assigned to the same block.

It is also important to consider the potential for delay propagation in a rail network, which results from the high interdependency of the trains. Thus, the objective is to minimize the total weighted tardiness to avoid delay propagation subject to a set of operational and safety constraints. More details about the train scheduling formulation and modeling the objective function and constraints are given in Section 5.2.

5.2 Mathematical formulation

In this study, we make use of similarity between train scheduling problem and the well-known job shop scheduling problem. Job shop scheduling assigns jobs to machines in a way that a machine can process only one job at a time. Likewise, a block can be occupied by only one train at a time according to the line blocking which is a safety principle for train movement. Thus, a train traversing a block is analogous to a job being processed on a machine, and is referred to as an operation.

The objective function is to minimize the total delay of train with priorities which is translated to minimizing the total weighted tardiness of the jobs. This differs from the classical job shop problem with makespan as the objective function. Thus, the conventional job shop model needs to be modified in terms of the objective function.

The conventional job shop model is also extended with regard to the constraints in order to consider railway operational and safety constraints. In this section, we firstly consider two main railway constraints including running time constraints and headway constraints. Then, a new set of constraints is added to make the model more realistic regarding the characteristics of the network signaling system.

In the following, it is shown how the model can be formulated by some configurations and incorporating new constraints in the conventional job shop scheduling model. In Subsection 5.2.1, we develop a mathematical programming model of the problem and Subsection 5.2.2 focuses on a graph representation and formulation of the modified job shop scheduling problem.

5.2.1 MILP model

Several programming formulations for the classical job shop scheduling problem $J||C_{\max}$ are introduced in the literature. There is a couple of integer programming formulations for the job shop problem. However, the well-known *disjunctive programming formulation* is the most commonly used formulation. In the classical job shop scheduling problem, a set $\mathcal{I}$ of jobs i for $i = 1, \dots, I$ should be processed on a set of machines denoted by $\mathcal{M}$ for machines $m = 1, \dots, M$. Each job i has a sequence of machines $(m_{i1}, \ldots, m_{i,l_i})$ that needs to be visited in a special order. No preemption is allowed so each job needs to be finished without interruption. Each job i becomes available at time zero and requires a fixed processing time p_{im} to finish. The objective is to find a sequence of jobs on each machine in order to minimize the makespan $C_{\max} = \max\limits_{i=1,\ldots,I}\{C_i\}$, where C_i denotes the completion time of the last operation of i $(i = 1, \ldots, I)$.

A schedule is a set of starting times t_{im} to satisfy the *conjunctive constraints*:

$$t_{i,m_{i,k}} - t_{i,m_{i,k-1}} \geq p_{i,m_{i,k-1}} \qquad i \in \mathcal{I},\ k = 2, \ldots, l_i \tag{5.1}$$

and the pairs of *disjunctive constraints*:

$$t_{jm} - t_{im} \geq p_{im} \quad \text{or} \quad t_{im} - t_{jm} \geq p_{jm} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{5.2}$$

where $\mathcal{O}$ is the set of operations defined by indices (i, m), for $i \in \mathcal{I}$ and $m \in \mathcal{M}$.

Thus, a mixed integer programming formulation can be presented by introducing a binary variable x_{ijm} for each disjunction which determines whether operation i can start before j on machine m, and a very large positive number $\bar{M}$.

$$\begin{aligned}
&\text{Minimize} \quad C_{\max} && && (5.3)\\
&\text{subject to} && && \\
&C_{\max} - t_{i,m_{i,l_i}} \geq p_{i,m_{i,l_i}} && i \in \mathcal{I} && (5.4)\\
&t_{i,m_{i,k}} - t_{i,m_{i,k-1}} \geq p_{i,m_{i,k-1}} && i \in \mathcal{I},\ k = 2, \ldots, l_i && (5.5)\\
&t_{jm} - t_{im} + \bar{M}(1 - x_{ijm}) \geq p_{im} && (i,m),\ (j,m) \in \mathcal{O} && (5.6)\\
&t_{im} - t_{jm} + \bar{M}(1 - x_{jim}) \geq p_{jm} && (i,m),\ (j,m) \in \mathcal{O} && (5.7)\\
&x_{ijm} + x_{jim} = 1 && (i,m),\ (j,m) \in \mathcal{O} && (5.8)\\
&x_{ijm} \in \{0, 1\} && (i,m),\ (j,m) \in \mathcal{O} && (5.9)\\
&t_{i,m_{i,1}} \geq 0 && i \in \mathcal{I} && (5.10)
\end{aligned}$$

Constraints (5.4) is to define the makespan in terms of the jobs starting times and processing times. The so-called conjunctive constraints (5.5) make sure that the starting time of the operation $(i, m_{i,k})$ is not before the completion of the operation

$(i, m_{i,k-1})$. Constraints (5.6) and (5.7) are known as disjunctive constraints and they guarantee that there is an order between different jobs to be processed on a machine. Therefore, this formulation is referred to as *disjunctive programming formulation*.

We use this formulation to define train scheduling problem. The fixed block signaling is the technology used for safety in the main lines in the UK network. So it is the basic principle that a block is protected by the signals. According to safety principles, only one train can enter and run on a block at a time. The model should include the running times and dwell times of the train which can be defined by conjunctive constraints.

Trains need to be away from each other as much as the headway value. Thus, headway constraints are very similar to disjunctive constraints, but they need to be slightly modified to be more realistic for both scheduling and rescheduling purposes. Therefore, the headway value is defined to be equal to the maximum value between scheduled headway in the timetable and a longer running time of a train on a block in case of a perturbation.

Another set of constraints is added to consider network signaling which can be customized for different types of signaling systems. These constraints are called *blocking constraints* in the flow shop scheduling literature as a job blocks a machine due to no buffer capacity between consecutive machines until the next machine becomes available for it to move further in its sequence. In another word, blocking forces jobs to remain on a machine after their completion until the next machine becomes available. This assumption is very beneficial as it is an important issue in many real-life scheduling problems (See Hall and Sriskandarajah (1996) for a survey of machine scheduling problems with blocking).

Mascis and Pacciarelli (2002) define the so-called *alternative constraints* in a generalization of the job shop scheduling formulation. The alternative constraints are a modification of the disjunctive constraints to incorporate blocking characteristic in the job shop model. In this study, we modify alternative constraints to define the restrictions imposed by the signaling system. We consider four-aspect signaling which means a signal is red for stop (danger), yellow for approach (caution), double yellow for advance approach (preliminary caution) and green for clear.

However, we define generic alternative constraints and they can be adapted to the other fixed block systems.

Given predetermined routes from a given origin to a given destination, a schedule determines starting times of trains entering each block and the order of trains on each block. We employ the approach to the currently operating timetable and we optimize the train sequence on each block in off-line and on-line traffic management. Considering the analogy between train scheduling and job shop scheduling, the following notation is used for parameters and decision variables in the mathematical programming formulation for the train scheduling and rescheduling problem.

$\mathcal{I}$:	set of jobs/trains
$\mathcal{M}$:	set of machines/blocks
i,j:	indices for jobs ($i = 1, \dots, I$ and $j = 1, \dots, J$)
r_i:	non-negative release time of job i/departure time of train i from its origin
d_i:	non-negative due date of job i/scheduled arrival time of train i at its destination
w_i:	non-negative importance weight of job i/train i
l_i:	number of machines to be visited by job i/number of blocks to be traversed by train i
$(m_{i1}, \dots, m_{i,l_i})$:	sequence of machines to be visited by job i/sequence of blocks to be traversed by train i
(i, m):	job, machine indices/train, block indices, for $i \in \mathcal{I}$ and $m \in \mathcal{M}$
$\mathcal{O}$:	set of operations defined by indices (i, m), for $i \in \mathcal{I}$ and $m \in \mathcal{M}$
p_{im}:	operation time for job i on machine m/running or dwell time for train i on block m
$s_i(m)$:	the immediate successor machine (the third successor machine) of (i, m) for two-aspect signaling (four-aspect signaling)
$S_i(m)$:	a set containing job, machine indices (i, m) for two-aspect signaling, and additionally containing the indices of its immediate and second successor operations for four-aspect signaling
h_{ijm}:	required time delay (headway) between the start of operations (i, m) and (j, m) when job i precedes job j on machine m
t_{im}:	starting time of job i on machine m
T_i:	tardiness of job i

It should be noted that there are three main differences in this model. Not all jobs can start at time zero, but they have a release time r_i which means trains can enter the network at a pre-defined time. There is a due date for each job which is used to calculate the tardiness of a job with $T_j = \max(L_j, 0)$, where the lateness of a job L_j is defined as $L_j = C_j - d_j$. This is similar to the pre-defined times that the train should reach its destination and it can be used to calculate how tardy is each train.

The objective function in train scheduling problem is minimizing total weighted tardiness $J|r_j|\sum w_j T_j$. It helps to consider the potential for delay propagation in a rail network, which results from the high interdependency of the trains. If some trains have priority over the others, it can be reflected in their weights; otherwise, weights can be defined as one. The generated schedule needs to be conflict-free and deadlock-free. Figure 5.1(a) presents a deadlock-free case and Figure 5.1(b) illustrates a deadlock situation.

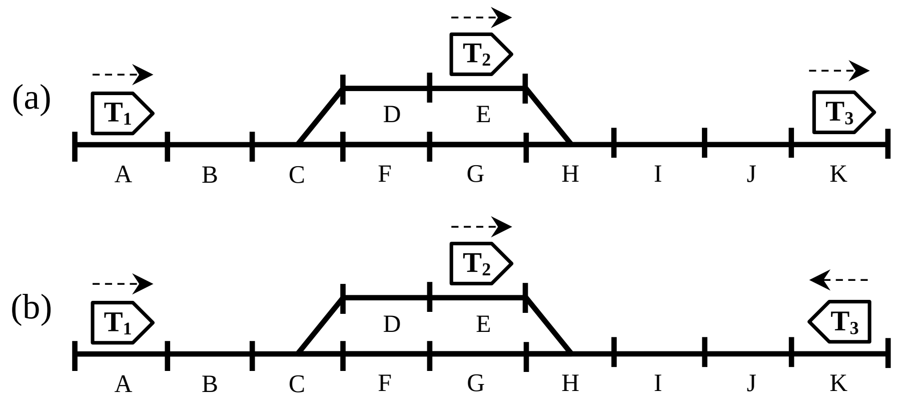

FIGURE 5.1: (a) Situation with no deadlock (b) Situation with deadlock

Thus, the train scheduling problem can be formulated as a job shop scheduling problem with additional constraints, and a corresponding Mixed Integer Linear Programming (MILP) model is specified in the following. As the model is similar to a job shop scheduling model with extra constraints to formulate train scheduling, we call this model *Modified Blocking Job Shop Scheduling (MB-JSS)* model.

$$\text{Minimize} \qquad z = \sum_{i \in \mathcal{I}} w_i T_i \tag{5.11}$$

subject to

$$T_i \geq t_{i,m_{i,l_i}} + p_{i,m_{i,l_i}} - d_i \qquad i \in \mathcal{I} \tag{5.12}$$

$$t_{i,m_{i,1}} \geq r_i \qquad i \in \mathcal{I} \tag{5.13}$$

$$t_{i,m_{i,k}} - t_{i,m_{i,k-1}} \geq p_{i,m_{i,k-1}} \qquad i \in \mathcal{I},\ k = 2, \ldots, l_i \tag{5.14}$$

$$t_{jm} - t_{im} + \bar{M}(1 - x_{ijm}) \geq \max\{p_{im}, h_{ijm}\} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{5.15}$$

$$t_{im} - t_{jm} + \bar{M}(1 - x_{jim}) \geq \max\{p_{jm}, h_{jim}\} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{5.16}$$

$$t_{jm} - t_{is_i(m)} + \bar{M}(1 - x_{ijm}) \geq \sum_{(i,k) \in S_i(m)} p_{ik} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{5.17}$$

$$t_{im} - t_{js_j(m)} + \bar{M}(1 - x_{jim}) \geq \sum_{(j,k) \in S_j(m)} p_{jk} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{5.18}$$

$$x_{ijm} + x_{jim} = 1 \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{5.19}$$

$$x_{ijm} \in \{0,1\} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{5.20}$$

$$T_i \geq 0 \qquad i \in \mathcal{I} \tag{5.21}$$

In this formulation, the total weighted tardiness objective function is defined in (5.11). The tardiness of a job is defined in constraint (5.12) by considering its starting time on the last machine of its sequence, its processing time on that machine and the due date of the job; this is equivalent to defining a train's delay. Ensuring that the starting time of a job on the first machine of its sequence is no earlier than its release time is achieved through constraint (5.13), which means a train can start only after it is ready on the first block.

Constraints (5.14) are the set of *conjunctive* constraints to ensure the processing order of a job on consecutive machines. It determines the running and dwell time constraints for trains. Modified *disjunctive* constraints (5.15) and (5.16) specify the ordering of different jobs on the same machine, and they are adapted to define the minimum headway between consecutive trains.

Alternative constraints (5.17) and (5.18) force a job to remain on a machine after completing its process until the next machine(s) becomes available. This pair of constraints can represent the signaling system of the network. The proposed MILP model is developed by Xpress-MP and the results are reported in Section 5.4. In the next Subsection 5.2.2, a graph representation is used to formulate the problem.

5.2.2 Modified disjunctive graph formulation

Modeling the conventional job shop scheduling problem for minimizing the makespan with a *disjunctive graph* is quite popular in the literature. This representation for $J||C_{\max}$ was firstly introduced by Roy and Sussman (1964). The conventional disjunctive graph is considered as a directed graph $G = (N, A, B)$ where N is the set of nodes and A and B are two sets of arcs to represent the conjunctions and disjunctions.

As it can be seen in Figure 5.2, an operation is shown by a node (i, m) in set N. There are also two dummy nodes in set N including a source U and a sink V. Solid arcs in set A refer to conjunctive arcs for jobs routes. If there is a conjunctive arc $(i, m) \rightarrow (i, n)$, it means that job i should be processed on m and then on n. The dotted pair of arcs in opposite direction are disjunctive arcs and belong to set B. They connect different jobs on a machine. For the sake of clarity, disjunctive arcs are only depicted for one machine in Figure 5.2. The length of the conjunctive and the disjunctive arcs are equal to the processing time of the operations from which they stem. There are arcs emanating from the source with the length zero and there are also arcs coming into the sink with the length of the last operation of each job.

Let $L(v, v')$ show the length of the critical (longest path from node v to v' in the graph. Using the following formulation one can calculate the *heads* and *tails* of the operations. The head of an operation r_{im} is the earliest time that an operation can start and it is defined by the length of the longest path from node U to the specific node (i, m) for job i and it is shown by

$$r_{im} = L(U, (i, m)), \tag{5.22}$$

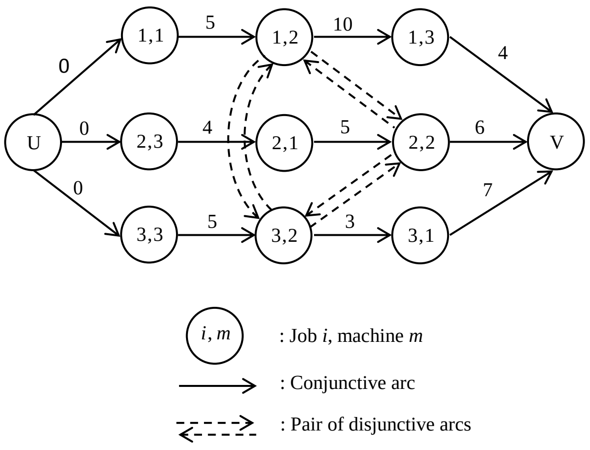

FIGURE 5.2: Disjunctive graph for $J||C_{\max}$

Similarly, length of the longest path from node (i, m) to the sink is called tail $q_{i,m}$ and shown by

$$q_{im} = L((i, m), V). \tag{5.23}$$

It should be noted that we use this convention that if there is no path from node (i, m) to another node (j, n), then $L((i, m), (j, n)) = -\infty$ which means that the length of the longest path between two nodes is either positive or $-\infty$.

Assume that the set of disjunctive arcs B is decomposed into cliques B_m for each machines such that

$$B = \bigcup(B_m : m \in \mathcal{M}). \tag{5.24}$$

A *selection* S_m in B_m contains only one member of each disjunctive arc pair of set B_m (Adams et al., 1988). A selection by choosing disjunctive arcs on a machine such that the graph is acyclic determines the order of the jobs to be processed on a machine. A selection is called *acyclic* if it contains no directed cycle and an acyclic selection S_m is associated with a unique sequence of the operations for machine m and vice versa. It should be noted that there is no negative cycle in the graph; Because, according to the definition of the longest path (See formula 6.24), there are either positive longest paths between nodes or there is no path between them.

A *complete selection* S includes the union of the selections S_m, one in each B_m, $m \in \mathcal{M}$). Thus, a complete selection of one disjunctive arc from each pair such that there is no cycle in the graph indicates a feasible schedule. Similarly, a *partial* *selection* is defined with the union taken over some subset $\mathcal{M}_0$ of $\mathcal{M}$. A partial selection corresponds to a partial schedule for the problem.

The longest path from source U to sink V defines the makespan. According to the definitions above, the problem is to find an acyclic complete selection $S \in B$ that minimizes the length of the longest path in directed graph G. So minimizing the makespan or finding the critical path is reduced to selecting the disjunctive arcs such that the length of the longest path is minimum.

Problem $J|r_j|\sum w_jT_j$ can also be represented by a disjunctive graph as shown by Pinedo and Singer (1999). The disjunctive graph can be slightly modified to be adapted to minimize total weighted tardiness in a job shop problem with jobs that have different release times. Similar to the conventional disjunctive graph, the graph $G = (N, A, B)$ can be defined with set N representing the nodes for each operation (i, m). The dummy source node U is the same, but there are I different sinks V_i for each job i. As before, the conjunctive arcs $(i, m) \to (i, n)$ in set A show pairs of consecutive operations (i, m) and (i, n) of job i. The length of the mentioned arc is p_{im} and the arc from (i, q) to V_i has the length p_{iq}. The arc from U to $(i, 1)$ has the length of r_i. Set B includes the pair of disjunctive arcs $(i, m) \to (j, m)$ and $(j, m) \to (i, m)$ with the lengths p_{im} and p_{jm} respectively. The disjunctive arcs come in pairs, and selecting one arc from a pair fixes the order of the corresponding two operations on a machine.

Figure 5.3 represents an example of a job shop with three jobs and three machines for $J|r_j|\sum w_jT_j$. To have a clearer picture, pairs of disjunctive arcs are only drawn for one machine. Associated with V_i nodes are the d_i values 10, 15 and 6 as the due dates of jobs 1, 2 and 3 respectively. d_i values are used to calculate the total weighted tardiness for the job shop problem. The dummy nodes V_i for each job i are introduced in the disjunctive graph formulation to define the lateness of the jobs. Consequently, the tardiness of each operation is determined with the help of the longest path calculations. More details of the computation is given in Subsection 5.3.5.2.

Using the analogy between the job shop scheduling and train scheduling, each operation is a train running on a block with a pre-determined time r_i to enter the network and a pre-determined time d_i to reach its destination. The conjunctive

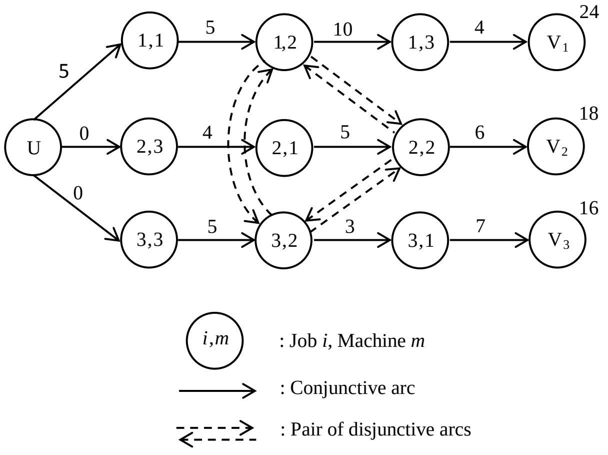

FIGURE 5.3: Modified disjunctive graph for $J|r_j|\sum w_j T_j$

arcs correspond to running and dwell time constraints. Likewise, the disjunctive arcs are associated with headway constraints and their weights need to be modified slightly to represent headway values as it is explained below.

However, modeling a job shop scheduling problem with a disjunctive graph does not take into account the buffer capacity between consecutive machines, which seems to be an important issue in many real-life scheduling problems. As a modification of the disjunctive graph, the alternative graph by Mascis and Pacciarelli (2002) addresses this restriction and it can be adapted to formulate different types of constraints. We adapt the modified disjunctive graph of Pinedo and Singer (1999) which minimizes total weighted tardiness, to the train scheduling problem with realistic operational and safety constraints. The alternative arcs are added to the disjunctive graph to define the signaling system.

Consequently, we introduce a modified disjunctive graph $G = (N, A, B, C)$ for train scheduling and rescheduling where set N contains a node for each operation (i, m), a dummy source U and I dummy sinks V_i for each job i. A is the set of conjunctive arcs that connects the pair of consecutive operations of the same job in order to take into account running and dwell time constraints. Set B is the set of *modified disjunctive arcs* that are represented by two arcs in opposite directions for every pair of operations (i, m) and (j, m). To represent headway for both following and opposite trains, the length of a disjunctive arc is simply modified as $\max\{p_{im}, h_{ijm}\}$ to consider the higher value between the running time and the headway.

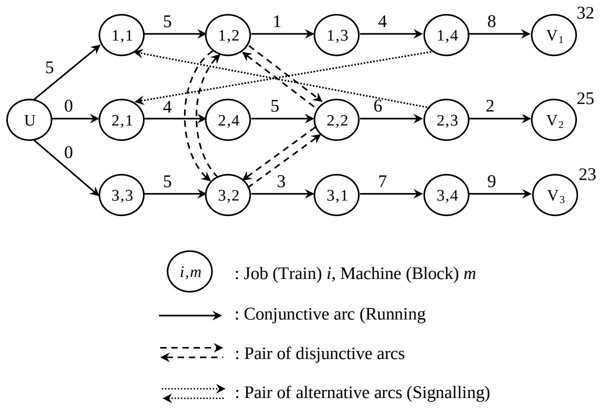
U
1,1
1,2
1,3
1,4
V1
32
2,1
2,4
2,2
2,3
V2
25
3,3
3,2
3,1
3,4
V3
23
5
0
0
5
1
4
8
4
5
6
2
5
3
7
9
i,m
: Job (Train) i, Machine (Block) m
: Conjunctive arc (Running
: Pair of disjunctive arcs
: Pair of alternative arcs (Signalling)

FIGURE 5.4: MB-JSS graph: Modified disjunctive graph with alternative arcs for $J|r_j|\sum w_jT_j$

The limitation of the disjunctive graph to be able to model the buffer capacity between consecutive machines properly is addressed in the new graph. In our problem, a job needs to stay on a machine after its processing time until the next machine becomes free. So set C includes the pairs of *alternative arcs* $(i, s_i(m))$ and (j, m) which are added according to the *alternative graph* of Mascis and Pacciarelli (2002). As shown in Figure 5.4, these arcs are adapted to have alternative arcs to keep following trains moving on green signals with a fixed speed under four-aspect signaling. For more clarity disjunctive arcs for only one machine and one pair of alternative arcs are drawn in Figure 5.4. More details about alternative arcs are given in the next Subsection 5.2.3.

5.2.3 Alternative arcs

The alternative arcs in this study are adapted from the study by Mascis and Pacciarelli (2002). D'Ariano et al. (2007) employ the alternative arcs for modeling the train scheduling in a two-aspect signaling system. Figure 5.5(a) shows alternative arcs for a two-aspect signaling system. There is potential for a conflict when a block is required by two trains at the same time. Thus, an order needs to be determined for the conflicting operations corresponding to conflicting trains which need to run on the same block. A pair of alternative arcs is introduced to model this situation. If (i, m) and (j, m) are two conflicting operations, we let $(i, s_i(m))$ and $(j, s_j(m))$ to be their immediate successor operations (their third successor operation) for two-aspect signaling (four-aspect signaling) of (i, m) and (j, m) respectively. $\bigcirc$ notation is to identify the pair of alternative arcs and a decision is made by choosing either the arc from node $(j, s_j(m))$ to (i, m) or the arc from node $(i, s_i(m))$ to (j, m).

In the model suggested by Mascis and Pacciarelli (2002), if (i, m) is scheduled before (j, m) on machine m and the length of the alternative arc from node $(i, s_i(m))$ to (j, m) is shown by $a_{s_i(m),j}$, (j, m) starts only $a_{s_i(m),j}$ time units after the start of operation (i, m). Similarly, if j is scheduled before i on machine m and the length of the alternative arc from node $(j, s_j(m))$ to (i, m) is denoted by $a_{s_j(m),i}$, (i, m) starts only $a_{s_j(m),i}$ time units after the start of operation (j, m). D'Ariano et al. (2007) define the length of the alternative arc as the setup time for the block section in the train scheduling problem with two-aspect signaling system.

Figure 5.5(b) represents the adapted alternative arcs in the current research for a four-aspect signaling railway network. Similar to two-aspect signaling system, there is a potential conflict for two trains which need to run on the same block. In addition, in a four-aspect signaling network, trains need to be kept three blocks away from each other. To resolve the conflict and keep the trains three blocks away, we introduce a pair of alternative arcs to define the order of the conflicting trains and keep them far enough. We let (i, m) and (j, m) to be two conflicting operations with $(i, s_i(m))$ and $(j, s_j(m))$ to be their third successor operations respectively.

If operation (i, m) is scheduled before operation (j, m) and the sum of operation time of operation (i, m) and its immediate and second successor operations are defined as $\sum_{(i,k)\in S_i(m)} p_{ik}$, (j, m) starts only $\sum_{(i,k)\in S_i(m)} p_{ik}$ time units after the start of operation (i, m). It means that (j, m) can only start after (i, m) and its immediate and second successor operations finished their operations. In other words, the modified alternative arc from node $(i, s_i(m))$ to (j, m) is selected and its length is assumed to be equal to zero. Similarly, if (j, m) is scheduled before (i, m) on machine m and the sum of operation time of operation (j, m) and its immediate and second successor operations are defined as $\sum_{(j,k)\in S_j(m)} p_{jk}$, (i, m) starts only $\sum_{(j,k)\in S_j(m)} p_{jk}$ time units after the start of operation (j, m). In this case, the alternative arc from node $(j, s_j(m))$ to (i, m) is selected and its length is equal to zero.

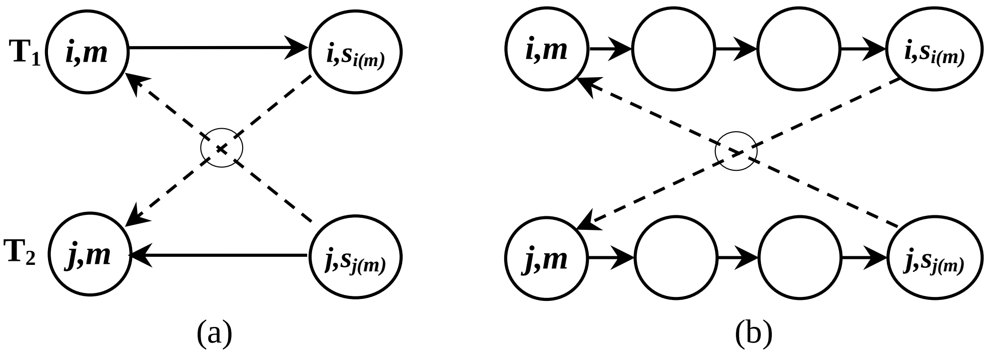

FIGURE 5.5: (a) alternative arcs for two-aspect signaling (b) Modified alternative arcs for four-aspect signaling

The role of the alternative constraints in modeling train separation and signaling should be emphasized here. The alternative arcs help to maintain enough separation between two following trains with regard to the network signaling system. For instance, in our case study with the four-aspect signaling, following trains are required to be three blocks away all the time. It can be clearly seen that alternative arcs do not allow following trains to get closer than three blocks even during disruptions. It is also obvious that headway constraints cannot singly take into account the signaling characteristics.

Any feasible schedule corresponds to an acyclic complete selection of disjunctive and alternative arcs such that one arc is chosen from each pair of disjunctive and alternative arcs. Conversely, a complete selection of the disjunctive and alternative arcs by choosing exactly one arc from each pair which result in a graph with no cycle, leads to a feasible schedule. In the following Section 5.3, some algorithms are suggested to solve this problem based on a special heuristic for job shop scheduling problem.

It should be noted that both disjunctive and alternative arcs are important in this study. As it is mentioned before, modified disjunctive arcs are added for both following and opposite trains whereas alternative arcs are added only for following trains. In the complete set of modified disjunctive graph formulation, the disjunctive arcs may be redundant for following trains when a disruption occurs. It is due to the longer running and dwell times compared to the headway value during disturbances. However, the disjunctive arcs are needed when a single machine is formulated for the solution method as it is discussed in Subsection 5.3.5.2. Alternative arcs are needed so that the following trains are kept three blocks far from each other even during a disruption. On the other hand, alternative arcs can be redundant for following trains when there is no disruption as the headway value in our case study is bigger than running and dwell time values.

5.3 Solution methods

This section deals with solution methods to solve the scheduling problem that we formulated in the previous section. We develop three types of algorithms: A FCFS simple dispatching heuristic, a greedy heuristic and a specific job shop scheduling heuristic called *Shifting Bottleneck procedure (SB)*. The aim is to suggest an algorithm that finds the order and timing of the trains on the network blocks so that the delay is minimized. This approach can be used in an offline decision making process and online planning if the algorithm is fast enough. Thus, the development of several algorithms which are based on SB heuristic is the main focus in this section.

In Subsection 5.3.1, the First Come First Served (FCFS) algorithm for this problem is represented. The FCFS algorithm is a heuristic based on a FCFS dispatching rule which simulates the real-world application. Since the problem can easily lead to infeasibility when FCFS is employed, we suggest an alternative heuristic in Subsection 5.3.2 in order to produce a baseline for comparisons. Finally, the novel SB algorithm and its variations are discussed in Subsection 5.3.5.

5.3.1 FCFS dispatching rule

First Come First Served (FCFS) also called First In First Out (FIFO) algorithm is the one of the simplest scheduling algorithm where operations are dispatched based on their arrival time on the ready machine. It seems to be fair according to common sense but applying it may make urgent jobs wait for non-urgent jobs and maybe important jobs wait for unimportant jobs. FCFS algorithm is easy to implement and understand although the major drawback is that it may increase the total weighted tardiness which is the objective function in the our problem. Also the average waiting time is long when FCFS is applied. In general, when FCFS is implemented as a solution method, no attention is made to the objective function of the problem under study whether it is total weighted tardiness or other criteria for optimization.

According to (D'Ariano, 2008), the practice is common among train dispatchers who permit the train which arrives in advance to enter a block. Furthermore, most of the rescheduling decisions taken by dispatching systems operate on the spot and may implement simple dispatching rules like FCFS. This means when a conflict arises the dispatcher resolves it by assigning the block to the first train that requires it. This rule does not have any special order and the dispatching decision lets the traffic flow with the actual order.

When developing the FCFS algorithm, one should make sure to comply with the train operational rule in this chapter. First rule is to allow a train to enter a block if it is clear. A train traversal time on a block is equal to its running time on that block. The time difference between two consecutive train is at least equal to maximum value between the headway and the running time of the preceding train which makes the second rule. Last rule is about allowing a following train to enter a block if the preceding train is far away enough according to the signaling system. For example in our case study, we only let a following train start its movement on a block when the preceding train is three blocks away.

5.3.2 Earliest Arc heuristic

A heuristic algorithm is provided in order to find feasible solutions for the train scheduling and rescheduling problem. As the problem is formulated as a modified blocking job shop scheduling problem, we need to find an acyclic selection of disjunctive and alternative arcs in the MB-JSS graph in order to have a feasible schedule.

It is reported in the literature that solving a blocking job shop scheduling problem seems to be much harder than the classical job shop scheduling and there is a concern for running into infeasibility when developing algorithms for this type of problems (Groeflin and Klinkert, 2009). A structural property of the blocking job shop scheduling problem is that in contrast to the job shop scheduling problem, a feasible partial schedule cannot be always extended to a complete schedule. Therefore, a heuristic which builds a solution incrementally has a high risk of resulting in infeasibility (Groeflin and Klinkert, 2009; Mascis and Pacciarelli, 2002). Algorithms based on priority rules are an example of repeatedly enlarging a selection.

As we have observed in computational tests, the FCFS dispatching algorithm fails to find a feasible solution in many cases. Therefore, we need to develop a solution method which can serve as a baseline for our comparisons and does not easily result in infeasible solutions. Mascis and Pacciarelli (2002) offer four variations of a greedy heuristic to find an acyclic selection in the alternative graph which has been later employed for train scheduling in a study by D'Ariano et al. (2007). Generally, it is an iterative algorithm which chooses a pair of alternative arcs based on a criterion which is different for each variation of the heuristic. Then, one of the arcs which is compatible with the current solution is selected.

We employ the general framework of the mentioned algorithms and modify it in terms of the selection criterion for the pair of arcs. In addition, we select one arc at a time in a pair of disjunctive arcs in the MB-JSS graph instead of alternative arcs in the greedy heuristic suggested by Mascis and Pacciarelli (2002). After adding the disjunctive arc, corresponding alternative arc is added. Then, alternative arcs are used to add implied alternative arcs for following trains which share common blocks. The implied alternative arcs are derived by employing some rules which are defined in the following. Further, we fix the implied disjunctive arcs among the jobs on a machine. This should be done for all the nodes which correspond to both following and opposite trains running on common blocks.

The basic idea of the algorithm is to incrementally extend a partial selection S which is initially equal to $\emptyset$. In order to illustrate some concepts with more clarity in this section, an operation (a, m) is denoted by a and subsequently an arc from node a to node b is defined as (a, b). The algorithm selects a pair of unselected disjunctive arcs (i, j) and (j, i) between operations i and j processed on machine m such that both operations have the smallest release date among all the other nodes. Only in this section, the release date for operation i is shown by r_i which is calculated similar to r_{im} by formula (5.22).

As the algorithm chooses the arcs with the earliest release dates, it is called an *Earliest Arc (EA)* heuristic. The EA heuristic enlarges a partial schedule repeatedly. So it risks the chance of ending up with an infeasible solution. We have incorporated a mechanism of fixing implied arcs in the MB-JSS graph which can be helpful to avoid infeasibility in many cases, but it cannot guarantee a feasible solution. The idea is inspired by the concept of immediate selection in a disjunctive graph by Brucker et al. (1994a) and static implication rules in an alternative graph by D'Ariano et al. (2007). The main idea in both studies is to fix some implied arcs in the corresponding graphs which reduces the amount of computational effort for the introduced branch-and-bound methods for the corresponding job shop problem.

5.3.3 Static implication rules and static implications

D'Ariano et al. (2007) offer two propositions quoted as below, where f_{ab} denotes the length of the alternative arc (a, b) and $L_S(b, i)$ shows the longest path from node b to i according to the current selection of arcs denoted by S. Assume that $L_\emptyset(b, i)$ is the longest path from node b to i according to an empty selection $S = \emptyset$. $S = \emptyset$ holds when MB-JSS graph G is decomposed of only conjunctive arcs which are associated with the train routes.

- "Consider a selection S and two unselected alternative pairs $((a, b), (c, d))$ and $((i, j), (h, k))$. If $f_{ab} + L_S(b, i) + f_{ij} + L_S(j, a) \geq 0$, arc (h, k) is implied by selection $S \cup (a, b)$ and arc (c, d) is implied by selection $S \cup (i, j)$."
- "Consider two alternative pairs $((a, b), (c, d))$ and $((i, j), (h, k))$. Then $f_{ab} + L_\emptyset(b, i) + f_{ij} + L_\emptyset(j, a) \geq 0$ if the following conditions hold.
  1. Nodes b and i are associated with train T_1 and are connected by a directed path of fixed arcs, i.e., T_1 executes b before i.
  2. Nodes j and a are associated with train T_2 and are connected by a directed path of fixed arcs, i.e., T_2 executes j before a.

3. T_1 and T_2 pass through two block sections $((a,b),(c,d))$ and $((i,j),(h,k))$."

The above mentioned propositions lead to the *static implication rules* by D'Ariano et al. (2007) which are effective for the train scheduling problem. D'Ariano et al. (2007) define *static implications* as implied alternative arcs that are selected according to the static implication rules among different alternative pairs as below.

- T_1 and T_2 are following trains running on adjacent blocks m_1 and m_2. According to Figure 5.6, nodes a and j as well as nodes c and k coincide. Thus, arcs (a,b) and (h,k) imply each other as well as (c,d) and (i,j).
- T_1 and T_2 are opposite trains running on blocks m_1 and m_2. According to Figure 5.7, arcs (a,b) implies arc (h,k) imply as well as arc (i,j) implies (c,d). It should be noted that arc (h,k) does not imply arc (a,b) and arc (c,d) does not imply arc (i,j).

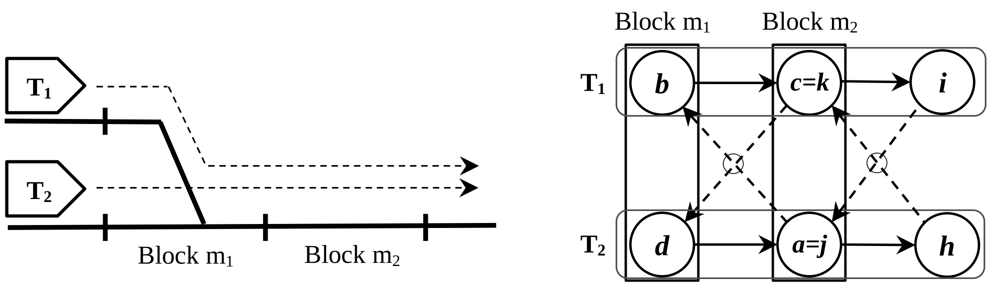

FIGURE 5.6: Illustration of implication rules for following trains on a rail network with two-aspect signaling

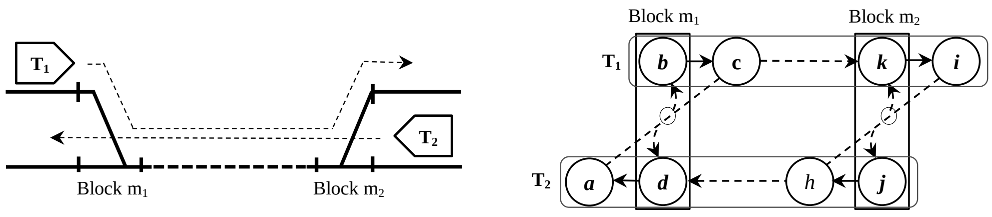

FIGURE 5.7: Illustration of implication rules for opposite trains on a rail network with two-aspect signaling

We need to modify the implication rules slightly to be compatible with four-aspect signaling system in our case study. So *modified static implications* are defined as implied alternative arcs which are added according to the *modified static implication rules* given in the following.

- T_1 and T_2 are following trains running on adjacent blocks m_1, m_2, m_3 and m_4. According to Figure 5.8, nodes a and j as well as nodes c and k are connected through fixed arcs. Thus, arcs (a, b) and (h, k) imply each other as well as (c, d) and (i, j).
- T_1 and T_2 are opposite trains running on blocks m_1, m_2, m_3, m_4 and m_5. According to Figure 5.9, arc (a, b) implies arc (h, k) as well as arc (i, j) implies (c, d). It should be noted that arc (h, k) does not imply arc (a, b) and arc (c, d) does not imply arc (i, j).

It should be noted that modified static implications are used in our study only for the following trains running on common blocks. The main difference to static implications is that these arcs aim to keep the following trains three blocks away. We have observed that the modified static implications help us to avoid an acyclic selection of arcs in many cases; otherwise, we may easily end up in having a cycle in the MB-JSS graph which is equivalent to a deadlock situation in train scheduling or rescheduling problem. From now on, we will simply refer to modified static implications as *static implications*.

However, it should be noted that in this study we only make use of implication rules for following trains as disjunctive arcs can ensure enough separation between opposite trains (see Subsection 5.2.2 for more details about the role of disjunctive and alternative arcs). Figure 5.8 presents the implications modified rules for two following trains on a network with four-aspect signaling.

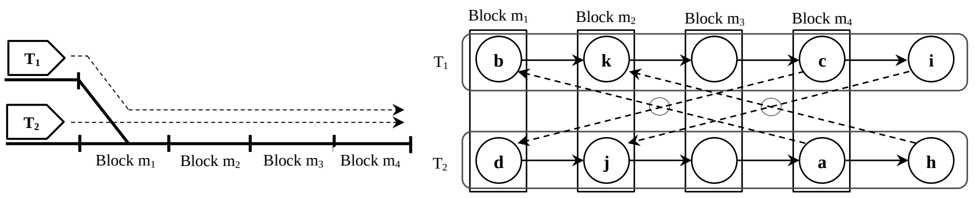

FIGURE 5.8: Illustration of modified implication rules for following trains on a rail network with four-aspect signaling

Figure 5.9 shows the modified implications rules for two opposite trains on a network with four-aspect signaling.

After fixing static implications, the precedence constraints are imposed among the nodes corresponding to the jobs operated on the same machine. We fix the

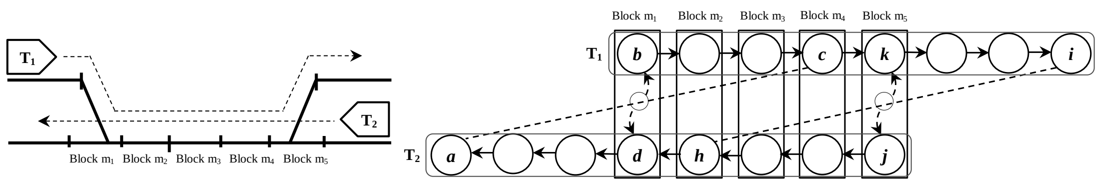
T1
T2
Block m1 Block m2 Block m3 Block m4 Block m5
Block m1 Block m2 Block m3 Block m4 Block m5
T1
b
c
k
i
T2
a
d
h
j

FIGURE 5.9: Illustration of modified implication rules for opposite trains on a rail network with four-aspect signaling

implied arcs among operations which are performed on the same machine and call them *simple implied arcs*. We fix these simple implied arcs which correspond to both following and opposite trains running on common blocks, after the static implications are fixed.

5.3.4 EA heuristic algorithm

Now, the proposed EA heuristic for the MB-JSS graph $G = (N, A, B)$ can be described formally in Algorithm 5.

Let i and j be two operations which are processed on machine m. After selection of a disjunctive arc (i, j), the corresponding alternative arc from operation i on machine $s_i(m)$ to operation j on machine m is added. Subsequently, static implications are added according to static implication rules by D'Ariano et al. (2007). Then, we add simple implied arcs for both following and opposite trains which share common blocks.

The EA heuristic is helpful in the comparison of the scheduling solution methods. The SB algorithms proposed in Subsection 5.3.5 are examined against this algorithm.

5.3.5 Modified Shifting Bottleneck procedure

The train scheduling problem is formulated as a job shop problem and the job shop problem is known to be NP-hard (see Garey and Johnson (1979) and Ullman (1975)). A train scheduling problem of a practical size can easily result in a huge job shop problem with numerous nodes and arcs. As we cannot solve the proposed

Algorithm 5 EA heuristic

```
$S \leftarrow \emptyset$
while $B \neq \emptyset$ do
  select arc $(i, j)$ out of a pair of modified disjunctive arcs such that $r_i$ is the
  first smallest value among the other operations and $r_j$ is the second smallest
  value among the other operations on machine $m$.
  select corresponding alternative arcs, static implications and the simple im-
  plied arcs among $i$ and $j$ and all the previously scheduled jobs on machine
  $m$.
  update $G = (N, A, B)$ with the newly selected modified disjunctive arc,
  alternative arcs, static implications and simple implied arcs.
  if there is a cycle in the graph or any of the alternative arcs or static impli-
  cations or simple implied arcs is forbidden then
    select arc $(j, i)$ and unselect $(i, j)$ and all the corresponding alternative
    arcs, static implications and simple implied arcs.
    select corresponding alternative arcs, static implications and the simple
    implied arcs among $j$ and $i$ and all the previously scheduled jobs on machine
    $m$.
    update $G = (N, A, B)$ with the newly selected modified disjunctive arc,
    alternative arcs, static implications and the simple implied arcs.
    if there is a cycle or any of the alternative arcs or static implications or
    simple implied arcs is forbidden then
      The procedure failed to find a feasible solution, exit.
    end if
  end if
end while
```

MILP model optimally in a reasonable amount of time, it is preferable to employ local search methods for which computational time is more predictable.

The *Shifting Bottleneck* (SB) procedure of Adams et al. (1988) is a well-known heuristic for solving a classical job shop scheduling problem $J||C_{\max}$ that is formulated as a disjunctive graph. The success in applying the SB procedure on benchmark instances in the job shop scheduling literature has led to a number of studies that employ the SB approach. It can be also used as a framework for other heuristics such as tabu search, simulated annealing and genetic algorithms. Although there is no theoretical performance guarantee for SB, its empirical performance has a good track record.

Shifting bottleneck procedure is selected as the most appropriate method to solve the scheduling problem as it can offer a good balance between computational complexity and the quality of generated schedules which is one of the reasons that shifting bottleneck procedure has attracted attention of researchers after its introduction (Liu and Kozan, 2012).

Another advantage of the SB algorithm is that it can be modified to solve various types of scheduling problems including real-life job shop scheduling applications. Thus, SB heuristic is chosen as it can consider special characteristics of the Train scheduling problem formulated as a modified job shop scheduling problem. In addition, the Shifting bottleneck heuristic can be improved to find feasible solutions as it is not trivial to find a feasible complete schedule for Modified Blocking Job Shop Scheduling (MB-JSS) model.

Moreover, we realize that while approaches other than shifting bottleneck such as branch and bound algorithm and metaheuristics have been applied on classical job shop shop problems, they can be difficult to implement on the additional complicating features of the modified job shop scheduling problem.

Furthermore, the proposed method is very promising because it can be implemented as a fundamental tool to model and solve many real-world scheduling problems that should consider the capacity of resources (machines) and different inter-resource buffer conditions. To the best of our knowledge, very few researchers have addressed the application of the SB heuristic to the train scheduling problem.

It is worthwhile to note that Groeflin and Klinkert (2009) is among the best algorithms for the blocking job shop problems compared to Mascis and Pacciarelli (2002), Mascis and Pacciarelli (2000) and Brizuela et al. (2001). The problem formulation with the Generalized Blocking Job Shop (GBJS) model by Groeflin and Klinkert (2009) includes additional features such as transfer times for moving a job from one machine to the next machine and sequence-dependent setup times between consecutive operations on a machine. In GBJS, four consecutive steps are associated with each operation of a job (i) a take-over step where the job is taken over from the machine that carried out the job's previous operation; (ii) a processing step where some work is done on the job; (iii) a waiting period during which the job waits on the machine and blocks it till it is transferred; (iv) a hand-over step where the job is handed over to the machine which does the job's next operation.

In GBJS graph, the node set includes two nodes for each operation, a take-over node and a hand-over node. It should be also noted that the generalized disjunctive graph differs from classical disjunctive graph in that a disjunctive set may have two arbitrary arcs, whereas in the classical disjunctive graph, there are two reverse arcs. For any two operations on a machine, the set of disjunctive arcs includes two arcs joining the hand-over node to the take-over node and vice versa. There is also the set of processing arcs which join the take-over node to the hand-over node of each operation (See Groeflin and Klinkert (2009) for more details about the generalized disjunctive graph).

As it is mentioned in Subsection 4.2.2.2, Groeflin and Klinkert (2009) construct feasible neighbors by exchanging critical arcs together with some other arcs based on two structural properties of the underlying disjunctive graph which are called the concept of closures and a key result on short cycles. Their tabu search algorithm is developed based on this neighborhood definition.

It can be clearly seen that the number of nodes and arcs in GBJS graph are not only more than the classical job shop, but also much more than our proposed MB-JSS problem. It should be noted that our modified disjunctive graph of a practical size instance may include hundreds of machines (blocks) and tens of jobs (trains), resulting in a complex job shop problem with blocking and other additional constraints to be solved within the strict time limits which is imposed by the real-time nature of the problem.

Formulating the train scheduling problem according to generalized disjunctive graph of Groeflin and Klinkert (2009) can result in much bigger number of nodes and arcs which can affect computational time. In addition, associating four consecutive steps with each operation of a job is not helpful with modeling the characteristics of the train scheduling problem and it can make the graph more complicated. Sequence-dependent setup times and transfer times, considered in the GBJS graph, are also not relevant in the train scheduling problem. Consequently, the neighbors structure suggested for the tabu search is not viable for solving the train scheduling problem.

Thus, we choose to formulate the problem according to modified disjunctive graph in Subsection 5.2.2 which is inspired by alternative graph of Mascis and Pacciarelli

(2002). We also select SB algorithm as the most appropriate method to solve the formulated train scheduling problem.

In the following, firstly we talk about the conventional SB algorithm in Subsection 5.3.5.1. Then, we propose a modified SB algorithm for the train scheduling problem in Subsection 5.3.5.2. In particular, a general framework for the modified SB algorithm with several variants is introduced for MB-JSS problem.

5.3.5.1 SB Algorithm

In this subsection, the conventional SB algorithm is presented and the main steps of the algorithm are discussed. Adams et al. (1988) introduce the SB procedure which sequences the jobs on machines consecutively. According to the study by Adams et al. (1988), an acyclic selection S_m is associated with a unique sequence of jobs on machine m and vice-versa. So *sequencing machine m* means choosing an acyclic selection in B_m, where B_m is a subset of cliques in set of disjunctive arcs as mentioned in equation (5.24). At each step, the machines which are not sequenced yet are sequenced by solving a one-machine scheduling problem. The machine with the maximum makespan is selected as the *bottleneck*. The same process is repeated until all machines are sequenced.

To be more specific, an iteration of the algorithm is described in the following. Let $\mathcal{M}$ denote a set of M machines and $\mathcal{M}_0$ show a subset of all machines $\mathcal{M}$ which have already been sequenced. Thus, $\mathcal{M}_0$ includes a selection of disjunctive arcs which has been fixed in the corresponding disjunctive graph G in the previous iterations. It should be noted that $\mathcal{M}_0 = \emptyset$ at the start. The aim in each iteration is to determine which machine in set $\mathcal{M} \setminus \mathcal{M}_0$ should be included in set $\mathcal{M}_0$. This is made possible by sequencing the operations on the $\mathcal{M} \setminus \mathcal{M}_0$ and evaluating which machine can be selected as the bottleneck.

Assuming that machines in set $\mathcal{M}_0$ are already sequenced, we keep the disjunctive arcs of the $\mathcal{M}_0$ and delete all disjunctive arcs of the machines which are still remained to be sequenced $\mathcal{M} \setminus \mathcal{M}_0$. This graph is called G' and it has one or more critical paths that determine the makespan called $C_{\max}(\mathcal{M}_0)$. When the disjunctive arcs of the machines are deleted, it implies that the operations in this graph are now carried out in parallel instead of one after another as if the machines have infinite capacity or each operation has the machine for itself.

There are different ways to decide that a machine is a bottleneck. We can express the bottleneck quality in terms of solving a one-machine scheduling problem to minimize the maximum lateness in a single machine problem with due dates. Thus, we consider each machine in $\mathcal{M} \setminus \mathcal{M}_0$ as a $1|r_j|L_{\max}$. This problem is proven to be strongly NP-hard Lenstra (1977). However, it has received a lot of attention in the literature which has resulted in approaches with reasonable performances Pinedo (2008).

The operation (i, m), $m \in \mathcal{M} \setminus \mathcal{M}_0$ has to be processed in a time window defined by head and tail of the node (i, m). The release date or head of the node (i, m) is equal to $L(U, (i, m))$ in G'. The due date is calculated as $C_{\max}(\mathcal{M}_0)$ minus $L((i, m), V)$ (the tail of the node) plus p_{im}.

Solving single machine problems, the minimum $L_{\max}$ denoted by $L_{\max}(m)$ is found for each machine m. The machine with the largest maximum lateness is the most critical machine among the others and it is the bottleneck. This machine is labeled as k and the maximum lateness is shown by $L_{\max}(k)$. Then, we need to fix all the corresponding disjunctive arcs for machine k in graph G'. When the disjunctive arcs are fixed, a sequence of the operations is defined for machine k.

Another step before starting the next iteration should be carried out which is re-sequencing all machines sequenced earlier. In this step which completes all steps in the current iteration, we re-sequence all the machines in set $\mathcal{M}_0$. It has to be done to see if the makespan can be reduced by an alternative sequence of jobs on machine m'. Machine m' is taken out from set $\mathcal{M}_0$ by deleting its disjunctive arcs. A single machine problem $1|r_j|L_{\max}$ is formulated for machine l by defining the release dates and due dates of the operation as mentioned before. Then, the sequence that minimizes $L_{\max}(l)$ is found. This should be repeated for all machine in set $\mathcal{M}_0$ and this completes the iteration. This step is also known as *re-optimization* (Adams et al., 1988).

An outline of the SB heuristic can be defined as below.

Classical SB algorithm

Step 1: Set the initial condition $\mathcal{M}_0 = \emptyset$. Graph G should include all conjunctive arc and no disjunctive arcs with $C_{\max}(\mathcal{M}_0)$ equal to longest path in graph G.

Step 2: Identify and solve a subproblem $1|r_j|L_{\max}$ for each un-sequenced machine $m \in \mathcal{M} \setminus \mathcal{M}_0$.

Step 3: Determine the bottleneck machine $m', m' \in \mathcal{M} \setminus \mathcal{M}_0$ with the largest maximum lateness $L_{\max}(m')$ and sequence the jobs on it according to the sequence obtained in step 2. Set $\mathcal{M}_0 = \mathcal{M}_0 \cup \{m'\}$.

Step 4: Stop if $\mathcal{M}_0 = \mathcal{M}$, otherwise go to Step 2.

There is a re-optimization step that is often used in the classical version of the SB algorithm. It is implemented after Step 3 and before starting a new iteration and it can be summarized as below

Re-optimize each machine $l \in \mathcal{M}_0 \setminus \{m'\}$ by solving its subproblem taking into account the sequences on machines $\mathcal{M}_0 \setminus \{l\}$.

In Subsection 5.3.5.2 we propose a modified SB algorithm and explain the modifications of the algorithm with regard to each step.

5.3.5.2 Modified SB algorithm

In this subsection, we develop a novel modified SB algorithm for the train scheduling problem which is formulated as a MB-JSS problem. This algorithm is inspired by a SB algorithm reported by Pinedo and Singer (1999). The novelty of our SB algorithm compared to Pinedo and Singer (1999) study is in its ability to address the train scheduling problem which require special characteristics. Most of these special characteristics arise from the formulation of the train scheduling problem which has additional operational and safety constraints. Whereas Pinedo and Singer (1999) solve the problem for a job shop scheduling problem to minimize total weighted tardiness, suggested SB algorithm is employed to handle two special constraints with regard to headway and signaling of the network.

In order to solve the problem, Pinedo and Singer (1999) use a partial enumeration algorithm which creates partial schedules in each step by using two elimination methods. However, the use of a partial enumeration algorithm for solving the single machine subproblems for the MB-JSS problem can result in cycles in the graph. In our novel approach, we handle this problem by employing alternative arcs and static implications which are described in Subsection 5.3.2 to create a deadlock-free schedule.

The proposed SB algorithm is also unique in that it can be adapted to consider different heuristics to solve the single machine subproblems. Therefore, we propose different variants of the SB algorithm which may or may not include the re-optimization step. However, Pinedo and Singer (1999) do not look at different variations of their SB algorithm.

D'Ariano et al. (2007) implement a branch-and-bound approach for an alternative graph formulation to optimize the railway traffic flow during disruptions. We offer a novel SB heuristic for an extended formulation of the train scheduling problem by employing alternative arcs in our MB-JSS problem. While branch-and-bound approach in D'Ariano et al. (2007) study fixes a precedence constraint between two trains to define their order at a conflict point such as a crossing or merging section, our SB algorithm solves the model for four-aspect signaling system in the UK network to keep following trains three blocks away.

Moreover, we extend and use the static implication rules suggested by D'Ariano et al. (2007) in our SB algorithm. D'Ariano et al. (2007) employ them in a pre-processing step mainly for speeding up their branch-and-bound algorithm; whereas, we use them dynamically in each iteration of our SB algorithm as a mechanism to avoid cycles in the graph and therefore to obtain infeasible solutions. It should be noted that D'Ariano et al. (2007) add alternative and static implications between all trains, but we make use of them only for following trains to conform to signaling system in the UK network.

Another special feature in our SB algorithm is to minimize total weighted tardiness of the jobs in the MB-JSS problem whereas branch-and-bound approach in D'Ariano et al. (2007) study addresses the makespan. More details of our algorithm novelties and modifications are indicated as we explain particular components and steps of the algorithm in the following.

In Section 5.2.2, we have discussed how the conventional disjunctive graph for $J||C_{\max}$ can be adapted to $J|r_j|\sum w_jT_j$. Using the modified disjunctive graph, Pinedo and Singer (1999) suggest an SB heuristic to solve $J|r_j|\sum w_jT_j$. An outline of the SB algorithm is shown in Figure 5.10.

In the subproblem formulation step, a subproblem is created for each machine that has not been sequenced. Solving the subproblem results in a sequence that estimates the increase in total weighted tardiness. The optimization subproblem heuristically solves the single machine problem to minimize the estimated increase. The machine with the maximum among these increases is selected in bottleneck selection step and its corresponding disjunctive arcs are added to the graph G. The problem iterates until all machines are sequenced.

The local release date for operation (j,m) is shown by r_{jm} which is similar to the release time that we have defined in equation (5.22). However, the due date is slightly different from the SB for classical job shop. The local due date for operation (j,m) with respect to each job k is defined as

$$d_{jm}^k = \begin{cases} \max\{C_k, d_k\} - L((j,m),V_k) + p_{jm} & \text{if } L((j,m),V_k) \text{ exists,} \\ \infty & \text{otherwise.} \end{cases} \tag{5.25}$$

where C_k is the completion time of job k and $L((j,m),V_k)$ is the longest path from operation (j,m) to V_k, where V_k is the sink corresponding to job k. Basically, d_{jm}^k is defined in terms of the longest path calculations for operation (j,m) and it translates the due date of operation (j,m) for each node V_k.

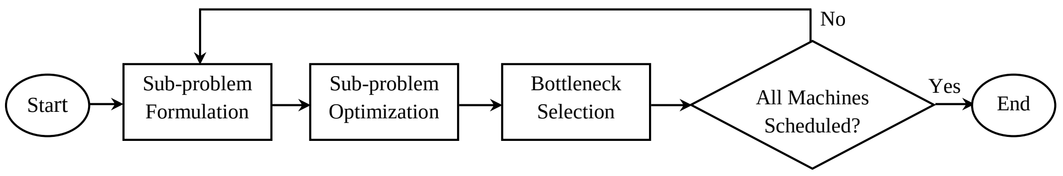

Figure 5.10: Shifiting Bottleneck (SB) flowchart. Adapted from Pinedo and Singer (1999)

In order to calculate the mentioned values for the single machine subproblem, we firstly apply the well-known *topological sort algorithm* or *topological ordering algorithm* in computer science (Cormen et al., 2001). This algorithm finds a linear order of the vertices of a directed graph so that for an arc (a, b) from node a to b, a comes before b in the presented order. In our study, the selected arc can be a conjunctive arc, a selected arc from the set of disjunctive arcs or a selected arc from the set of alternative arcs. Then, longest path calculations are carried out based on the modified disjunctive graph.

A partial enumeration heuristic is developed to solve the $1|prec; p_j = p; r_j|\sum T_j$ approximately based on a well-known priority rule developed by Vepsalainen and Morton (1987) for minimizing total weighted tardiness on a single machine. The so-called the *Apparent Tardiness Cost* (ATC) rule has shown to produce good quality solutions and it is superior to the other dispatching rules for minimizing total weighted tardiness in single machine problems. The ATC index can be defined as

$$I_j(t) = \frac{w_j}{p_j} \exp\left(- \frac{\max(d_j - p_j - t, 0)}{K\bar{p}} \right) \tag{5.26}$$

where t is the earliest time that the machine becomes available, $\bar{p}$ is the average processing time of jobs assigned to the machine, and K is a scaling parameter whose value can be determined through computational tests.

Later, Pinedo and Singer (1999) adapt the ATC index to take into account jobs release time and local due dates for $1|r_j|\sum w_j T_j$. We use the precise ATC index formula from Pinedo and Singer (1999) and just slightly modify the indices for more clarity as below

$$I_{jm}(t_m) = \sum_{k=1}^{n} \frac{w_k}{p_{jm}} \exp\left(- \frac{\max(d_{jm}^k - p_{jm} + (r_{jm} - t_m), 0)}{K\bar{p}_m} \right) \tag{5.27}$$

where t_m denotes the earliest time that the machine can start processing and $\bar{p}_m$ averages processing time of jobs assigned to machine m.

It should be noted that if there is no path from operation (j, m) to the sink V_k, local due date is $d_{jm}^k = \infty$. So it is obvious that the corresponding term in the summation is equal to zero. Using the modified ATC index, operations are sequenced on the un-sequenced machines. The operation with the highest priority is sequenced in the first position. Then, the remaining indices are recalculated to find the operation in second position. This should be repeated till all operations are sequenced.

Let C_{jm} be the completion time of operation (j, m). The tardiness of job k is increased by at least $C_{jm} - d_{jm}^k$ if the operation finishes after its local due date d_{jm}^k. Thus, we have the tardiness of operation (j, m) with respect to the due date of job k defined as

$$T_{jm}^k = \max\{C_{jm} - d_{jm}^k, 0\}. \tag{5.28}$$

Scheduling all operations on machine m increases the tardiness of job k by at least

$$\max_{(j,m)\in N_m} T_{jm}^k \tag{5.29}$$

where N_m is the set of nodes corresponding to the operations processed on machine m. Therefore, we can expect an overall increase of

$$\sum_{k=1}^{n} w_k \Big(\max_{(j,m)\in N_m} T_{jm}^k \Big), \tag{5.30}$$

in the objective function if operations on machine m are sequenced. The aim is to find the bottleneck machine with a sequence of operations which corresponds to the subproblem with the highest value.

In general, the SB procedure is a decomposition approach to solve multiple machine problems by selecting a machine and defining its corresponding arcs in turn. The proposed SB algorithms differ from the conventional SB in solving the single machine problem and finding the bottleneck. While the original SB considers an exact method to solve the single machine problem of minimizing the maximum lateness of jobs with release dates on a single machine (problem $1|r_j|L_{\max}$), the new SB employs a heuristic to solve the single machine problem of minimizing the total weighted tardiness of jobs with release dates on a single machine (problem $1|prec; r_j|\sum w_j T_j$). Bottleneck selection is based on maximum lateness calculations in original SB, whereas the proposed SB makes use of total weighted tardiness evaluations.

All variations of the SB algorithm are developed based on the MB-JSS graph which is a modified disjunctive graph for minimizing total weighted tardiness (see Subsection 5.2.2 for more details). The framework of the SB algorithms that we propose is inspired by Pinedo and Singer (1999). However, there are major differences which are mentioned here. The main difference is based on the problem formulation and constraints which affect the solution method.

We solve a problem for MB-JSS graph which includes conjunctive, modified disjunctive and alternative arcs; whereas, Pinedo and Singer (1999) solve a problem for a modified disjunctive graph with conventional conjunctive, disjunctive and delayed precedence arcs. They use elimination rules to improve the heuristic search on an enumeration tree while we employ implication rules described in Subsection 5.3.2 in order to fix arcs which help to avoid cycles and solve the problem more effectively. Re-optimization and a backtracking technique are used in the study of Pinedo and Singer (1999) as control structure of the heuristic. In contrast, we try several variations of the SB algorithm with and without re-optimization, where re-optimization uses different schemes with regard to adding and deleting arcs and the number of re-optimization cycles.

It should be highlighted that the main feature of the train scheduling problem under study is the presence of the alternative arcs in the modified disjunctive graph. Alternative arcs help us to formulate the train separation according to network signaling system. Alternative arcs are chosen and fixed in the graph after adding modified disjunctive arcs resulted by the solution of the single machine problem. Subsequently, the other arcs are added as mentioned in the following. Inclusion of the alternative arcs in the problem formulation results in new features in the job shop scheduling problem which necessitates modifications in the SB algorithm proposed by Pinedo and Singer (1999). The required changes in the graph formulation and the suggested solution method are discussed in the following.

In what follows, we propose different variations of the SB algorithm which are different in solving the single machine problem. The first SB heuristic solves the single machine based on ATC rule. Because of this, we call it *SB-ATC* heuristic. Due to the similarity of the SB heuristics, we illustrate the general framework of the SB-ATC algorithm and employ it to define the other SB algorithms as below.

SB-ATC algorithm

- *Step 1*: Set the initial condition $\mathcal{M}_0 = \emptyset$. Graph G includes only conjunctive arcs.

- *Step 2*: For each $m \in \mathcal{M} \setminus \mathcal{M}_0$,

  Generate an instance of $1|prec; r_j| \sum w_j T_j$ and for each operation calculate Calculate r_{jm} by using equation (5.22) Calculate d^k_{jm} using equation (5.25)

  Select operation (j, m) with the highest index according to equation (5.27) to be processed as the next operation on machine m and add the disjunctive arcs corresponding to the operations sequence on machine m. t_m is the time that the machine becomes available, $\bar{p}_m$ is the average processing time of jobs assigned to machine m, and K is a scaling parameter whose value can be determined through computational tests.

  Compute

$$\bar{T}_m = \sum_{k=1}^{n} w_k \Big(\max_{(j,m) \in N_m} T^k_{jm} \Big), \quad T^k_{jm} = \max\{C_{jm} - d^k_{jm}, 0\} \tag{5.31}$$

  where N_m is the set of nodes corresponding to the operations processed on machine m.

- *Step 3*: Let

$$\bar{T}_{\max}(m') = \max_{m \in \mathcal{M} \setminus \mathcal{M}_0} (\bar{T}(m)) \tag{5.32}$$

  If $\bar{T}_{\max}(m') \geq 0$, determine machine m' as the bottleneck machine.

  If $\bar{T}_{\max}(m') = 0$, find j', k', and m'' such that

$$d'^{k'}_{j',m''} = \min_{j,k,m} d^k_{jm} \tag{5.33}$$

  for all $m \in \mathcal{M}$, $k \in \mathcal{I}$ and $j \in \mathcal{I}$ for the jobs processed on machine m. Determine machine m'' as the bottleneck machine.

  Solve the single machine subproblem for the bottleneck machine according to the sequence obtained in Step 2 for that machine and add corresponding disjunctive arcs, alternative arcs, static implications and simple implied arcs in Graph G.

If m' is the bottleneck machine, set $\mathcal{M}_0 = \mathcal{M}_0 \cup \{m'\}$. If m'' is the bottleneck machine, set $\mathcal{M}_0 = \mathcal{M}_0 \cup \{m''\}$.

- *Step 4*: Stop if $\mathcal{M}_0 = \mathcal{M}$, otherwise go to Step 2.

Within the SB solution process, arcs are added gradually to the problem through subproblem optimization step. We need to ensure that the added disjunctive and alternative arcs do not lead to infeasible solutions. Assume that machine m' is selected in the bottleneck selection step. The disjunctive arcs can be added based on the sequence of the jobs on machine m'. Consequently, we add an alternative arc from $(i, s_i(m'))$ to (j, m') if there is a disjunctive arc from (i, m') to (j, m').

The precedence constraints which are imposed by adding alternative arcs are used to derive implications. So the next step is to use the static implication rules to add implied alternative arcs for the following trains running on common blocks. Adding static implications is effective in making acyclic selection in the MB-JSS graph. A cycle in the graph is analogous to deadlock in the rail network and should be avoided in order to have a feasible schedule (see Subsection 5.3.2 for more discussions about static implications). The simple implied arcs are also added for all the following and opposite trains which correspond to the jobs processed on the same machines. Through this process, the main characteristics of a timetable to be conflict-free and deadlock-free can be achieved.

There may be cases that the proposed solution approach results in a infeasible solution. We try to avoid deadlocks in the network with the help of static implications which are added in the MB-JSS graph for only following trains after selecting alternative arcs. To the best of our knowledge, there is no approach in the literature to guarantee a deadlock-free solution. There are some discussions around this issue mentioned in Subsection 5.3.2 where two propositions by D'Ariano et al. (2007) are given. These propositions are used to add static implications.

We set K equal to 4 based on empirical results which is also compatible with the results that Vepsalainen and Morton (1987) report in their study for a suitable range of the look-ahead parameter K.

More variants of the modified SB algorithm are developed which sequence jobs in subproblem optimization step by using one of the suggested single machine heuristics in the following.

More variants of the modified SB algorithm

The other variants of the modified SB algorithm are similar to the SB-ATC algorithm and Step 2 is modified by a new criterion to solve the single machine subproblems. The first two variants of the SB algorithm are developed based on two well-known scheduling heuristics called *Active Schedule Generation* (ASG) and *Schrage scheduling* heuristic. Active schedule generation chooses the job with the smallest local due date $d^*_{jm} = \min_k d^k_{jm}$ among the potential candidates with

$$r_{jm} < \text{ECT} \tag{5.34}$$

where ECT stands for the Earliest Completion Time of the jobs to be scheduled next. *SB-ASG* algorithm employs this heuristic to solve the single machine problems.

The third SB algorithm is developed on the basis of the Schrage scheduling heuristic and is therefore named *SB-SCH*. SB-SCH solves the single machine problem by choosing the job with $d^*_{jm} = \min_k d^k_{jm}$, among the potential candidates with

$$r_{jm} \leq \text{EST} \tag{5.35}$$

where EST is the Earliest Starting Time of the jobs to be scheduled next.

Then, we try mixing ASG and ATC index rules for solving the single machine problem in another algorithm called *SB-ASG&ATC*. In order to solve the single machine problem, SB-ASG&ATC creates a subset $\mathcal{J}$ of jobs such that $\mathcal{J}= \{j|r_{jm} < ECT\}$ and sequences the job $j \in \mathcal{J}$ with the biggest ATC index.

The other algorithm is *SB-ASG-II* which is a modification of SB-ASG algorithm that selects the job with the smallest local due date $d^*_{jm} = \min_k d^k_{jm}$ among the potential candidates with

$$r_{jm} < (\text{ECT}+\text{EST})/2 \tag{5.36}$$

SB-ASG&ATC-II algorithm modifies SB-ASG&ATC to choose the job $j \in \mathcal{J}$ with the biggest ATC index, where $\mathcal{J}= \{j|r_{jm} < ECT + EST)/2\}$.

In this subsection, the main algorithms for the SB algorithm are shown which differ from each other in terms of the solution method for the single machine problem. More variants of the main SB algorithms based on different re-optimization steps are suggested below.

Re-optimization procedure

The next variations of SB algorithm incorporate a re-optimization step to the proposed SB algorithms based on the discussions above in this subsection. It should be noted that re-optimization in these versions is performed according to the same criteria that optimization is performed; e.g. the algorithm which sequences jobs according to ASG, performs re-sequencing according to ASG as well. The only exception is *SB-ASG-ReOpt* which does sequencing based on ASG and re-sequencing based on ATC index. This version is different from the other variants with re-optimization as it employs two different criteria, ASG for sequencing and ATC index for re-sequencing. Whereas, the other versions of SB algorithm with re-optimization use the same criteria for sequencing and re-sequencing steps.

Assuming that machine m' is sequenced, in order to re-optimize machine $l \in \mathcal{M}_0 \setminus \{m'\}$, all corresponding disjunctive, alternative, static implications and simple implied arcs of machine l should be deleted from the MB-JSS graph. The order of re-optimizing machines of $\mathcal{M}_0 \setminus \{m'\}$ is given by $l(1), \ldots, 1(p)$, $p = |\mathcal{M}_0| - 1$, where $l(i)$, $i = 1, \ldots, p$ is ordered according to the decreasing solution value of the sequencing problem.

It is observed that after some iterations, no bottleneck machine is selected based on the equation (6.24) as $\bar{T}_{\max}(m') = 0$ and the bottleneck machine is selected according to the equation (5.33). In this case, no re-optimization is performed after selecting such a machine.

It should be noted that we explore modified SB algorithm versions where worse solutions are not rejected. It is due to our observation that there are SB algorithm variants with better final solutions although worse partial solutions, obtained after re-optimization, are not rejected. While solving the algorithm, in the intermediate steps, we often need to let the solution go worse before it gets better. Therefore, we keep the SB algorithm variants where in the re-optimization process worse solutions are not rejected.

Another variant of re-optimization process is tried on the mentioned SB algorithms based on rejecting bad solutions. Assume f_0 to be the objective function value (calculated as $\sum w_j T_j$) after sequencing machine m' and f_l to be the objective function after re-sequencing machine l where $l \in \mathcal{M}_0 \setminus \{m'\}$. If $f_l > f_0$, machine l is not re-sequenced. An example for naming this version of the algorithm is *SB-ATC-ReOpt-Rej* which includes re-optimization with rejecting worse solutions to the main SB-ATC Algorithm.

The next family of SB algorithms is proposed for a re-optimization scheme inspired by Adams et al. (1988) and applied on all mentioned varaints of SB algorithms. Assuming $\mathcal{M}_0$ to be the set of machines already sequenced in the current iteration, we go through three re-optimization cycles as long as a tardy machine is found as mentioned above. A re-optimization cycle is defined in the following. For $i = 1, \dots, p$, re-sequence the corresponding single machine problem for machine $l \in \mathcal{M}_0 \setminus \{m'\}$ where m' is the newly sequenced machine . In other word, re-optimization is repeated two additional times compared to previous variants of SB algorithm.

5.4 Computational results

In this section, we discuss a real-world implementation of the proposed EA heuristic, MILP and SB algorithms. The experiments are based on the London Bridge area in Kent, South East of the UK, which is chosen because of its dense and complicated network of interconnected lines for passengers in and out of London, East Sussex and the Channel Tunnel. We consider a partial network including London Bridge area which is a bottleneck in Kent (See Section 3.6 for more details and maps of the area). The partial network is about 15 km long and it includes busy stations such as London Charing Cross, London Waterloo, London Cannon Street, New Cross and Deptford with a total of 28 platforms.

Using this data we simulate real-life traffic conditions under different types of disruptions in the network by perturbing the known running times on certain blocks and late departure of the trains from their origin.All SB algorithms are developed in MS Visual C++ 2010 and run on a PC with a dual core, 3.00GHz and 4GB RAM. The Xpress-IVE (FICO, 2013) optimization package has been used for solving the MILP model and executed on the same PC.

As the developed MILP involves big M, the performance of the Xpress depends on the choice of M. Large values of M can cause the solver to make slow progress solving the MILP model. If a very big value is considered for M without any thought, the constraint works in theory, but it can result in models which can feature very bad numerical behaviour. The linear relaxations which are used in solving the MILP can be very weak leading to excessive branching and increased computation time. Therefore, smaller value of big M is more desirable as it can make the computation time shorter. Camm et al. (1990) provide computational results which focuses on the use of M in an example of lot sizing model and they show the dramatic effect of reducing big M.

In order to find the value of big M in our MILP model, we have performed some initial experiments to reduce the value of big M. The value of big M has been selected so that the computation time is reduced. Initial experiments has shown that the computational time is not sensitive to the value of the chosen big M. It also seems to be difficult to find a specific value of big M such that it can be used in each of the constraints.

5.4.1 Test instances

We consider passenger timetable and timetable components information for 2010 on the Kent area where Southeastern is the main operator. General characteristics of the test instances such as the number of jobs and machines are mentioned in Section 3.4 for the train scheduling problem. Our experimental data focus on off-peak period commuters. The timetable cycles every 30 minutes for the passenger trains and includes 27 trains.

According to Southeastern *Service Disruption Procedure (SDP)*, disruptions are classified into three types as follows. A *minor disruption* is where no individual delay is more than 15 minutes. A *general disruption* is where multiple services are running with delays between 15 to 30 minutes. A *major disruption* is where the majority of train services are delayed by over 30 minutes. It should be noted that SDP is a property of Southeastern company and subject to a non-disclosure agreement.

There is no exact number for disrupted trains in this classification. The only specifications that can be derived for the number of trains are that general disruption affects multiple services and a major disruption is when the majority of services are affected. Therefore, we try to introduce appropriate ranges according to the given definitions for each class. As we have 27 trains in a cycle in total, it is reasonable to generate minor disruption by perturbing 1 to 5 trains, general disruption by perturbing 6 to 12 trains and major disruption by perturbing 13 to 27 trains.

In the experiments, we generate 4 data sets randomly where each data set includes several instances for different scenarios of disruptions in terms of extended running/dwell times on blocks and late departures of the trains from their origin. For each data set, instances differ from each other in terms of the set of disrupted blocks, disrupted trains and amount of disruption. Block delays and departure delays are shown in minutes for extended running/dwell times on blocks and late departures of the trains respectively. Computational experiments compare the total delay of the schedule resulting from the Xpress, EA heuristic and SB algorithms listed as SB-ATC, SB-ATC-ReOpt, SB-ATC-ReOpt-Rej and SB-ASG-ReOpt.

Table 5.1 describes the first set of experiments for longer running/dwell times where the running/dwell time of a number of trains are extended by a specific amount. This set of data is made up of instances with smaller delays including 6 minor and 6 general disruption instances. Minor disruption delays are chosen in the set $\{5, 10, 15\}$ and general disruption delays are chosen in the set $\{20, 25, 30\}$. The values for mentioned block delays for minor and general disruption are shown in minutes and they are defined in the relevant range for each type of disruption. We look at extended running/dwell times on one and two different blocks. Number of affected trains in minor disruption instances is 3 or 5 and this number is 9 or 12 for general disruptions. There are 6 instances in each category of disruption.

TABLE 5.1: Longer running/dwell times - minor and general disruption

Disruption Type	# of Instance(s)	Block Delay	# of Block(s)	# of Train(s)
Minor	6	5, 10, 15	1, 2	3, 5
General	6	20, 25, 30	1, 2	9, 12

Second set of experiments in Table 5.2 defines instances for late departure time of a specific number of trains from their origin. Similar to the first set, smaller amounts of delay are introduced in 9 minor and 9 general disruption instances. Minor disruption affects trains for a delay value chosen in the set $\{5, 10, 15\}$ and general disruption create delays in the set $\{20, 25\}$. The values in each set for delayed departure are defined in minutes. Number of delayed trains is selected from $\{1, 3, 5\}$ for minor disruption and $\{6, 9, 12\}$ for general disruptions.

TABLE 5.2: Late departures - minor and general disruption

Disruption Type	# of Instance(s)	Departure Delay	# of Train(s)
Minor	9	5, 10, 15	1, 3, 5
General	6	20, 25	6, 9, 12

In Table 5.3, we consider longer running/dwell times with larger values of delay on different number of blocks which perturbs all 27 trains in a cycle. There are 15 instances of major disruption which are created for different amounts of delay in set $\{40, 50, 60, 70, 80\}$. Block delay are defined in minutes. Set $\{4, 5, 6\}$ show the different number of blocks that are affected by delays.

TABLE 5.3: Longer running/dwell times - major disruption

Disruption Type	# of Instance(s)	Block Delay	# of Block(s)	# of Train(s)
Major	15	40, 50, 60, 70, 80	4, 5, 6	27

Last Table 5.4 summarizes 16 major disruptions instances with both longer running/dwell times and late departures. They are categorized as major disruptions as block delays are 40 or 50 minutes occurring on 4 or 6 blocks, which affects the majority of services although departure delays are less than 25 minutes. It should be emphasized that introducing departure delays bigger than 25 minutes results in shifting trains to run mostly out of the original cycle. This means that the train has no conflicting trains to compete for resources and the instance is not interesting anymore. Thus, we avoid generating any departure delay bigger than 20 minutes. So delayed departures of 10 and 20 minutes are tried for different number of trains in set $\{3, 5, 9, 12\}$.

Table 5.4: Longer running/dwell times and Late departures - major disruption

Disruption Type	# of Instance(s)	Block Delay	# of Block(s)	Departure Delay	# of Train(s)
Major	8	40	4, 6	10, 20	3, 5, 9, 12
	8	50	4, 6	10, 20	3, 5, 9, 12

In the Subsection 5.4.2, the performance of different solution methods are evaluated for the generated instances.

5.4.2 Analysis and comparison of the methods

According to some initial computational experiments, we have identified SB algorithms with better performances among the others; SB-ATC, SB-ATC-ReOpt, SB-ATC-ReOpt-Rej and SB-ASG-ReOpt. It should be noted that the re-optimization in SB-ATC-ReOpt is performed based on ATC index as well as the re-optimization in SB-ASG-ReOpt. *SB-ATC-ReOpt-Rej* algorithm follows the re-optimization scheme which rejects worse solutions. In the following, computational results for the proposed SB algorithms are discussed.

The following tables detail the total delay of the scheduling arising from the approaches described in Subsection 5.3.5, where SB algorithms SB-ATC, SB-ATC-ReOpt, SB-ATC-ReOpt-Rej and SB-ASG-ReOpt has been chosen among several SB variants based on some initial experiments.

Tables 5.5, 5.6, 5.7 and 5.8 summarize the result of our experiments for Xpress, EA heuristic and SB algorithms separately for each data set mentioned in Subsection 5.4.1. The first column in each table shows the solution methods. The second, third and fourth columns in each table show the performances of all instances.

For some instances, the value of delay is not available due to the SB algorihms or EA heuristic resulting in deadlock or Xpress giving no answer in the determined time limit. If there is any instance resulting in deadlock or no answer, we exclude its results across all solution methods in the performance calculations in columns five, six and seven which are all under heading all instances excluding unsolved instances.

In table 5.6, all instances are solved successfully and there is no unsolved instances due to either deadlock or no answer. Last two columns in all tables show average runtime and percentage of feasible solutions for each solution method across all instances of the corresponding data set respectively. The percentage of feasible solutions corresponds to the instances which does not result in either deadlock by the heuristics or no answer by Xpress.

In each table, all instance results list the average delay (Avg Delay) for each solution method across all instances of the corresponding data set per train, average *Relative Deviation (RD)* across all instances firstly with respect to the EA heuristic (Avg RD wrt EA Heauristic) and secondly with respect to Xpress results (Avg RD wrt Xpress). Dalay is equal to tardiness and total delay is analogous to total weighted tardiness as it is mentioned in Section 3.4. In the case study, we have unit weights. So total delay is equal to total tardiness. Let an instance be denoted by a, its delay for instance a be denoted by T_a and the number of trains for instance a be denoted by n_a, then average delay is computed as below

$$\sum_a (T_a/n_a)/A \tag{5.37}$$

where A shows the total number of instances. Average delay is shown in minutes.

It should be noted that the relative deviation is calculated with the following formula

$$RD = \frac{baseline \ \ solution - algorithm \ \ solution}{baseline \ \ solution} \times 100 \qquad (5.38)$$

where the baseline solution is considered as EA heuristic and Xpress. A large value of RD reflects a good performance with respect to EA Heuristic or Xpress.

Table 5.5, 5.6,5.7 and 5.8 show an improvement in the average total delay of SB algorithms and Xpress relative to the EA heuristic provided result for all instances and results that exclude unsolved instances. This is expected since there is no attempt by the EA heuristic to reduce total weighted tardiness. When considering RD with respect to the EA heuristic for each data set, it seems that the EA heuristic has the worst performance compared to all the other methods for data set 5.1.

Table 5.5 and 5.6 show that the MILP model outperforms both the EA heuristic and SB algorithms other than one exception (726.65 ¡ 747.64). Xpress performs better considering the results for data sets for longer running/dwell times described in Table 5.1 in comparison to late departures in Table 5.2, because the average RD with respect to Xpress has bigger values for the former compared to the latter group of data. In both data sets 5.1 and 5.2 for longer running/dwell times and late departures respectively, where smaller perturbations are involved and only minor and general disruptions incur, Xpress can find feasible solutions in 100% of instances and produce schedules with less average delays.

Table 5.5: Longer running/dwell times - minor and general disruption: Average Delay and RD's

Solution Method	All Instances			Excluding the Unsolved Instances			Avg Runtime (secs)	% of Feasible Solutions
	Avg Delay per Train	Avg RD wrt EA Heuristic	Avg RD wrt Xpress	Avg Delay per Train	Avg RD wrt EA Heuristic	Avg RD wrt Xpress		
Xpress	27.69	**42.29%**	-	**10.41**	**42.63%**	-	400	**100%**
EA heuristic	33.52	-	−85.17%	18.07	-	−86.10%	2352	77%
SB-ATC	33.79	29.29%	−26.78%	12.03	29.90%	−25.52%	**349**	**100%**
SB-ATC-ReOpt	30.66	34.03%	**−16.37%**	11.71	32.64%	**−20.71%**	411	**100%**
SB-ATC-ReOpt-Rej	**26.91**	33.81%	−20.90%	11.74	32.12%	−21.80%	402	92%
SB-ASG-ReOpt	31.51	21.17%	−33.53%	13.31	21.87%	−41.77%	416	77%

Considering the average delay column of Table 5.5 for all instances, SB-ATC-ReOpt-Rej seems to have the smallest average delay, it is obvious that Xpress has a better result when considering average delay values for all instances excluding unsolved instances. It is due to the fact that Xpress solves the problem for 100% of instances and SB-ATC-ReOpt-Rej gives an answer only for 92% of instances. It is compatible with the average RD values with respect to Xpress that show the superiority of Xpress by considering either all instances or all instances excluding unsolved instances.

Table 5.6: Late departures - minor and general disruption: Average Delay and RD's

Solution Method	All Instances			Avg Runtime (secs)	% of Feasible Solutions
	Avg Delay per Train	Avg RD wrt EA Heuristic	Avg RD wrt Xpress		
Xpress	**5.92**	**17.66%**	-	400	**100%**
EA Heuristic	7.00	-	−22.83%	2442	**100%**
SB-ATC	6.63	6.23%	−14.27%	**346**	**100%**
SB-ATC-ReOpt	6.28	11.25%	**−7.92%**	410	**100%**
SB-ATC-ReOpt-Rej	6.28	10.96%	−8.43%	409	**100%**
SB-ASG-ReOpt	6.43	10.02%	−9.50%	402	**100%**

As expected, Xpress outperforms the other algorithms in terms of average delay. Improvements in the schedule suggested by Xpress confirm the advantage of MILP model within the strict time limit imposed by real-time application. It should be noted that Xpress is restricted to 400 seconds which is within the time frame of 15 minutes determined in SDP for the incident/service update.

It should be noted that SB-ATC-ReOpt algorithm has relatively the best performance among the SB algorithms in both 5.5 and 5.6 with respect to Xpress results when unsolved instances are excluded. Both SB-ATC-ReOpt and SB-ATC consistently result in feasible solutions in 100% of instances.

The complete results for longer running/dwell times - minor and general disruption are shown in Tables A.1, A.2 and A.3 in Appendix A. Results for individual instances in Tables A.1 indicates that Xpress performs better for smaller delays, in particular for minor disruptions it can find optimal solutions in 3 cases. As disruption becomes bigger, it seems that instances become harder to solve; more specifically for EA heuristic as it results in deadlocks for 3 instances. Moreover, Tables A.4, A.5 and A.6 in Appendix A provide the comprehensive results for late departures - minor and general disruption.

We design another group of data sets where bigger delays are involved and instances can be categorized as major disruption. The MILP model does not perform as well for major disruptions in data sets in Tables 5.3 and 5.4 where there are only longer running/dwell times or both Longer running/dwell times and Late departures. The percentage of feasible solutions found by the MILP model reduces to 69% and 88% in the given time limit; whereas, SB-ATC and SB-ATC-ReOpt algorithms can still find feasible solutions for 100% of instances. SB-ATC-ReOpt-Rej algorithm still performs better than Xpress with regard to finding more feasible solutions although its 100% performance for the data set in Table 5.4 decreases to 81% for the data set in Table 5.3.

TABLE 5.7: Longer running/dwell times - major disruption: Average Delay and RD's

Solution Method	All Instances			Excluding the Unsolved Instances			Avg Runtime (secs)	% of Feasible Solutions
	Avg Delay per Train	Avg RD wrt EA Heuristic	Avg RD wrt Xpress	Avg Delay per Train	Avg RD wrt EA Heuristic	Avg RD wrt Xpress		
Xpress	383.35	13.11%	-	295.41	13.11%	-	400	69%
EA heuristic	**367.86**	-	−15.17%	335.34	-	−15.17%	2403	31%
SB-ATC	453.53	6.58%	−6.73%	292.76	11.41%	−2.07%	**352**	**100%**
SB-ATC-ReOpt	407.12	13.20%	**1.47%**	292.42	12.85%	−0.37%	413	**100%**
SB-ATC-ReOpt-Rej	417.92	10.99%	−0.24%	293.27	12.61%	−0.65%	426	81%
SB-ASG-ReOpt	384.12	**14.61%**	−0.03%	**289.72**	**13.19%**	**−0.02%**	415	56%

The most significant result is the quality of the solutions provided by SB-ATC-ReOpt which shows an improvement with regard to Xpress. All RD values of SB-ATC-ReOpt algorithm with respect to Xpress for data sets in Tables 5.3 and 5.4 are positive considering the results for all instances and results excluding unsolved instances. There is an exception with a negative RD for SB-ATC-ReOpt with respect to Xpress. We have a negative value of −0.37% when unsolved values are taken out of the calculation performances for average delay with respect to Xpress. This is due to the fact that many better solutions of SB-ATC-ReOpt are excluded because the other algorithms do not have a solution. This can be easily observed in Table A.7 in Appendix A by checking the results for the individual instances.

In addition, RD values with respect to Xpress for major disruptions instances in Tables 5.3 and 5.4 are smaller than these values for instances of minor and general disruptions in Tables 5.2 and 5.1. It indicates that the quality of the MILP model solution reduces as the disruption becomes bigger.

Generally the percentage of feasible solutions for Xpress and EA heuristic in two former tables are also less compared to the two later tables. It appears that EA heuristic has the weakest performance compared to the other algorithms as deadlocks arise in many instances and the percentage of feasible solutions reduces dramatically in major disruptions. In contrast, SB algorithms seem to have relatively very high percentage of feasible solutions in comparison with both EA heuristic and Xpress when major disruptions in Tables 5.3 and 5.4 are considered. Hence, the reliability of the SB algorithms and specifically SB-ATC-ReOpt is substantial in dealing with bigger disruptions.

Table 5.8: Longer running/dwell times and Late departures - major disruption: Average Delay and RD's

Solution Method	All Instances			Excluding the Unsolved Instances			Avg Runtime (secs)	% of Feasible Solutions
	Avg Delay per Train	Avg RD wrt EA Heuristic	Avg RD wrt Xpress	Avg Delay per Train	Avg RD wrt EA Heuristic	Avg RD wrt Xpress		
Xpress	311.63	11.61%	-	265.44	11.61%	-	400	88%
EA heuristic	306.74	-	−13.64%	298.92	-	−13.64%	2501	41%
SB-ATC	346.62	7.45%	−10.89%	280.40	6.90%	−5.25%	**367**	**100%**
SB-ATC-ReOpt	**304.95**	**11.72%**	**1.85%**	**263.67**	**11.82%**	**0.25%**	415	**100%**
SB-ATC-ReOpt-Rej	316.92	10.93%	−1.65%	267.22	11.05%	−0.61%	417	**100%**
SB-ASG-ReOpt	313.15	11.69%	−3.18%	265.81	11.28%	−0.46%	419	71%

The complete results for longer running/dwell times - major disruption are represented in Tables A.7, A.8 and A.9 in Appendix A. Also, Tables A.10, A.11 and A.12 in Appendix A offer the comprehensive results for longer running/dwell times and late departures.

Comparing across different approaches, SB algorithms show a consistent average runtime of about 400 seconds. Hence, a time limit of 400 seconds is set for solving the MILP model. As mentioned earlier, this time limit is well within the SDP time frame of 15 minutes for incident/service update. EA heuristic has an average runtime of about 2400 seconds which makes it less attractive. The long runtime of the EA heuristic is due to the fact that we look at only one pair of modified disjunctive arcs in an iteration and consequently the corresponding longest path calculations are carried out in each iteration.

The ability of the MILP model to obtain better solutions for minor and general disruptions makes it a useful tool for small disruptions. Nevertheless, the quality of the schedules reduces as disruptions become bigger. The results for major disruptions in Tables 5.7 and 5.8 show that SB-ATC-ReOpt provides schedules that improve over Xpress, EA heuristic and the other SB algorithms for average delay and provides in feasible solutions 100% for all instances. Therefore, SB-ATC-ReOpt can serve as a good approach to not only provide better solution quality in the operational framework, but also to provide more feasible solutions compared to the other methods.

5.5 Summary

In this chapter, the train scheduling and rescheduling problems are modelled in a microscopic level. The problem is formulated as a job shop scheduling problem with additional constraints which adapts the disjunctive graph to represent train safety and operational constraints in a so-called MB-JSS graph. A generic MILP model is formulated that extends existing models by considering more realistic and detailed constraints from the UK railway industry. The proposed mathematical formulation for train scheduling and rescheduling has significant computational advantages that shows the model's ability to solve real-world problems. A special feature of our formulation is its flexibility in modeling a different rail network with a fixed block signaling systems.

Moreover, we describe a novel optimization framework based on the SB procedure which decomposes the job shop scheduling problem to single machine scheduling subproblems. Several variants of the SB algorithm are suggested that employ different characteristics with respect to solving this problem. The most efficient SB algorithms are identified through initial experiments.

Our computational experiments focus on a section of the UK rail network that is dense, complicated and congested. It provides a problem instance that is among the most computationally difficult job shop problems where the graph is extremely large. It is clear that simply finding a feasible solution is nontrivial, since FCFS frequently results in a deadlock. It should be noted that EA heuristic which is an algorithm developed to avoid acyclic sections in the MB-JSS graph can easily run into infeasibility and hence deadlocks when bigger disruptions are considered. Also, EA heuristic has a long runtime. Therefore, the proposed optimization framework for SB algorithms, which can usually find feasible solutions for all instances, is very promising to model and solve this large and complex problem with all the practical constraints.

Computational experiments looks at different scenarios for delays by extending running/dwell times and departure delays for train in the case study. More specifically, computational results show that the MILP model finds better solutions in comparison to the SB algorithms and EA heuristic when network delays are small. When delay becomes bigger, the ability of MILP model and EA heuristic deteriorates in terms of feasibility and solution quality. For larger disruption, a special version of SB called SB-ATC-ReOpt provides significantly better schedules than other approaches. It is also shown to be more reliable in terms of finding feasible solutions for restricted real-time decisions which makes it viable for real-world applications. Generally, it appears that when trains are disrupted more severely, the MILP model and EA heuristic are prone to infeasibility and hence deadlock; whereas, SB algorithms result in more feasible solutions and its quality increases relative to the other methods.

Further research to improve the solution time and quality of the algorithm includes investigating more efficient heuristics that can be embedded in the current SB framework and exploiting potential computational speedups. Further research should be addressed to design of the SB algorithm in order to produce better solutions even for smaller amounts of delay. Developing faster algorithms can be fruitful in integrating the scheduling process with other steps in train planning.

Chapter 6

Train routing and disruption management

In order to achieve higher level of efficiency and providing better service and safety, planning the proper time slot and the proper tracks for trains becomes critical. Allocating time slots to trains is the concern of timetabling problem and assigning tracks to trains is dealt with in train routing problem. To increase the performance of the railway system, timetables are developed and routes are defined in advance and they are modified in real time to handle disruptions.

It is clear from the above discussion that train routing problem is very important for efficient railway planning and different variants of it is applicable in strategic, tactical and operational levels (Lusby et al., 2011). In strategic planning, train routing problem answers questions about the capacity of the network whereas in tactical level train routing problem checks the compatibility of the timetable with the infrastructure layout. Finally in operational level, it deals with rerouting trains while some routes are infeasible or other routes are more desirable due to disruption.

In this chapter train routing problem is addressed in tactical and operational level. We try to minimize delay propagation in the network by proposing alternative routes for trains in advance planning and reroute the trains in respond to disruptions. The train routing and rerouting problem is formulated as a parallel machine job shop scheduling problem in Section 6.2. A generic mixed integer programming model of the problem is proposed for the train routing and rerouting problem in Section 6.2.1 which can be adapted for different rail networks. Then, a modified disjunctive graph is presented for the suggested parallel machine job shop scheduling problem in Subsection 6.2.2.

Furthermore, a novel algorithm is developed and implemented based on the parallel machine job shop scheduling to solve the train routing and rerouting problem in Section 6.3. We explore the performance of the suggested algorithm with a real-life case study of the Kent area which is discussed earlier in Section 3.6. Analyses of the mentioned critical corridor with a complex infrastructure and congested traffic indicates the computational advantage and viability of the method in Section 6.4. The instances are of a practical size for London Bridge area including single and parallel tracks and trains travelling in one and opposite directions. We conclude the chapter with a summary of our observations and further studies to improve the problem formulation and enhance the algorithm in Section 6.5.

6.1 Introduction

Train routing is an indispensable component of planning process as no time table can be effective without considering train routes. In particular, train routing decisions becomes more critical in dense networks. As railway stations and junctions are the most important part of the network with regards to capacity and delay, addressing robust routing in these bottleneck areas can increase both capacity and punctuality. In this context, robustness is regarded as railway system insensitivity to small disturbances during daily operations (Hansen and Pachl, 2008).

After a plan is created in the timetabling phase for an aggregated network, the detailed layout of the network is considered in order to have a feasible routing through the nodes defined in the aggregated level. It is called *train routing* problem and it is necessary to adjust the timetable with the detailed topology as the nodes are usually consisted of many interconnected tracks that merge and split. Thus, the aim of the train routing problem is to assign conflict-free paths over time for the traversal of the trains in a timetable while satisfying several operational constraints and rules.

If a node is an station, this problem is referred to as *train platforming* because routing allocates trains to the platforms. A route can be defined as consecutive block sections that a train enters, traverses and leaves. If the route goes through a station, it may include stopping at an available platform. If the selection of a platform uniquely defines the route into and out of the station the problem is a platforming problem. Whereas, routing problem is about selecting between available alternative routes which can reach a special platform (Lusby et al., 2011).

This study is directed towards routing trains in a partial network with a given timetable in tactical and operational level. The benefit of formulating and solving the train routing problem by the proposed approach is two fold. One is to give the planner the opportunity to choose between alternative routes and select the combination of routes for trains so that it results in the minimum delay. The other advantage is that the same model can be used to reroute the trains in case of disturbances and replace the blocked or time-consuming routes with better ones in order to minimize the knock-on effect of delay on the partial network. A novelty of the model is that train routing and scheduling are integrated and a route plan gives train orders and timing on the suggested route. This feature of the model is quite desirable in real-life application.

It should be mentioned that delay is permitted for late trains and train can wait on a track section until the subsequent one becomes available. So the problem can be used in operational level. Whereas no wait is allowed in the models that are focused on the capacity assessments which are related to strategic level planning. There may not be many routing options to and from the platforms; however, the platforming is also relevant in this problem as it is still required to make sure that the allocation of the platforms does not have any conflict.

6.2 Mathematical formulation

As it is discussed in Chapter 5, we consider a railway network including stations linked together by tracks which are divided into block sections. We use a new concept in this chapter which is called track section. A track section is a division of a track which is determined from a TIPLOC to a consecutive TIPLOC.

TIPLOC is an abbreviation used in the UK railway network and stands for Timing Information Point LOCation. Train planners use TIPLOCs in train schedules to define what time trains should arrive, depart or pass a particular point in the network. A TIPLOC is a timing point and it may be but not necessarily a station, part of a station, siding, signal, depot, junction, etc. This means that there may be a TIPLOC at a station, part of a station, siding, signal, depot, junction, etc, but it is not always the case. The railway network in this study uses this concept and it includes several single and multiple track sections similar to the simple network shown in Figure 6.1.

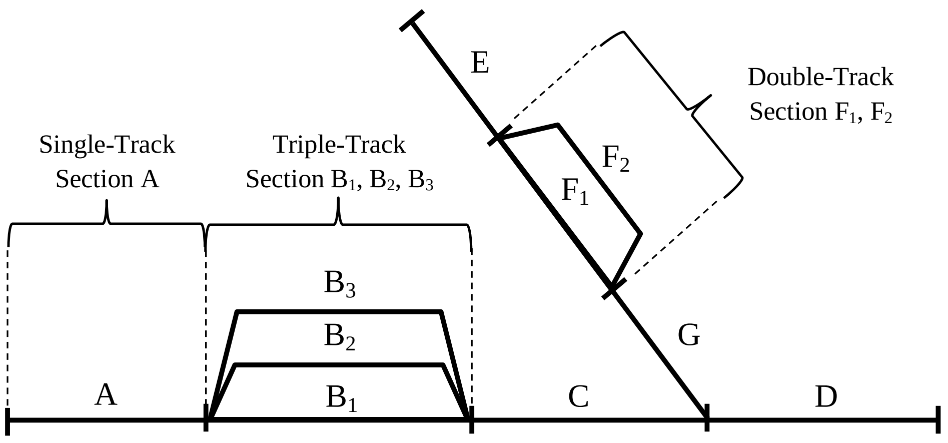

Figure 6.1: A simple network with single and multiple parallel tracks

Operational characteristics of the train routing is similar to train scheduling problem discussed in Chapter 5. However, essential terminology differs slightly and needs to be clarified particularly in perspective of this chapter for train routing problem. It should be emphasised that the track section between two consecutive TIPLOCs is shortly referred to as a track section.

In this chapter, the following operational characteristics are considered. The *running time on a track section* corresponds to the minimum time required for the traversal of the train on a track section and for the sake of simplicity it is shortened as *running time* in this chapter. *Headway* is the minimum time between two consecutive trains to keep them far enough for safety reasons. The minimum time that a train needs to stop at a platform is called *dwell time*.

Dividing tracks according to TIPLOCs is helpful for defining the problem. Firstly, data is accurate and in line with how timetable components are defined in the UK. Secondly, it is an efficient way of aggregating information which results in a reduction in the number of variables and constraints. Thirdly, the single and multiple track sections arising from this partitioning help in problem formulation which is discussed in the following.

Although this aggregation is very convenient in terms of reducing the problem size, it does not oversimplify the problem such that main operational data is lost. For instance, if the whole path of a train on a single track is considered as a single track section, the capacity of the line is incorrectly restricted to one train at a time.

The railway network in this study is such that dividing the tracks according to the TIPLOCs is compatible with the characteristics of the physical layout. That is, TIPLOCs in our partial network are located at the points that the infrastructure changes the track layout. Moreover, splitting tracks with regard to TIPLOCs is in line with the operational rules. The running time of train on the track sections between two consecutive TIPLOCs is smaller than or equal to the headway value. Thus, it can be claimed that only one train can travel on a track section at a time as two consecutive trains are separated by the headway.

Considering the above argument, we can find similarities between the train routing problem and job shop scheduling problem. We can consider a *stage* consisting of *single* or *parallel machines* as a track section including single or parallel tracks between two consecutive TIPLOCs and a job is analogous to a train. Traversal of a train on a track section is similar to the operation of a job on a stage of single or parallel machines. Moreover, only one train can travel on a track section at a time which is similar to the job shop principle where a machine can process only one job at a time.

In the partial network which is concerned in this research, multiple tracks of a track section have similar running times. This is analogous to parallel machines in the job shop. The train moves from one track section to the next and the track section should keep the train until it moves to the proceeding track section. Consequently, the train routing problem is modelled as a parallel machine job shop scheduling problem.

Similar to the scheduling problem in Chapter 5, a cyclic timetable is considered with a cycle length of 30 minutes which means that the frequency of the same service to be repeated is half an hour. In this problem information is aggregated and a track section consists of a couple of blocks. On the other hand, network information is not aggregated in the form of a compact macro topology of stations and parallel tracks between stations. Hence, the problem can be categorised as a mesoscopic level planning.

The aim is to explore the use of alternative routes so that the delay is minimized. Thus, the objective function is to minimize the total weighted tardiness of jobs in the parallel machine job shop scheduling problem. Operational constraints consist of running time, dwell time, headway and blocking constraints as discussed earlier. Therefore, we call this problem as *Modified Parallel Machine Job Shop Scheduling* problem abbreviated as *MPM-JSS*.

The suggested routing plan should have two essential characteristics including *conflict-freeness* and *deadlock-freeness*. A *conflict* happens in this routing problem when more than two train claim the same track section. A *deadlock* happens for a set of trains when each train claims a track section ahead which is not available as it is either occupied by another train or blocked due to a disruption. Figure 6.2(a) shows a deadlock in the network and Figure 6.2(b) presents a deadlock-free situation.

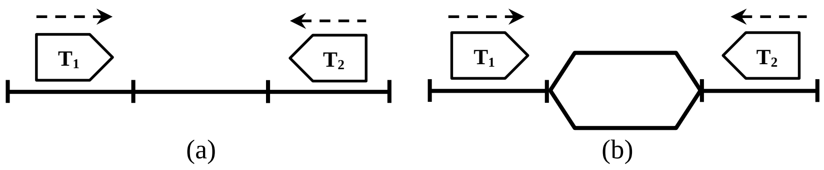

FIGURE 6.2: (a) Situation with a deadlock (b) Situation with no deadlock

6.2.1 MILP model

We have shown the disjunctive programming formulation for the classical job shop scheduling problem in Chapter 5. In this section, we extend the disjunctive programming formulation to incorporate particular properties of the the train routing problem. Hence, train routing problem is modeled as a parallel-machine job shop scheduling problem which is inspired by Liu and Kozan (2009).

Liu and Kozan (2009) propose a Mixed Integer Programming (MIP) model with nonlinear objective function and constraints in order to minimize maximum completion time of the jobs or makespan. Minimizing makespan can be useful when a schedule is developed. However, in this study, we are concerned with delay measurement which is more important in the tactical and operational level. A schedule is more robust when delay propagation is taken into account. In addition, we are interested in a linear model. Hence, we suggest a MILP model to minimize total weighted tardiness.

Introducing buffer time in the timetable can help in absorbing the delays when a disruption happens. However, in this study we look at a working timetable with fixed components as enough buffer times in the timetable has already been incorporated by the train operating company. We need to explore if there is any room for improvement through more efficient ordering and routing of the trains. Therefore, we propose minimizing the total weighted tardiness as the objective function in order to minimize delay propagation. The operational and safety constraints are also modeled such that the problem can be solved with job shop scheduling techniques. We have called this problem earlier as modified parallel machine job shop scheduling problem (MPM-JSS) in Section 6.2.

Hence, the objective function of MPM-JSS is different from the classical job shop scheduling problem with the makespan. There are also additional constraints in the MPM-JSS problem to represent operational and safety constraints for the train routing problem. For a comparison, we refer the reader to Subsection 5.2.1 for the classical job shop scheduling problem $J||C_{\max}$ and its mathematical programming formulation.

It has been observed in the Chapter 5 that one can minimize delay in the railway network by adjusting the order and timing of the trains in a timetable with fixed routes in both offline and online planning mode. Another approach to manage delay is changing train routes in addition to adjusting train time and order. This process can be carried out when the routes in a timetable are created in advance or in real-time. This section offers a mathematical programming formulation for the train routing problem which can be modeled as a MPM-JSS problem.

Given an input timetable with alternative routes defined as different stages of single and parallel machines, a schedule determines selected track sections with the starting times of trains entering them and the order of trains on them. Therefore, it can be observed that this approach is applicable to solve a combined train routing and scheduling problem at the same time in a tactical and operational level. We use the following notation for the parameters in the MILP model for the train routing and scheduling problem.

$\mathcal{I}$:	set of jobs/trains
i,j:	indices for jobs ($i = 1, \ldots, I$ and $j = 1, \ldots, J$)
r_i:	non-negative release time of job i/departure time of train i from its origin
d_i:	non-negative due date of job i/scheduled arrival time of train i at its destination
w_i:	non-negative importance weight of job i/train i
l_i:	number of stages to be visited by job i/number of track sections to be traversed by train i
$(s_{i1}, \ldots, s_{i,l_i})$:	sequence of stages to be visited by job i/sequence of track sections to be traversed by train i
$\mathcal{S}$:	set of stages/track sections
$\mathcal{M}_s$:	set of machines for stage s/set of tracks for track section s
$\mathcal{O}$:	set of operations defined by indices (i, m, s), for $i \in \mathcal{I}$, $s \in \mathcal{S}$ and $m \in \mathcal{M}_s$
p_{ims}:	operation time for job i on machine m of stage s/running time for train i on track m of track section s
h_{ijms}:	required time delay (headway) between the start of operations (i, m, s) and (j, m, s) when job i precedes job j on machine m of stage s
$\bar{M}$:	a very large positive number

The following notation is used for the variables in the MILP model for train routing and scheduling problem.

y_{ims}: $\begin{cases} 1, & \text{if job } i \text{ is assigned to machine } m \text{ of stage } s \\ 0, & \text{otherwise} \end{cases}$

x_{ijms}: $\begin{cases} 1, & \text{if job } i \text{ precedes job } j \text{ on machine } m \text{ of stage } s \\ 0, & \text{otherwise} \end{cases}$

t_{ims}: starting time of job i on machine m of stage s if job i is assigned to machine m of stage s

T_i: tardiness of job i

In this problem, jobs i needs to wait for a release time r_i to start its process on some machine m of some stage s. It is analogous to the time that a train enters the network to start its service. Similarly, the due date d_i of train i is the time that we expect the train to leave the network. The due date helps us to determine the tardiness of a job $T_j = \max(L_j, 0)$ which is equivalent to the train delay.

Tardiness calculation helps to determine the objective function of the train routing and scheduling problem. The aim is to minimize delay propagation which can be translated as the total weighted tardiness $\sum w_j T_j$. Weights are defined with regard to the train priorities and they can be set equal to 1 to deal with similar trains.

The train routing problem (MPM-JSS) has several similarities with the train scheduling problem (MB-JSS) in Chapter 5. However, there are several differences which are mainly resulted by considering single and parallel machines in this chapter instead of only single machines. In what follows, it becomes more clear how this special characteristic can thoroughly affect the problem formulation and solution method.

It is obvious how this formulation on one hand can aggregate the information to have less number of variables for the same partial network addressed in Chapter 5 which can offer an advantage in terms of the problem size and computational effort. On the other hand, it adds to the problem complexity as there is another degree of freedom which is about selecting among alternative routes. Finally, the main characteristic of the problem is to combine two decisions of scheduling and routing together which makes the problem more challenging and interesting for the real-world applications.

The MILP model for formulating the train routing and scheduling problem as a MPM-JSS problem is presented as below.

$$\text{Minimize} \qquad z = \sum_{i \in \mathcal{I}} w_i T_i \tag{6.1}$$

subject to

$$t_{i,m,s_{i,l_i}} + p_{i,m,s_{i,l_i}} - d_i - \bar{M}(1 - y_{i,m,s_{i,l_i}}) \leq T_i \qquad (i, m, s_{i,l_i}) \in \mathcal{O} \tag{6.2}$$

$$(t_{i,m,s_{i,1}} - r_i y_{i,m,s_{i,1}}) \geq 0 \qquad (i, m, s_{i,1}) \in \mathcal{O} \tag{6.3}$$

$$(t_{i,m,s_{i,k-1}} + p_{i,m,s_{i,k-1}}) - \bar{M}(1 - y_{i,m,s_{i,k-1}}) \leq t_{i,m',s_{i,k}} + \bar{M}(1 - y_{i,m',s_{i,k}})$$
$$(i, m, s_{i,k-1}), (i, m', s_{i,k}) \in \mathcal{O},\ k = 2, \ldots, l_i \tag{6.4}$$

$$t_{ims} + \max\{p_{ims}, h_{ijms}\} + \bar{M}(x_{ijms} - 1) \leq t_{jms} \quad (i, m, s),\ (j, m, s) \in \mathcal{O} \tag{6.5}$$

$$t_{jms} + \max\{p_{jms}, h_{jims}\} + \bar{M}(x_{jims} - 1) \leq t_{ims} \quad (i, m, s),\ (j, m, s) \in \mathcal{O} \tag{6.6}$$

$$x_{ijms} \leq y_{ims} \quad (i, m, s),\ (j, m, s) \in \mathcal{O} \tag{6.7}$$

$$x_{jims} \leq y_{jms} \quad (i, m, s),\ (j, m, s) \in \mathcal{O} \tag{6.8}$$

$$x_{ijms} + x_{jims} \leq 1 \qquad (i, m, s),\ (j, m, s) \in \mathcal{O} \tag{6.9}$$

$$y_{ims} + y_{jms} - x_{ijms} - x_{jims} \leq 1 \qquad (i, m, s),\ (j, m, s) \in \mathcal{O} \tag{6.10}$$

$$\sum_{m \in \mathcal{M}_s} y_{ims} = 1 \quad i \in \mathcal{I},\ s \in (s_{i1}, \ldots, s_{i,l_i}) \tag{6.11}$$

$$x_{ijms}, y_{ims} \in \{0, 1\}\ (i, m, s),\ (j, m, s) \in \mathcal{O} \tag{6.12}$$

$$T_i \geq 0 \qquad i \in \mathcal{I} \tag{6.13}$$

The objective function to avoid delay propagation is defined with the total weighted tardiness in (6.1). Constraints (6.2) determines the tardiness of a job which defines a train's delay. To do so, jobs starting time on the last machine of its sequence, its processing time on that machine and the due date of the job is considered if the job is processed on machine m of stage s_{i,l_i}.

Constraints (6.3) are to make sure that the starting time of a job on the first machine of its sequence is no earlier than its release time if it is processed on machine m of stage $s_{i,1}$. It means that a train is restricted to start its service after it has entered the railway network.

The starting time of succeeding operation of job i of stage s_k should be no earlier than its finish time on the proceeding operation on stage s_{k-1} considering the machine on which it is processed. This is achieved through constraints (6.4) which are called the set of *conjunctive* constraints. They enforce the running and dwell time constraints for trains.

The pair of constraints (6.5) and (6.6) restrict the order of jobs i and j if both jobs are processed on machine m of stage s. They are a modification of *disjunctive* constraints. These constraints are adapted to represent that there should be a minimum headway h_{ijms} (h_{jims}) between two consecutive trains i and j running on the same track m of track section s if train i precedes train j (train j precedes train i).

Constraints (6.7) and (6.8) and the binary constraints (6.9) and (6.10) force the condition that either job i precedes job j or vice versa only if they are both assigned to the same machine m of stage s. More specifically, if both job i and j are allocated to the same machine of a stage ($y_{ism} = y_{jsm} = 1$), then considering (6.9) and (6.10) together results in constraint $x_{ijms} + x_{jims} = 1$. This is equivalent to deciding the precedence order of two trains traversing on the same track of a track section.

Constraint (6.11) determines that each job i has to be allocated to one machine of stage s. This is similar to the safety rule of only one train running on a track of a track section at a time. Finally, constraint 6.12 ensures the nonnegativity of the tardiness of jobs. Now that the mathematical programming model of the combined train routing and scheduling problem is shown, we move on to next Subsection 6.2.2 for the graph representation of the problem.

6.2.2 Extended disjunctive graph formulation

The extended disjunctive programming model introduced in the previous section is illustrated with a directed graph for analysing the train scheduling and routing problem. Most of the research studies in job shop scheduling are based on the disjunctive graph formulation (See Chapter 5 for more discussions on disjunctive graph).

The proposed parallel machine job shop scheduling is also modelled with an extended disjunctive graph in this section to be implemented in an efficient solution algorithm. We adapt the disjunctive graph by Liu and Kozan (2012) which models a parallel machine job shop scheduling. With some modifications we modify the graph slightly in order to solve a parallel machine job shop scheduling with release times. Moreover, the graph is transformed to represent a job shop problem for minimizing total weighted tardiness. It should be noted we set out an activity-on-edge while Liu and Kozan (2012) has an activity-on-node graph. In the following, the other changes are mentioned in detail where appropriate, while we elaborate on graph specifications.

The extended graph $G = (N, A, B)$ is described with a set of nodes shown by N, set of conjunctive arcs A and a set of disjunctive arcs B. There are two types of nodes in set N, namely *actual* and *virtual* nodes. Assume that we have a node for the operation of a job $i, i \in \mathcal{I}$ on either a single machine stage s or on each machine m of parallel machine stage s.

As not all jobs are processed on all stages and only one machine of a parallel machine stage is assigned to a job at a time, there are some nodes for a job i with no process. That is why we have some actual nodes with a corresponding process on a stage with a single or parallel machines and some nodes are virtual with no associated process.

A dummy node U is needed as a source node of the graph and there are dummy sink nodes $V_i, i \in \mathcal{I}$ for each job. The sink nodes for each job help us to calculate the tardiness of each job and the total weighted tardiness consecutively. Whereas in Liu and Kozan (2012) the objective function is to minimize makespan and a single sink node would suffice for makespan calculations.

Set A of *conjunctive arcs* includes fixed arcs that represent the precedence relation between two consecutive operations corresponding to the same job. Conjunctive arcs also connect the source node to the first operation of each job and the last operation of each job is connected to the sink of the corresponding job.

Length of the conjunctive arcs are equal to the processing time of the operations from which they branch. However, the length of the conjunctive arcs which stem from the source node to the first operation of each job is equal to the release time r_i of job i. Here, we can point out the second difference to Liu and Kozan (2012) where release time is not an issue of interest.

B is the set of *modified disjunctive arcs* which are not fixed. They represent the undetermined precedence between two jobs which are operated on the same machine. The precedence relation is presented as a pair of arcs in opposite directions between two actual operations on the same machine. The length of the modified disjunctive arcs are equal to $\max\{p_{ims}, h_{ijms}\}$ or in other words it reflects the bigger value between the processing time (train running time) and the time separation between two consecutive jobs on the same machine (headway between two consecutive trains).

Assuming that we have S number of stages, the set of disjunctive arcs can be divided into S subsets shown by $B_1, B_2, \dots, B_S$, where B_k, $(k = 1, 2, \dots, S)$ denotes the disjunctive arcs among actual operations that are processed on single or parallel machines of a stage s.

The characteristics of the disjunctive arcs on the single machine stages are very much similar to the disjunctive graph shown in Chapter 5. For each actual operation on a single machine stage, there is a pair of disjunctive arcs with the other operation on the same stage.

However, modified disjunctive arcs on the multiple machine stages are more complicated. Every actual operation of a job can only be paired with each actual operation of the other jobs which are processed on the same machine. The following figure helps us to illustrate the mentioned concepts.

Figure 6.3 is an example of a parallel machine job shop with 6 jobs and 7 stages. An operation of job i on machine m of stage s is represented as a node denoted by (i, m, s). A rectangle embracing some nodes in the graph shows that the corresponding operations are processed on the same-type machine.

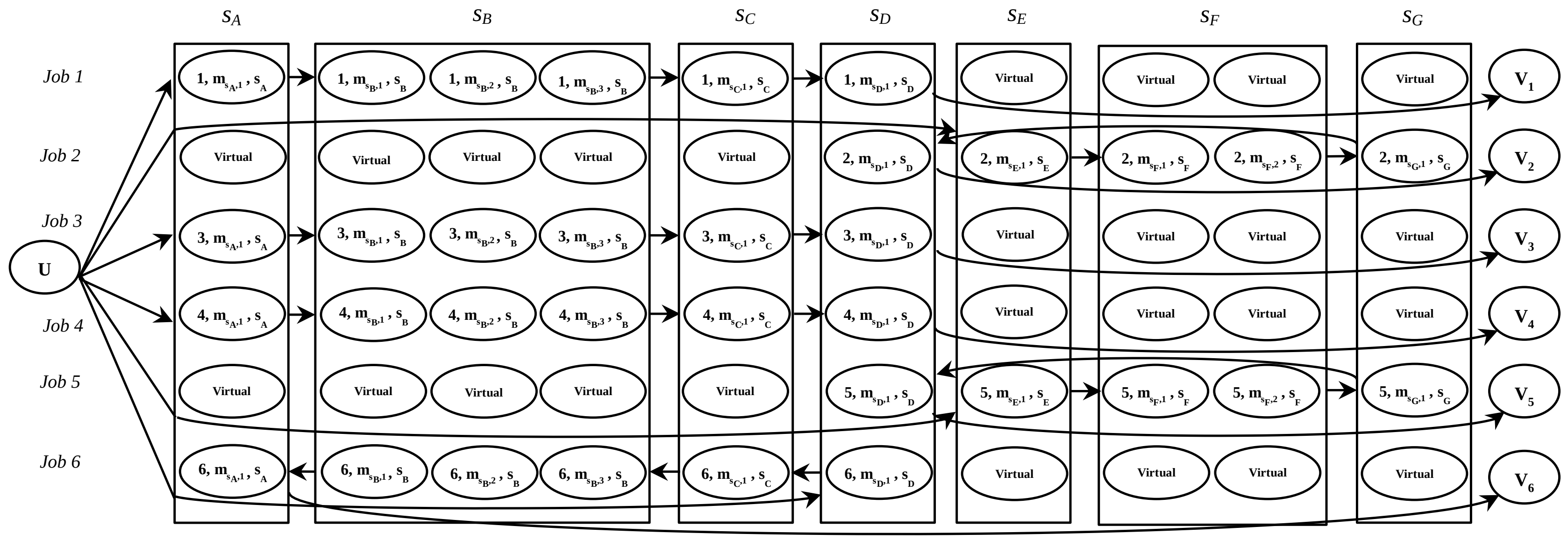

FIGURE 6.3: MPM-JSS graph: modified parallel machine job shop scheduling problem for $Pm|r_j|\sum w_jT_j$

In addition, conjunctive arcs are drawn in this graph. For instance, we have the following set of conjunctive arcs for job $i = 1$

$$(1, m_{s_A,1}, s_A) \rightarrow (1, m_{s_B,1}, s_B) \rightarrow (1, m_{s_C,1}, s_C) \rightarrow (1, m_{s_D,1}, s_D) \tag{6.14}$$

where (i, m, s) denotes the set of operations, for $i = 1$, $s = s_A, \dots, s_G$ and $m = m_{s1}, \dots, m_{s,l_s}$.

For more clarity, disjunctive arcs are not illustrated in the graph. Instead, they are listed separately in the following for a single machine stage and a double machine stage. For each node in the single machine stage A with 4 jobs, we have

$$\begin{aligned}(1, m_{s_A,1}, s_A) : {} & (1, m_{s_A,1}, s_A) \rightarrow (3, m_{s_A,1}, s_A);\ (1, m_{s_A,1}, s_A) \rightarrow (4, m_{s_A,1}, s_A); \\ & (1, m_{s_A,1}, s_A) \rightarrow (6, m_{s_A,1}, s_A)\end{aligned} \tag{6.15}$$

$$\begin{aligned}(3, m_{s_A,1}, s_A) : {} & (3, m_{s_A,1}, s_A) \rightarrow (1, m_{s_A,1}, s_A);\ (3, m_{s_A,1}, s_A) \rightarrow (4, m_{s_A,1}, s_A); \\ & (3, m_{s_A,1}, s_A) \rightarrow (6, m_{s_A,1}, s_A)\end{aligned} \tag{6.16}$$

$$\begin{aligned}(4, m_{s_A,1}, s_A) : {} & (4, m_{s_A,1}, s_A) \rightarrow (1, m_{s_A,1}, s_A);\ (4, m_{s_A,1}, s_A) \rightarrow (3, m_{s_A,1}, s_A); \\ & (4, m_{s_A,1}, s_A) \rightarrow (6, m_{s_A,1}, s_A)\end{aligned} \tag{6.17}$$

$$\begin{aligned}(6, m_{s_A,1}, s_A) : {} & (6, m_{s_A,1}, s_A) \rightarrow (1, m_{s_A,1}, s_A);\ (6, m_{s_A,1}, s_A) \rightarrow (3, m_{s_A,1}, s_A); \\ & (6, m_{s_A,1}, s_A) \rightarrow (4, m_{s_A,1}, s_A)\end{aligned} \tag{6.18}$$

Choosing disjunctive arcs on a single machines is similar to Chapter 5. For a pair of nodes (i, m, s) and (j, m, s) on a single machine stage, there are only two options; either processing (i, m, s) before (j, m, s) or vice versa. When a schedule is determined, only one of disjunctive arcs are selected.

The disjunctive arcs for the double machine stage F with 2 jobs are summarized for each node as below

$$(2, m_{s_F,1}, s_F) : (2, m_{s_F,1}, s_F) \rightarrow (5, m_{s_F,1}, s_F);\ (2, m_{s_F,1}, s_F) \rightarrow (5, m_{s_F,2}, s_F) \tag{6.19}$$

$$(2, m_{s_F,2}, s_F) : (2, m_{s_F,2}, s_F) \rightarrow (5, m_{s_F,1}, s_F);\ (2, m_{s_F,2}, s_F) \rightarrow (5, m_{s_F,2}, s_F) \tag{6.20}$$

$$(5, m_{s_F,1}, s_F) : (5, m_{s_F,1}, s_F) \rightarrow (2, m_{s_F,1}, s_F);\ (5, m_{s_F,1}, s_F) \rightarrow (2, m_{s_F,2}, s_F) \tag{6.21}$$

$$(5, m_{s_F,2}, s_F) : (5, m_{s_F,2}, s_F) \rightarrow (2, m_{s_F,1}, s_F);\ (5, m_{s_F,2}, s_F) \rightarrow (2, m_{s_F,2}, s_F) \tag{6.22}$$

Choosing among disjunctive arcs in a parallel machines needs more attention. Because there are obviously more possibilities for the order of the jobs which are processed on the same machine due to identical parallel machines. In this case, not all of the nodes that share the same stage can have a selected disjunctive arc. Disjunctive arcs are selected only for the jobs which are scheduled on the same machine within a stage. For example, initially we have all the above arcs among nodes on machine F. Later, when a schedule is found, only one arc in the pair of $(2, m_{s_F,1}, s_F) \rightarrow (5, m_{s_F,1}, s_F)$ or $(5, m_{s_F,1}, s_F) \rightarrow (2, m_{s_F,1}, s_F)$; and only one arc in the pair of $(2, m_{s_F,2}, s_F) \rightarrow (5, m_{s_F,2}, s_F)$ or $(5, m_{s_F,2}, s_F) \rightarrow (2, m_{s_F,2}, s_F)$ can be selected. However, each of job 2 and 5 have two options to do the same process on machine 1 or 2 of stage F.

We have discussed how the relevant disjunctive arcs are selected in single and parallel machines stages. A selection of one direction of the relevant disjunctive arcs determines the order of the jobs to be processed on the same machine for both single and parallel machine stages. At the same time, it defines the chosen machine in a stage to carry out the process on a parallel machine stage. A feasible schedule for the MPM-JSS problem corresponds to a selection of one direction for the relevant pair of disjunctive arcs such that there is no cycle in the graph. Conversely, an acyclic selection of the disjunctive arcs by choosing exactly one arc from the relevant pair of disjunctive arcs results in a feasible solution for the MPM-JSS problem.

The longest path from source U to one of the sinks $V_i, i \in \mathcal{I}$ helps to determine the tardiness of a job and calculate the total tardiness consecutively. We elaborate more on this calculation in Section 6.3.

In this section, we have extended the classical disjunctive graph to formulate the train routing and scheduling problem. The extended graph is used in order to analyse the problem and develop a novel heuristic approach in Section 6.3.

6.3 Solution methods

This section addresses solution methods for the routing and scheduling problem formulated in Section 6.2 as a MILP model and the so-called MPM-JSS graph. A novel approach is suggested based on the well-known SB heuristic which is introduced in Chapter 5. This approach is configured around the single and multiple machine formulation discussed earlier

The SB heuristic aims to propose train routes among alternative options so that the delay propagation in the network is minimized. Simultaneously, train timing and ordering on the corresponding routes are determined. This problem can obviously be solved in both levels of tactical and operational level with concerns on feasibility of the timetable with the infrastructure and rerouting in real-time respectively.

In Subsection 6.3.1 an algorithm for train routing and scheduling based on FCFS dispatching rule is presented which is close to real application. A modified SB algorithm is provided in Subsection 6.3.2 which improves classical features of the SB algorithm with substantial novelties. We report on the computational results for London bridge area case study firstly introduced in Chapter 3. It shows efficiency and viability of the SB algorithm to be employed in real-life train routing problem.

6.3.1 FCFS dispatching rule

FCFS is developed based on a well-known FCFS dispatching rule which sequences trains in non-decreasing order of their release times on each track section. In this case, the FCFS algorithm for train routing and scheduling problem is implemented with respect to the operational and safety rules which are described in the following.

The train traversal on a track section takes as long as its running time. A train can only enter a track of a track section if the track section is free. Enough separation between two consecutive trains which consecutively use the same track is ensured by taking into account the maximum value between running time of the preceding train on the track section and the headway value. It should be highlighted that trains can consecutively use the same track which is the case when one train enters the same track after the other one has left that track. Therefore, there is no more than one train per track section at a time.

This algorithm simulates real-life dispatching decisions and therefore, it can reasonably serve as a baseline for the comparisons in computational experiments. In the following a sophisticated SB algorithm is suggested for the train routing and scheduling problem.

6.3.2 Modified Shifting Bottleneck procedure

The idea of using Shifting Bottleneck procedure to solve the classical job shop scheduling problem has been extensively explored in Chapter 5. Based on the analysis that has been performed in the in Section 6.2, we need to develop a variation of SB algorithm to deal with both single and parallel machines to solve the MPM-JSS problem. It is due to the fact that the introduced train routing and scheduling problem consists of single and parallel tracks modelled as stages.

In order to calculate the total weighted tardiness, we introduce a novel approach inspired by Liu and Kozan (2012). As it has been mentioned in the problem formulation in Section 6.2, our problem differs from theirs in two main aspects which are the objective function and the job properties.

Although both studies consider the parallel machine job shop scheduling problem, they can be considered as two completely different variation of the problem. We consider the objective function of minimizing total weighted tardiness whereas Liu and Kozan (2012) minimize the makespan. Moreover, the release time and tardiness of jobs are incorporated in our model and solution method while Liu and Kozan (2012) assume that all jobs are available at time zero and tardiness is not a concern in their study.

The SB algorithm proposed for the train routing and scheduling problem is similar in framework to the approach in Chapter 5. Here, we need to mention about the novelty of the suggested algorithm compared to Pinedo and Singer (1999) and D'Ariano (2008) studies which are the main inspiration for our SB algorithm suggested in Chapter 5. The same arguments stands for novelty of our SB algorithm for the train routing and scheduling problem as discussed in Subsection 5.3.5.2. In the following, we mention the main distinctive features of our proposed our SB algorithm.

A special characteristic of our SB algorithm in comparison with Pinedo and Singer (1999) algorithm is the presence of both single and parallel machine subproblems. Our novel algorithm extends the ATC index to be viable for the parallel machine subproblems.

Our novel approach addresses scheduling and routing simultaneously in each iteration of the SB algorithm; whereas, D'Ariano (2008) solve the scheduling and routing problems iteratively in separate steps that is to compute an optimal train sequence for given train routes and then improve it by locally rerouting some trains.

Other novelties and modifications suggested in the introduced SB algorithm is discussed in more detail as we explain each step of the algorithm.

In general, SB is known as a decomposition approach for $J||C_{\max}$ problem. In each iteration, SB decomposes the problem into multiple instances of single-machine $1|r_j|L_{\max}$ which are called subproblems. Then, the so-called subproblems are solved by a subproblem solution procedure which is a branch-and-bound technique by Carlier (1982) in the original version of SB.

Then, they are evaluated in terms of the machine criticality. Finally, the most critical machine is chosen and sequenced and the whole process is repeated until all machines are scheduled. For a comprehensive discussion of the SB procedure and the original approach we refer the reader to Subsection 5.3.5.

In this section, the major steps of the SB are maintained and we adapt and improve the SB algorithm to solve the MPM-JSS problem efficiently. Assuming a given MPM-JSS graph, it consists of several *single machine* (SM) and *parallel machine* (PM) subproblems. Hence, in this case, the SB approach needs to decompose and solve an SM or PM problem in each iteration.

In the following, we summarize the novel improvements in the suggested SB. A *topological-sequence algorithm* of Liu and Ong (2002) is modified with regard to the decomposition step. Liu and Ong (2002) employ the topological-sequence algorithm originally for critical path calculations in flow shop problems. The outcome of applying the modified topological-sequence algorithm to the MPM-JSS problem is a set of PM and SM problems with different heads r_i's and tails q_i's.

If the subproblem is an SM, a modified ATC index by Pinedo and Singer (1999) is implemented. This index solves the $1|r_j|\sum w_jT_j$ and we call it ATC-SM index. For a PM subproblem, we adapt an ATC index to include release times of jobs inspired by Lee and Pinedo (1997) who apply ATC index on parallel machines with setup times. The adapted variant of the ATC index in our study is customised for the $Pm|r_j|\sum w_jT_j$ and it is named as ATC-PM index.

We need to highlight that the presence of parallel machines in the train scheduling problem is the main feature of the train routing problem. The proposed SB algorithm decomposes the problem into single and parallel machine subproblems. This distinctive feature is addressed by an extending ATC index, which is originally developed for single machine problem, to be viable for the parallel machine problems. The parallel machine characteristic requires certain changes in the classical SB algorithm which is discussed in the following.

Given mentioned improvements, following framework is offered for an EXTended SB (EXT-SB) algorithm.

EXT-SB algorithm

Step 1: Set initial conditions $\mathcal{S}_0 = \emptyset$. Graph G should include all conjunctive arcs and no disjunctive arcs.

Step 2: Do the following to identify and solve each unscheduled stage $s \in \mathcal{S} \setminus \mathcal{S}_0$.
Step 2-1: Decompose the problem into SM and PM problems using the topological-sequence algorithm.
Step 2-2: Solve the SM subproblems by ATC-SM index for $1|r_j|\sum w_j T_j$ and solve the PM problems by ATC-PM index for $Pm|r_j|\sum w_j T_j$.

Compute

$$\bar{T}_s = \sum_{k=1}^{n} w_k \Big(\max_{(j,m,s)\in N_s} T^k_{jms} \Big), \quad T^k_{jms} = \max\{C_{jms} - d^k_{jms}, 0\} \tag{6.23}$$

where T^k_{jms} is the tardiness of job j processed on machine m of stage s calculated with respect to each sink V_k for job k, and N_s is the set of nodes corresponding to the operations processed on stage s.

Step 3: Determine the bottleneck stage s', where

$$\bar{T}_{\max}(s') = \max_{s \in \mathcal{S}\setminus\mathcal{S}_0} (\bar{T}(s)) \tag{6.24}$$

Sequence the jobs on stage s' according to the sequence obtained in the previous step. Set $\mathcal{S}_0 = \mathcal{S}_0 \cup \{s'\}$.

Step 4: Stop if $\mathcal{S}_0 = \mathcal{S}$, otherwise go to Step 2.

A re-optimization step is usually carried out in the classical version of the SB algorithm. We also implement it in the EXT-SB algorithm version with re-optimization after Step 3 which is explained in the following

Re-optimize each stage $l \in \mathcal{S}_0 \setminus \{s'\}$ by solving its subproblem taking into account the sequences on stages $\mathcal{S}_0 \setminus \{l\}$.

Based on our observations earlier in Chapter 5, SB algorithm based on the ATC index with and without re-optimization, namely SB-ATC and SB-ATC-ReOpt, show better performance compared to the other variants of the SB algorithm. Hence, two variants of the EXT-SB algorithm are developed; The SB algorithm without re-optimization and the SB algorithm with re-optimization are called RSB-ATC and RSB-ATC-ReOpt, respectively. It should be noted that R in RSB-ATC and RSB-ATC-ReOpt corresponds to the Routing algorithm. Given that machine s' is sequenced, in order to re-optimize stage $l \in \mathcal{S}_0 \setminus \{s'\}$, all corresponding disjunctive arcs of stage l are deleted from the MPM-JSS graph. We assume that the order of re-optimizing stages $l \in \mathcal{S}_0 \setminus \{s'\}$ is given by $l(1), \ldots, 1(p)$, $p = |\mathcal{S}_0| - 1$. Consequently, $l(i)$, $i = 1, \ldots, p$ is ordered according to the decreasing solution value of the sequencing problem. It is observed that sometimes no tardy stage is selected based on the bottleneck selection. Hence, stage $s' \in \mathcal{S}$ is selected such that for all j and k, $d'_{j'm'} = \min d^k_{jm}$. In this case, no re-optimization is performed after selecting such a machine.

More details for each step of the EXT-SB are given in the rest of this chapter. We begin with a discussion on topological-sequence algorithm in Subsection 6.3.2.1. Then, single and parallel machine subproblems are solved in Subsection 6.3.2.2.

6.3.2.1 Adapted topological-sequence algorithm

Adapted topological-sequence algorithm is implemented in each step of the EXT-SB to analyse a partial MPM-JSS graph consisting of all conjunctive arcs for operations of the same job and the disjunctive arcs belonging to the stages which are already sequenced $s \in \mathcal{S}_0$. This algorithm aims at decomposing the multiple machine stages in MPM-JSS problem into SM and PM subproblems so that they can be solved in Step 3 of EXT-SB algorithm in order to determine the bottleneck machine in each iteration.

The topological-sequence algorithm was initially introduced by Liu and Ong (2002) to calculate head r_i (the length of the longest path from the source to the node associated with job i) and tail q_i (the length of the longest path from the node associated to job i to the sink) of each node based on critical path calculations in a disjunctive graph. The algorithm is used to solve two variants of the flow shop problems. Later, Liu and Kozan (2012) implement the same algorithm in a study for parallel-machine job shop scheduling problem. The first steps of the algorithm are similar to the well-known topological sort algorithm or topological ordering algorithm which is mentioned firstly in Subsection 5.3.5.2.

In the extended disjunctive graph for MPM-JSS problem, we have a sink $V_i, i \in \mathcal{I}$ for each job. Thus, it is necessary to change the original topological-sequence procedure for MPM-JSS graph with several dummy sinks for each job. In the following, the modified topological-sequence algorithm is described.

Modified topological-sequence algorithm

Step 1: Count the in-degree (the predecessors) of the nodes.

Step 2: Find a topological order of the nodes as below.

Step 2.1: Select and put the source node in the list as it is the only node that it has the in-degree of zero at the beginning.

Step 2.2: Decrease the in-degree of the immediate successors of the selected node by 1.

Step 2.3: Choose a node with an in-degree of zero from unselected nodes and add it to the list.

Step 2.4: Repeat step 2.2 and 2.3 until all nodes are ordered.

Step 3: Set the head of the source node equal to zero. Compute the head of the rest of the nodes (other than the first operation of a job) in the topological order according to the following equation

$$r_i = \max\{r_{PM_i} + p_{PM_i}, r_{PJ_i} + p_{PJ_i}\} \tag{6.25}$$

where r_i is the head of the operation i (A single index i is used to show the operation in this algorithm), PM_i is the operation on the same stage processed just before operation i if it exists and PJ_i is the operation of the same job processed just before operation i if it exists. For the first operation of a job, r_i is equal to the release time of the job associated with operation i.

Step 4: Do the following for each sink $V_k, k \in \mathcal{I}$. Set the tail of the sink V_k equal to zero. Compute the tail of the rest of the nodes (other than the last operation of a job) with respect to sink k denoted as $L(i, V_k)$ in the reverse order of the topological order according to the following equation

$$L(i, V_k) = \max\{L(SM_i, V_k) + p_{SM_i}, L(SJ_i, V_k) + p_{SJ_i}\} \tag{6.26}$$

where SM_i is the operation on the same stage processed immediately after operation i if it exists and SJ_i is the operation of the same job processed immediately after operation i if it exists. For the last operation of a job, $L(i, V_k)$ is equal to the processing time of the job associated with operation i.

In Subsection 6.3.2.2, we show how SM and PM problems arising from the Modified topological-sequence algorithm are solved.

6.3.2.2 SM and PM subproblems

When the modified topological-sequence algorithm is implemented in an iteration of the EXT-SB algorithm, the SM and PM subproblems are generated. In Step 2.2 of the EXT-SB algorithm, we need to formulate and solve these subproblems. The SM and PM subproblems with different heads and tails have the objective functions of minimizing total weighted tardiness $\sum w_j T_j$. In what follows, we discuss the formulation and solution methods for the SM and PM subproblems.

SM subproblem

It can be clearly seen that arising SM problem is minimizing total weighted tardiness on a single machine with release times $1|r_j|\sum w_j T_j$.

In the SM problem arising from the modified topological-sequence algorithm, a set of jobs $j \in \mathcal{I}$ has to be processed on a stage with a single machine m. Each job has a fixed processing time p_{jm}, a head r_{jm} and a tail $L((j, m), V_k)$ with respect to job k.

Now that an instance of $1|r_j|\sum w_jT_j$ is generated, we can also define the local due date of operation (j,m) according to equation (5.25).

Thus, the local tardiness of the operation (j,m) with respect to job k can be computed by equation (5.28).

Assuming that $\mathcal{S}_0$ is the set of currently sequenced stages, there is a SM subproblem corresponding to stage $s \in \mathcal{S}\setminus \mathcal{S}_0$ which is resulted from the decomposition by the modified topological-sequence algorithm. We assume a notation similar to train scheduling and rescheduling problem in Section 5.2 for $t_{i,m_{i,l_i}}$, $p_{i,m_{i,l_i}}$, t_{im}, p_{im}, h_{ijm}, x_{ijm}. Thus, the mathematical programming of the decomposed SM problem for stage $s \in \mathcal{S}\setminus \mathcal{S}_0$ is given in the following.

$$\text{Minimize} \qquad z = \sum_{k\in\mathcal{I}} w_k(\max_{i,m} T^k_{im}) \tag{6.27}$$

subject to

$$T^k_{im} \geq t_{i,m_{i,l_i}} + p_{i,m_{i,l_i}} - d_i \qquad (i,m) \in \mathcal{O}, k \in \mathcal{I} \tag{6.28}$$

$$t_{i,m_{i,1}} \geq r_i \qquad i \in \mathcal{I} \tag{6.29}$$

$$t_{jm} - t_{im} + \bar{M}(1 - x_{ijm}) \geq \max\{p_{im}, h_{ijm}\} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{6.30}$$

$$t_{im} - t_{jm} + \bar{M}(1 - x_{jim}) \geq \max\{p_{jm}, h_{jim}\} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{6.31}$$

$$x_{ijm} + x_{jim} = 1 \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{6.32}$$

$$x_{ijm} \in \{0,1\} \qquad (i,m),\ (j,m) \in \mathcal{O} \tag{6.33}$$

$$T^k_{im} \geq 0 \qquad (i,m) \in \mathcal{O}, k \in \mathcal{I} \tag{6.34}$$

After modeling the problem, we need a mechanism to solve the SM subproblem. We implement an enumeration heuristic based on the modified ATC-SM index. Pinedo and Singer (1999) adapt a priority rule called ATC index to solve $1|r_j|\sum w_jT_j$. The ATC index is initially developed by Vepsalainen and Morton (1987) and it is well-known for solving $1||\sum w_jT_j$. We have mentioned earlier in Subsection 5.3.5 that we use the ATC index proposed by Pinedo and Singer (1999) to minimize total weighted tardiness in a single machine problem with release times. We employ the same ranking index which is called ATC-SM index in this chapter and has been defined in equation (5.27) (See Subsection 5.3.5 for more details about the ATC index for the single machine problems).

After calculating the ATC-SM index for all the operations, we select the operation (j, m) with the highest index and put it in the first position on machine m. In order to define which operation should be scheduled next, indices are recalculated for the remaining operations and the job with the highest value is selected. This process is repeated until all operations are scheduled.

PM subproblem

Similarly, we can define the PM problem which is generated after the decomposition step in SB-EXT. In each iteration of SB-EXT, only one stage is sequenced which consist of some parallel machines. So instead of a single machines problem, we need to formulate and solve a parallel machine problem.

In the PM problem arising from the modified topological-sequence algorithm, there is a set of jobs $j \in \mathcal{I}$ which need to be processed on machine m in stage s. Each job $j \in \mathcal{I}$ has a processing time p_{js}, a head r_{jms} and a tail $L((j, m, s), V_k))$ which is defined with respect to job k.

In order to solve the parallel machine problem we need to firstly define which jobs have to be assigned to which units of the parallel machine and secondly to determine the sequence of the jobs assigned to each unit of the parallel machine. The local due date of each operation (j, m, s) is defined in the following.

$$d^k_{jms} = \begin{cases} \max\{C_k, d_k\} - L((j, m, s), V_k) + p_{js} & \text{if } L((j, m, s), V_k) \text{ exists,} \\ \infty & \text{otherwise.} \end{cases} \tag{6.35}$$

Consequently, we can compute the local tardiness of the operation (j, m, s) with respect to job k by using the formula

$$T^k_{jms} = \max\{C_{jms} - d^k_{jms}, 0\}. \tag{6.36}$$

The mathematical programming formulation of the PM sub-problem is determined as below.

$$\text{Minimize} \qquad z = \sum_{k \in \mathcal{I}} w_k (\max_{i,m,s} T^k_{ims}) \tag{6.37}$$

subject to

$$T^k_{ims} \geq \sum_{s \in \mathcal{S}} (t_{i,m,s_{i,l_i}} + p_{imsi,l_i} - d_i - \bar{M}(1 - y_{i,m,s_{i,l_i}}))$$

$$(i,m,s) \in \mathcal{O}, k \in \mathcal{I} \tag{6.38}$$

$$t_{i,m,s_{i,1}} \geq r_i \qquad i \in \mathcal{I} \tag{6.39}$$

$$t_{jms} - t_{ims} + \bar{M}(1 - x_{ijms}) \geq \max\{p_{ims}, h_{ijms}\} \qquad (i,m,s),\ (j,m,s) \in \mathcal{O} \tag{6.40}$$

$$t_{ims} - t_{jms} + \bar{M}(1 - x_{jims}) \geq \max\{p_{jms}, h_{jims}\} \qquad (i,m,s),\ (j,m,s) \in \mathcal{O} \tag{6.41}$$

$$x_{ijms} + x_{jims} \leq 1 \qquad (i,m,s),\ (j,m,s) \in \mathcal{O} \tag{6.42}$$

$$y_{ims} + y_{jms} - x_{ijms} - x_{jims} \leq 1 \qquad (i,m,s),\ (j,m,s) \in \mathcal{O} \tag{6.43}$$

$$\sum_{m \in \mathcal{M}_s} y_{ims} = 1 \qquad (i,m,s) \in \mathcal{O} \tag{6.44}$$

$$x_{ijms}, y_{ims} \in \{0,1\} \qquad (i,m,s),\ (j,m,s) \in \mathcal{O} \tag{6.45}$$

$$T^k_{ims} \geq 0 \qquad (i,m,s) \in \mathcal{O},\ k \in \mathcal{I} \tag{6.46}$$

In order to determine the bottleneck machine in Step 3 of the SB-EXT algorithm, the PM subproblem should be solved. The ATC-PM index is introduced to develop an enumeration heuristic. ATC-PM index is an extension of ATC index by Pinedo and Singer (1999) which is extensively discussed in Subsection 5.3.5. Whereas ATC index is suitable for single machine problems, ATC-PM needs certain modifications to be applicable to parallel machine problems. ATC-PM index is adapted to solve $Pm|r_j|\sum w_j T_j$ as below.

$$I_{jms}(t_s) = \sum_{k=1}^{n} \frac{w_k}{p_{jms}} \exp\left(- \frac{\max(d^k_{jms} - p_{jms} + (r_{jms} - t_s), 0)}{K\bar{p}_{ms}} \right), \tag{6.47}$$

where t_s is the earliest time at one of the parallel machines of stage s can be used which is the earliest time that the last job on the machine is completed or if more than one machine is freed at a time, one can be chosen arbitrarily. Moreover, $\bar{p_{ms}}$ is the average processing time of jobs assigned to machine m of stage s that are calculated by averaging the processing time of the operations which are still to be sequenced on machine m of stage s. K is a scaling parameter whose value has been determined to be equal to 4 empirically with computational tests (See Subsection 5.3.5 for more details about the ATC index for the single machine problems).

We compute the ATC-PM index for all the operations and choose the operation (j, m, s) with the highest index. The operation with the highest index should be processed at time t_s. At each time t_s, the same process is repeated to determine the next operation to be scheduled until all operations are sequenced. It means indices for the un-sequenced operations are computed and the job with the largest ATC-PM is chosen to be processed on the next position.

Computational results are provided for the suggested SB algorithms RSB-ATC and RSB-ATC-ReOpt in Subsection 6.4.

6.4 Computational results

The computational result of this chapter is based on London Bridge area based on Kent which is mentioned earlier in Section 3.6. We investigate possible effect of different routing decisions in this area. There are many parallel tracks in the area which gives us the opportunity to propose new routes for trains. London Bridge area has a complicated infrastructure and it includes several junctions and stations. The parallel tracks in this area are indicated according to *Track Diagrams* of the Kent area which is a property of the Southeastern company and subject to a non-disclosure agreement. Track Diagrams show the detailed topology of the tracks with possible directions for train journeys. Hence, we can define the potential routes for the trains according to these diagram.

Given the above information for the routes, we simulate different traffic conditions on the network. We look into different scenarios for blocked routes and perturbed timetables. We investigate the impact of possible routing options on the quality of the suggested routes and schedules. The proposed EXT-SB algorithm tries to offer routes and schedules for trains so that the delay propagation is minimized in the network. All variants of EXT-SB algorithm have been coded in MS Visual C++ 2010 and run on a PC with a dual core, 3.00 GHz and 4GB RAM.

6.4.1 Test instances

We consider the 2010 timetable and timetable components which have been used in Chapter 5. Similarly, we address off-peak services for passenger trains which cycles every 30 minutes. For General characteristics of the test instances for the train routing problem such as the number of jobs and machines, the reader is referred to Section 3.4. We study the impact of different types of route blockages on single and multiple track segments and disturbances namely minor, general and major disruption. This classification of disruptions is based on SDP, a document for handling disruptions (See Subsection 5.4.1 for more details about SDP document and different types of disruptions) provided by the major train operator in the the Kent area. In summary, delays less than 15 minutes correspond to a minor disruption. When delay is between 15 and 30, we have a general disruption and delays more than 30 minutes are categorized as a major disruption.

We have clear measures for the length of delay for each category of disruption, but the number of disrupted trains are not clearly stated in SDP. We try to introduce reasonable ranges with regard to the definition of each disruption category as it is discussed earlier in Subsection 5.4.1. We generate minor disruption by delaying 1 to 5 trains, general disruption by perturbing 6 to 12 trains, and major disruption by disrupting 13 to 27 trains.

We generate 4 data sets randomly for various scenarios of disruption: blocked routes on a single track section, blocked routes on multiple track sections, longer running/dwell times on tracks sections, late departures of the trains from their origins. Generated instances in each data set differ from each other in terms of number of blocked tracks, set of disrupted tracks, set of delayed trains and amount of delay. Computational experiments compare the total delay of the schedules arising from FCFS, RSB-ATC and RSB-ATC-ReOpt algorithms.

First set of data in Table 6.1 represent blocked routes on a single track section where different number of tracks are blocked on a single track segment so the number of disrupted track segments in all instances is equal to 1. Number of blocked segments in a track segment is chosen from the set $\{1, 2, 3, 4\}$. This number clearly depends on how many parallel tracks exist on a particular track segment. Number of parallel tracks on a track segment is 2,3 or 5; That is, in the partial network of our case study, we have track segments which have 2, 3 or 5 parallel tracks.

When a track segment is chosen to have blocked tracks, one should be careful not to block all tracks of the track segment as the problem is not valid anymore. For example, when the number of parallel tracks is equal to 2, only 1 track can be blocked. The percentage of unavailable tracks is a relative measure of number of blocked tracks to the total number of tracks. Hence, by blocking 1 track on a track segment with 2 parallel tracks, %50 of the tracks are unavailable. In summary, instances are generated by differing the track segments and number of blocked tracks and last two columns only show the characteristics of blockages. Totally, 19 instances are generated in this set.

Table 6.1: Blocked tracks on a single track section

# of Disrupted Track Segment(s)	# of Blocked Track(s)	# of Parallel Tracks	% of Unavailable Tracks
1	1	2	50
1	1,2	3	33,67
1	1,2,3,4	5	20,40,60,80

Table 6.2 describes the second set of experiments for blocked routes on multiple track sections and the number of disrupted track segments is selected from $\{2, 3, 4, 5\}$. We combine different track segments with different number of parallel tracks, hence the number of blocked tracks can be a value in $\{2, 3, 4, 5, 6, 7, 8\}$. Last two columns of the table only show the characteristics of the blockages. For example, blockages are tried on track segments with different number of parallel tracks which is 2,3 or 5 and it create different percentages of unavailable tracks which is given in the last column of the table. In total, 16 instances are generated.

Instances generated for longer running/dwell times are shown in Table 6.3. Instances are generated across different types of disruptions; minor, general and major disruptions. Totally 6 instances are in each category of disruption by differing block delays in a range which is relevant to a particular disruption. For

Table 6.2: Blocked tarcks on multiple track sections

# of Disrupted Track Segment(s)	# of Blocked Track(s)	# of Parallel Tracks	% of Unavailable Tracks
2	2,3,4,5	2,3,5	20,40,60,80
3	3,4,5,6	2,3,5	20,40,60,80
4	4,5,6,7	2,3,5	20,40,60,80
5	5,6,7,8	2,3,5	20,40,60,80

example, block delay is chosen from $\{5, 10, 15\}$ for a certain number of trains selected from $\{3, 5\}$. Number of blocks in the last second column indicates how many blocks are delayed in each instance. For example, 1 or 2 blocks are delayed in instances for minor disruption.

Table 6.3: Longer running/dwell times

Disruption Type	# of Instances	Block Delay	# of Block(s)	# of Trains
Minor	6	5,10,15	1,2	3,5
General	6	20,25,30	2,3	9,12
Major	6	40,50,60	4,6	18,23

Table 6.4 defines instances for late departure for only minor and general disruption. In minor disruption category, there are 9 instances with departure delay chosen from $\{5, 10, 15\}$ on a particular number of trains selected from $\{1, 3, 5\}$. 6 instances are generated with departure delay in set $\{20, 25\}$ which affects a certain number trains in set $\{6, 9, 12\}$. As we have discussed earlier in Subsection 5.4.1, a train which is affected by a delay of more than 25 minutes is shifted outside the cycle and it is not an interesting instance as it has little conflict with the other trains.

Table 6.4: Late departures

Disruption Type	# of Instances	Departure Delay	# of Trains
Minor	9	5,10,15	1,3,5
General	6	20,25	6,9,12

In Subsection 6.4.2, we compare the solution quality of RSB-ATC and RSB-ATC-ReOpt algorithms in comparison with FCFS.

6.4.2 Analysis and comparison of the methods

Tables 6.5, 6.6, 6.7 and 6.8 summarize the performance of the routes and schedules obtained from RSB-ATC and RSB-ATC-ReOpt algorithms for each data set described in Subsection 6.4.1. The first column in each table represents the solution methods. The performance of solution methods for all instances are shown in the second and third columns.

Not all instances have a feasible solution as either the FCFS algorithm or SB algorithms result in deadlock and therefore infeasible solutions. Such instances are taken out of the performance calculations. Tables 6.5, 6.6, 6.7 report on the results obtained for all instances excluding unsolved instances in the fourth and fifth columns. Table 6.8 does not include these columns as data set 6.4 has no infeasible result.

The results of both all instances and all instances excluding unsolved instances across all Tables 6.5, 6.6, 6.7 and 6.8 list the average delay in minutes. Average Relative Delay (RD) is also calculated with respect to FCFS across all instances for each solution method. Average RD is calculated for all instances and all instances excluding unsolved instances as below

$$RD = \frac{FCFS \ \ solution - algorithm \ \ solution}{FCFS \ \ solution} \times 100 \tag{6.48}$$

where the algorithm solution is the result of the RSB-ATC and RSB-ATC-ReOpt algorithms. Recall the definitions for average delay in formula 5.37 and RD in formula 5.38 in Subsection 5.4.2.

The last two columns in all Tables 6.5, 6.6, 6.7 and 6.8 refer to the average runtime of each solution method in seconds and the percentage of feasible solutions of each solution method respectively. It should be noted that the runtime of the RSB-ATC and RSB-ATC-ReOpt algorithms are very short (about 8 seconds). In terms of reliability of the algorithm to find feasible solutions, RSB-ATC has the best performance with 100% feasible solutions. FCFS has a poor outcome regarding feasibility with 100% feasible solutions only for late departure instance in Table 6.8. RSB-ATC-ReOpt algorithms performs close to RSB-ATC in terms of feasibility of the solution and feasibility drops to 95% only in Table 6.5.

All four tables show an improvement in the average total delay of RSB-ATC algorithm relative to FCFS in average RD calculations for all instances and all instances excluding unsolved instances. It conforms to what we expected, as FCFS algorithm does not have any mechanism to reduce the total delay. Given the results for RSB-ATC-ReOpt, there is usually no improvement in the suggested route and schedule. RSB-ATC-ReOpt has better results in comparison to FCFS only in instances for blocked routes on a single track section which is shown in Table 6.5.

The re-optimization process worked very well for scheduling in Chapter 5 and SB-ATC-ReOpt have had the best performance among the suggested SB algorithms. However, it seems that re-optimization cannot provide good solutions for routing and scheduling in this chapter as the sequence suggested by RSB-ATC has already a high quality and re-sequencing generally has a diminishing effect.

RSB-ATC algorithm has the minimum average delay almost in all cases for all instances and all instances excluding unsolved instances across all tables. However, there are two exceptions in Tables 6.6 and 6.7 where average delay for all instances are minimum for FCFS algorithm. It is due to the fact that FCFS can only find feasible solutions in 76% and 74% of the cases.

Tables B.1 and B.2 in Appendix B represent the complete results for blocked routes on a single track section. They clearly show which track sections are more important when they are blocked. Individual results also show the level of blockage for each track segment that the delay becomes more significant. The critical track segments can be listed as 8, 9 and 10 which are shown in Figure 3.4. The track segments 1 and 7 have 5 parallel tracks and they are critical in terms of amount of delay when 4 out of 5 tracks are blocked.

An interesting observation is about track segment 9 which is the track segment including London Bridge and it has 3 parallel tracks in total. Any blockage on this track segment, whether it is only one or two tracks, results in high delays relative to the other blocks. It is the only case that FCFS results in deadlock for blocked routes on a single track section. It is very well in line with the Route Plans 2012 by NetworkRail (2010) which states that services operate near capacity in London Bridge. Therefore any blocked route results in big total delays in the network. A summary of experiments of the individual instances experiments is given in Table 6.5.

Table 6.5: Blocked tracks on a single track section

Solution Method	All Instances		Excluding the Unsolved Instances		Avg Runtime (secs)	% of Feasible Solutions
	Avg Delay per Train	Avg RD wrt FCFS	Avg Delay per Train	Avg RD wrt FCFS		
FCFS	0.36	-	0.36	-	**0.06**	95
RSB-ATC	**0.25**	**66.97%**	**0.19**	**66.97%**	7.19	**100**
RSB-ATC-ReOpt	0.51	17.05%	0.41	17.05%	7.51	**100**

According to complete results in Tables B.3 and B.4 in Appendix B, the results of multiple blocked tracks conform to the result of single blocked tracks in terms of criticality of some track segments. In general, blockage of important track segments can cause bigger delays and sometimes deadlocks arising by FCFS. In other words, which track section is blocked has greater impact on delay compared to how many tracks are blocked. Table 6.6 shows a summary of blocked routes on multiple track sections results.

Table 6.6: Blocked tracks on multiple track sections

Solution Method	All Instances		Excluding the Unsolved Instances		Avg Runtime (secs)	% of Feasible Solutions
	Avg Delay per Train	Avg RD wrt FCFS	Avg Delay per Train	Avg RD wrt FCFS		
FCFS	**0.55**	-	0.55	-	**0.01**	76
RSB-ATC	0.58	**32.41%**	**0.44**	**32.41%**	7.16	**100**
RSB-ATC-ReOpt	1.12	-17.38%	0.73	-17.38%	8.19	**100**

Tables B.5 and B.6 in Appendix B represent complete results for longer running/dwell times which are created for 3 types of disruptions. It seems that FCFS performs better in smaller delays as sometime it gives even better results compared to RSB-ATC and RSB-ATC-ReOpt in minor disruption instances. When delay becomes bigger, FCFS runs into infeasibility and deadlock more often. It can be clearly seen that RD of RSB-ATC algorithm with respect to FCFS becomes bigger which indicates RSB-ATC has more computational advantages in bigger disruptions. RSB-ATC outperforms FCFS not only in terms of quality of the solution, but also in terms of number of feasible solutions. Table 6.7 summarizes the results of the instances with longer running/dwell times.

Table 6.7: Longer running/dwell times

Solution Method	All Instances		Excluding the Unsolved Instances		Avg Runtime (secs)	% of Feasible Solutions
	Avg Delay per Train	Avg RD wrt FCFS	Avg Delay per Train	Avg RD wrt FCFS		
FCFS	**27.29**	-	27.29	-	**0.01**	74
RSB-ATC	41.88	**3.39%**	**26.07**	**3.39%**	7.21	**100**
RSB-ATC-ReOpt	37.78	−28.09%	28.69	−28.09%	9.90	95

Tables B.7 and B.8 in Appendix B provide the complete results for late departure instances where we have minor and general disruptions. The results show RD values of RSB-ReOpt with respect to FCFS are bigger when train departure is less delayed which is compatible with what we expected. When a train departure is delayed more it shares less amount of the cycle time with the other train during its journey on the partial network. Thus, it is less probable for it to have conflicts with the other trains. Hence, a bigger departure can result in an easier instance. Consequently, no instance is generated for delayed departure of more than 25 minutes. As the train is moved out of the cycle. Table 6.8 is a summary of the results of the late departures.

Table 6.8: Late departures

Solution Method	All Instances		Avg Runtime (secs)	% of Feasible Solutions
	Avg Delay per Train	Avg RD wrt FCFS		
FCFS	4.23	-	**0.01**	**100**
RSB-ATC	**4.03**	**17.00%**	7.19	**100**
RSB-ATC-ReOpt	5.64	−9.45%	8.36	**100**

Another set of statistics are summarized for FCFS and RSB-ATC algorithms in Table 6.9 for late departure instances described in Table 6.4. The results of average delay and average relative delay with respect to total delay is separately calculated for late trains and on-time trains. The results arising from RSB-ATC clearly show that more delays occur for late train compared to on-time train. Whereas, FCFS spreads the delay more to on-time trains. The results are compatible with the intended objective of the algorithm as RSB-ATC is developed to minimize total delay in order to minimize delay propagation in the network. Complete results of this test is given in Table B.9 in Appendix B.

Table 6.9: Late and On-time Trains

	FCFS		RSB-ATC	
	Late Trains	On-time Trains	Late Trains	On-time Trains
Avg Delay per Train	3.40	0.25	3.32	0.12
Avg Relative Delay	77.45%	22.61%	85.78%	14.22%

In order to test the quality of the suggested routes by RSB-ATC, we input the route plan arising from RSB-ATC into scheduling algorithm SB-ATC-ReOpt which is shown in Section 5.4.2 to have the best performance among SB algorithms proposed for train scheduling problem. The instance considered for this test is the original train timetable for 27 trains and the cycle length of 30 minutes. The total delay of timetable arising from scheduling algorithm is 48.17 minutes which can be reduced to 41.17 minutes by employing the suggested route by routing algorithm.

In general, RSB-ATC outperforms FCFS and RSB-ATC-ReOpt in terms of solution quality. Moreover, it has a record of %100 feasibility of the solutions across all instances. Although runtime of the FCFS algorithm is less than 0.1 second, the results are not superior regarding neither quality nor feasibility. In addition, RSB-ATC has a very short runtime about 7 seconds which makes it suitable for implementation in practice. Considering RSB-ATC performance, its viability in both tactical and operational planning decisions can be ensured.

It should be mentioned that no experiments were carried out for the MILP model in Subsection 6.2.1. The main limitation is the presence of $\bar{M}$ in the constraints 6.2 and 6.4 which does not produce a strong formulation. Consequently, this issue results in long computational time for the MILP solution. Another issue is about the additional details of the model with regard to parallel machines which can add to the difficulty of the formulation. The capability of the model is in reducing the size of the problem which is due to aggregation of the railway network information. There are less number of the variables and constraints without losing the accuracy of the operational data. In addition, special formulation of the MPM-JSS model helps with developing EXT-SB algorithm for a parallel machine job shop scheduling problem. This explains why we have smaller computational time and higher quality of the SB solution for the train routing and scheduling problem in comparison with the train scheduling problem. Another advantage of the proposed

MILP model for smaller size instances is that routing and routing decisions are made simultaneously by solving a parallel machine job shop scheduling problem. More details for the benefits of the proposed MILP are given in Section 6.2.

6.5 Summary

This chapter addresses train routing and rerouting problem and models the problem with details about train movements and network topology in a mesoscopic level. A generic mathematical programming model is suggested to incorporate main safety and operational constraints with regard to the relevant level of details for trains and tracks. In addition, an extended disjunctive graph (MPM-JSS) is represented which modifies the classic disjunctive graph to formulate a parallel machine job shop scheduling problem with release times to minimize total weighted tardiness.

Planning in mesoscopic level with MPM-JSS formulation, reduces the number of variables and constraints and consequently gives us the opportunity to explore bigger networks. Another special feature of this formulation is that it can combine scheduling and routing decisions. Hence, this formulation has a significant advantage for integrating different levels of planning process in railway industry.

Employing the MPM-JSS formulation, we propose a novel framework of the SB algorithm framework for parallel machine job shop scheduling problem with release times in order to minimize total weighted tardiness. Two variants of the so-called EXT-SB algorithm are suggested where one version incorporates re-optimization process (RSB-ATC-ReOpt) and the other one does not include it (RSB-ATC).

Computational experiments are performed for different scenarios of disruption including blocked tracks on single and multiple track segments, longer running/dwell times and late departure of trains from their origin. Different categories of minor, general and major disruptions are also examined. Instances are generated based on a case study in London Bridge area. The complexity of the infrastructure and congestion of the network has been discussed earlier in Section 3.6.

Our experimental results show a strong performance of the RSB-ATC algorithm. The average run time of the algorithm is about 7 seconds and the percentage of feasible solutions is 100% across all instances. Short runtime of the RSB-ATC algorithm and and its ability to find high quality solutions makes it suitable for implementation in practice.

Test results also represent that in the route plans and schedules arising from RSB-ATC algorithm, the delay is focused more on the late trains compared to the on-time trains. Whereas, FCFS spreads the effect of delay relatively more to on-time trains. Thus, RSB-ATC algorithm can efficiently prevent delays to propagate throughout the network. Benefit of the suggested routes by routing algorithm is also presented through a test on the original timetable. Delay can be significantly reduced if trains are operated according to the new routes.

Further research can be addressed to integrate our routing algorithm in mesoscopic level into our scheduling approach in microscopic level. This would allow to improve the quality of the micro schedules with efficient routing. Another direction is to explore the performance of the routing algorithm in bigger networks. Further study can incorporate more realistic constraints such as lost connections when a route is partially blocked in a big rail network. One can also investigate the possibility of suggesting efficient routes by prioritizing different types of train traffic. Assigning weights to various operators who share the same part of the network and its effect on routing decisions can be also of interest.

Chapter 7

Concluding Remarks

Nowadays, there are various ongoing projects and future plans in many countries to transform the railway transportation. Because it is one of the major transport systems for both passenger and freight and there are huge opportunities to unlock capacity of the existing railway system for improving customer satisfaction and reducing cost. One of the major challenges for the industry is optimizing system performance through decision making tools which work based on functions and algorithms in order to optimize railway operation and control. Operational research and in particular scheduling and routing has significant potential to offer algorithmic solution approaches to improve railway operation and control.

7.1 Conclusion

This thesis focuses on providing models and algorithms for train scheduling and routing problems. The aim is to design algorithms to obtain good solutions in a reasonable amount of time so that they can be implemented in practice. The suggested models and algorithms for train scheduling are presented in Chapter 5. Chapter 6 offers models and algorithms for train routing problem.

We introduce generic models and several algorithms for train scheduling and disruption management in a microscopic level including detailed information about train movements and topology of the network. We aim to address the train scheduling and rescheduling in order to minimize delay propagation in the tactical and operational levels, respectively.

Train scheduling and rescheduling is modeled as a modified job shop scheduling problem with additional operational and safety constraints to minimize total weighted tardiness. A MILP model and a modified disjunctive graph are represented to formulate the modified job shop problem. The novelty of the suggested models are firstly in their ability to formulate special operational and safety rules and in particular signaling system of the UK rail network and secondly in their flexibility to model the other railway networks.

A mathematical programming approach, a simple heuristic and several variants of a modified Shifting Bottleneck heuristic are proposed for solving train scheduling and rescheduling problems. The mathematical programming approach is implemented with a commercial optimization package. The developed heuristic has a mechanism to avoid infeasibility of the solution as simply finding a feasible solution is nontrivial and dispatching rules such as FCFS result in infeasible solutions and consequently deadlock in the rail network.

Modified Shifting Bottleneck heuristic is an approach which decomposes job shop problems into single machine subproblems in order to solve them. Hence, the quality of the algorithm depends on the subproblem solutions. Classic SB is modified to solve subproblems by minimizing total weighted tardiness in a single machine problem with release times. Special modifications are required to avoid infeasibility in the solution method.

Computational experiments are performed on a case study of a complicated and congested partial network in Kent area. A timetable of 27 trains is considered which cycles every 30 minutes. Several scenarios of disruptions in terms of late departure of trains from their origin, extended running/dwell times of trains are assessed in three categories of minor, general and major disruptions. Results show that the mathematical programming approach and SB algorithm variants outperform the simple heuristic in terms of solution quality and runtime.

In addition, the mathematical programming approach is quite effective for smaller delays categorized as minor and general disruptions. However, when delays become bigger, the ability of the mathematical program and the simple heuristic diminishes in terms of feasibility and quality of the solution. The so-called SB-ATC-ReOpt algorithm, a particular version of the SB algorithms which considers ATC index and re-optimization step, provides significant improvements relative to the mathematical programming approach and the EA heuristic when trains are severely delayed. With 100% feasible solutions across all instances for different disruption scenarios, this variant of the SB algorithm seems to be highly reliable with regard to avoiding infeasibility and therefore avoiding deadlock in the rail network.

It should be noted that both the mathematical programming approach and SB algorithms have runtimes about 400 seconds which is within the operators time limit for incident/service update. Thus, both approaches can be implemented in rail operations when strict real-time limits apply.

In the second part of the research, the train routing problem and disruption management are studied. We investigate the impact of alternative routes for minimizing delay propagation in the network in a mesoscopic level of detail for trains movements and track topology. This thesis addresses the train routing problem in the tactical level to suggest routes in advance or in the operational level to reroute trains in response to a disruption.

The train routing and rerouting problem is formulated as a modified parallel machine job shop scheduling problem. A MILP model and a modified disjunctive graph represent the modified parallel machine job shop problem. Although models can formulate the UK railway specifications, they are quite generic and can be adapted to the other railway networks. A special characteristic of these models is that train routing decisions include order and timing of the trains. Thus, train routing and scheduling are integrated which is an interesting feature for railway industry.

Two variants of a modified SB algorithm are proposed and their performance is examined relative to a FCFS algorithm. Computational experiments report on the same case study in Kent area with a timetable with 27 trains and cycle length of 30 minutes. We examine the proposed solution methods for different types of disruption including blocked tracks on single and multiple tracks segments, extended running/dwell time and delayed departure.

Test results show substantial improvements of the route and schedules obtained by RSB-ATC, one of the SB algorithm variants. With a runtime of about 7 seconds and 100% feasible solutions found across all instances, this SB algorithm variant seems to be viable for practical use. Because it performs much better than the other SB algorithm and FCFS both in terms of solution quality and possibility of obtaining feasible solutions.

Moreover, the superior SB algorithm is shown to minimize delay propagation in a test for comparing relative delay of late and on-time trains. Benefits of routing with this algorithm is also tested by solving the scheduling algorithm, suggested in the first part of this thesis, for the original timetable with suggested new routes.

7.2 Extensions and future work

In this thesis, we have studied train scheduling and routing problems. Both problems are applicable in tactical and operational level planning. In order to model train scheduling and routing problems, we make use of the job shop scheduling formulation with additional operational and safety constraints.

Various algorithms based on the Shifting Bottleneck procedure, which is a well-known heuristic for job shop problems, are developed in order to minimize delay propagation in the rail network. Various SB algorithms have been developed to provide optimal or near-optimal solutions in a very short time. Experimental tests show promising results for solving these problems both in terms of solution quality and run-time. The chance of running into infeasibility appears to be low based on the experiments.

There are special features in the current research on train scheduling and train routing and scheduling problems which can be addressed in the future in depth. Problems studied in this thesis can be extended in terms of theoretical aspects of the study as mentioned in the following.

- The structural properties of the two problems can be explored. For instance, special features of the routes in the associated job shop problems can be addressed as there are some jobs with the same route as there are usually many following trains running on same tracks. There are also routes that the machine order of one job corresponds to a reverse machine order for the other one. This is the case when we have opposite trains which share the same tracks of the railway network partially or completely in their routes.

- The issue of equal processing times or almost equal processing times of the jobs can be considered. This is due to the fact that trains have equal running times on the same tracks when they all run according to the timetable. When a disruption happens, the running time of the train differs from the other trains running on the same disrupted tracks.

- The redundancy of the alternative arcs or disjunctive arcs can be incorporated in the the solution method. In our case study, the headway is nearly four times larger than the running times. So the disjunctive arcs which are associated with headway are redundant for the following trains when a disruption happens. Moreover, alternative arcs which are corresponding to the signaling system of the railway network are redundant for the following trains when there is no disruption.

- The theoretical aspects of the two formulated extended job shop scheduling problems can lead to development of better algorithms.

- A simple heuristic can be developed to ensure deadlock does not happen as it is not trivial to find feasible solutions for both scheduling and routing problems. This simple heuristic can be helpful for the comparison with the SB algorithms as the current suggested FCFS algorithms suggested for both problems do not always obtain feasible solutions.

With regard to practical aspects of the research, we suggest some directions to address future work based on this thesis as below.

- Integration of scheduling problem in microscopic level with routing and scheduling in mesoscopic level can be studied. Developing a laboratory decision making system to incorporate these algorithms as its main optimizing components can provide interesting pilot tools for rail industry.

- Further improvements in solution time and quality can be investigated by embedding more efficient heuristics in the current SB framework.

- Further enhancements of the two extended job shop problems in order to reflect the railway context can be addressed. Introducing more realistic features to both scheduling and routing problems can be considered. Some suggestions are listed as introducing mixed traffic of passengers and train with various priorities for services, considering lost train connections as an undesirable factor in planning, and incorporating train length as an important restriction to schedule and route trains into some platforms.

- Applying proposed algorithms on the other rail networks can provide a comparative assessment of performance of the algorithms which is quite rare in the literature mainly due to confidentiality issues.

Appendix A

Train Scheduling and Rescheduling: Experimental Results

Table A.1: Longer Running/Dwell Time - minor and general disruption: Delay and Average Delay

Block Delay	# of Block(s)	# of Train(s)	Instance	Xpress				EA Heuristic		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
				Delay	LR	Avg Delay per Train	Status	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train
Timetable			T01	**47.00**	0.00	1.74	-	55.67	2.06	51.17	1.90	48.17	1.78	48.17	1.78	48.17	1.78
5	1	3	P01	**94.50**	13.00	3.50	-	181.67	6.73	111.50	4.13	108.50	4.02	108.50	4.02	151.83	5.62
	2	5	P02	**126.67**	22.00	4.69	Optimal	270.00	10.00	249.50	9.24	234.67	8.69	223.83	8.29	278.00	10.30
10	1	3	P03	**155.17**	24.50	5.75	Optimal	328.50	12.17	208.00	7.70	208.00	7.70	208.00	7.70	248.33	9.20
	2	5	P04	**286.33**	47.50	10.60	-	624.00	23.11	328.33	12.16	315.17	11.67	315.17	11.67	Deadlock	NA
15	1	3	P05	**246.83**	42.50	9.14	-	410.33	15.20	252.00	9.33	249.00	9.22	249.00	9.22	282.83	10.48
	2	5	P06	**307.33**	68.00	11.38	Optimal	810.67	30.02	343.67	12.37	341.33	12.64	343.67	12.73	370.00	13.70
20	1	9	P07	**1032.00**	170.50	38.22	-	2153.00	79.74	1722.17	63.78	1508.00	55.85	117.67	43.51	Deadlock	NA
	2	12	P08	**989.83**	230.50	36.66	-	1357.75	50.29	1057.33	39.16	1030.08	38.15	1031.58	38.21	1136.00	42.07
25	1	9	P09	**1283.83**	214.00	47.55	-	Deadlock	NA	1814.50	67.20	1711.33	63.38	1711.33	63.38	1796.83	66.55
	2	12	P10	**1793.17**	286.50	66.41	-	Deadlock	NA	1942.00	71.93	1942.00	71.93	1947.33	72.12	11800.83	66.70
30	1	9	P11	**1019.50**	203.50	37.76	-	Deadlock	NA	1384.50	51.28	1023.50	37.91	1023.50	37.91	Deadlock	NA
	2	12	P12	**2337.17**	354.50	86.56	-	2860.33	105.94	2395.92	88.74	Deadlock	NA	2376.42	88.02	2395.75	88.73
	Avg Delay	All Instances		747.64				905.19		912.35		**726.65**		827.78		850.86	
		All Instances Excl. Unsolved Instances		**281.05**				487.80		324.74		317.11		316.11		359.31	
	Avg Delay per Train	All Instances		27.69				33.52		33.79		**26.91**		30.66		31.51	
		All Instances Excl. Unsolved Instances		**10.41**				18.07		12.03		11.74		11.71		13.31	
	Avg Runtime (secs)			400				2352		**349**		402		411		416	

Table A.2: Longer Running/Dwell Time - minor and general disruption: Average RD wrt EA Heuristic

Block	# of	# of	Instance	EA Heuristic	Xpress		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
Delay	Block(s)	Train(s)		Delay	Delay	RD	Delay	RD	Delay	RD	Delay	RD	Delay	RD
Timetable			T01	55.67	47.00	**15.57%**	51.17	8.08%	48.17	13.47%	48.17	13.47%	48.17	13.47%
5	1	3	P01	181.67	94.50	**47.98%**	111.50	38.62%	108.50	40.28%	108.50	40.28%	151.83	16.42%
	2	5	P02	270.00	126.67	**53.09%**	249.50	7.59%	234.67	13.09%	223.83	17.10%	278.00	-2.96%
10	1	3	P03	328.50	155.17	**52.76%**	208.00	36.68%	208.00	36.68%	208.00	36.68%	248.33	24.40%
	2	5	P04	624.00	286.33	**54.11%**	328.33	47.38%	315.17	49.49%	315.17	49.49%	Deadlock	NA
15	1	3	P05	410.33	246.83	**39.85%**	252.00	38.59%	249.00	39.32%	249.00	39.32%	282.83	31.07%
	2	5	P06	810.67	307.33	**62.09%**	343.67	57.61%	341.33	57.89%	343.67	57.61%	370.00	54.36%
20	1	9	P07	2153.00	1032.00	**52.07%**	1722.17	20.01%	1508.00	29.96%	1174.67	45.44%	Deadlock	NA
	2	12	P08	135.75	989.83	**27.10%**	1057.33	22.13%	1030.08	24.13%	1031.58	24.02%	1136.00	16.33%
25	1	9	P09	Deadlock	1283.83	NA	1814.50	NA	1711.33	NA	1711.33	NA	1796.83	NA
	2	12	P10	Deadlock	1793.17	NA	1942.00	NA	1942.00	NA	1947.33	NA	1800.83	NA
30	1	9	P11	Deadlock	1019.50	NA	1384.50	NA	1023.50	NA	1023.50	NA	Deadlock	NA
	2	12	P12	2860.33	2337.17	**18.29%**	2395.92	16.24%	Deadlock	NA	2376.42	16.92%	2395.75	16.24%
		Avg Deviation	All Instances		**42.29%**		29.29%		33.81%		34.03%		21.17%	
			Instances Excl. Unsolved Soln's		**42.63%**		29.90%		32.12%		32.64%		21.87%	

TABLE A.3: Longer Running/Dwell Time - minor and general disruption: Average RD wrt Xpress

Block Delay	# of Block(s)	# of Train(s)	Instance	Xpress Delay	EA Heuristic Delay	EA Heuristic RD	SB-ATC Delay	SB-ATC RD	SB-ATC-ReOpt-Rej Delay	SB-ATC-ReOpt-Rej RD	SB-ATC-ReOpt Delay	SB-ATC-ReOpt RD	SB-ASG-ReOpt Delay	SB-ASG-ReOpt RD
Timetable			T01	47.00	55.67	-18.44%	51.17	−8.87%	48.17	**−2.48%**	48.17	**−2.48%**	48.17	**−2.48%**
5	1	3	P01	94.50	181.67	−92.24%	111.50	−17.99%	108.50	**−14.81%**	151.83	**−14.81%**	151.83	−60.67%
	2	5	P02	126.67	270.00	−113.16%	249.50	−96.97%	234.67	−85.26%	223.83	**−76.71%**	278.00	−119.47%
10	1	3	P03	155.17	328.50	−111.17%	208.00	**−34.05%**	208.00	**−34.05%**	208.00	**−34.05%**	248.33	−60.04%
	2	5	P04	286.33	624.00	−117.93%	328.33	−14.67%	315.17	**−10.07%**	315.17	**−10.07%**	Deadlock	NA
15	1	3	P05	246.83	410.33	−66.24%	252.00	−2.09%	249.00	**−0.88%**	249.00	**−0.88%**	282.83	−14.58%
	2	5	P06	307.33	810.67	−163.77%	343.67	−11.82%	341.33	**−11.06%**	343.67	−11.82%	370.00	−20.39%
20	1	9	P07	1032.00	2153.00	−108.62%	1722.17	−66.88%	1508.00	−46.12%	1174.67	**−13.82%**	Deadlock	NA
	2	12	P08	989.83	1357.75	−37.17%	1057.33	−6.82%	1030.08	**−4.07%**	1031.58	−4.22%	1136.00	−14.77%
25	1	9	P09	1283.83	Deadlock	NA	1814.50	−41.33%	1711.33	**−33.30%**	1711.33	**−33.30%**	1796.83	−39.96%
	2	12	P10	1793.17	Deadlock	NA	1942.00	−8.30%	1942.00	−8.30%	1947.33	−8.60%	1800.83	**−0.43%**
30	1	9	P11	1019.50	Deadlock	NA	1384.50	−35.80%	1023.50	**−0.39%**	1023.50	**−0.39%**	Deadlock	NA
	2	12	P12	2337.17	2860.33	−22.38%	2395.92	−2.51%	Deadlock	NA	2376.42	**−1.68%**	2395.75	−2.51%
		Avg Deviation	All Instances		−85.17%		−26.78%		−20.90%		**−16.37%**		−33.53%	
			Instances Excl. Unsolved Soln's		−86.10%		−25.52%		−21.80%		**−20.70%**		−41.77%	

TABLE A.4: Late Departure - minor and general disruption: Delay and Average Delay

Late	# of	Instance	Xpress				EA Heuristic		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
Departure	Train(s)		Delay	LR	Avg Delay per Train	Status	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train
Timetable		T01	**47.00**	0.00	1.74	-	55.67	2.06	51.17	1.90	48.17	1.78	48.17	1.78	48.17	1.78
5	1	R01	**66.33**	5.00	2.46	-	92.33	3.42	72.00	2.67	75.83	2.81	75.83	2.81	72.33	2.68
	3	R02	**60.17**	13.00	2.23	-	84.50	3.13	75.50	2.80	72.50	2.69	70.67	2.62	67.17	2.49
	5	R03	**73.67**	18.00	2.73	-	107.33	3.98	91.83	3.40	80.33	2.98	80.33	2.98	89.50	3.31
10	1	R04	**70.67**	10.00	2.62	-	85.00	3.15	97.00	3.59	94.00	3.48	94.00	3.48	85.17	3.15
	3	R05	**82.17**	26.00	3.04	Optimal	91.50	3.39	90.00	3.33	84.83	3.14	84.83	3.14	85.83	3.18
	5	R06	**141.83**	44.50	5.25	-	181.17	6.71	165.67	6.14	146.00	5.41	146.00	5.41	143.17	5.30
15	1	R07	**67.67**	15.00	2.51	-	99.83	3.70	77.33	2.86	74.67	2.77	70.17	2.60	74.17	2.75
	3	R08	**123.00**	39.50	4.56	-	139.50	5.17	140.17	5.19	129.17	4.78	129.00	4.78	130.83	4.85
	5	R09	**171.33**	69.50	6.35	-	173.50	6.43	182.50	6.76	184.50	6.83	189.33	7.01	189.33	7.01
20	6	R10	**182.33**	115.00	6.75	-	230.00	8.52	211.83	7.85	192.50	7.13	184.83	6.85	212.17	7.86
	9	R11	**240.50**	171.50	8.91	-	286.67	10.62	265.17	9.82	252.33	9.35	252.33	9.35	272.00	10.07
	12	R12	**347.33**	229.00	12.86	-	381.83	14.14	365.00	13.52	348.83	12.92	353.17	13.08	364.67	13.51
25	6	R13	**230.67**	143.50	8.54	-	270.83	10.03	253.67	9.40	254.67	9.10	240.33	8.90	252.67	9.36
	9	R14	**275.67**	215.50	10.21	-	310.83	11.51	303.83	11.25	287.17	10.64	298.50	11.06	295.50	10.94
	12	R15	**379.33**	287.50	14.05	-	434.00	16.07	420.17	15.56	397.00	14.70	397.00	14.70	393.50	14.57
Avg Delay	All Instances		**159.98**				189.03		178.93		169.59		169.66		173.51	
	All Instances Excl. Unsolved Instances		**159.98**				189.03		178.93		169.59		169.66		173.51	
Avg Delay per Train	All Instances		**5.92**				7.00		6.63		6.28		6.28		6.43	
	All Instances Excl. Unsolved Instances		**5.92**				7.00		6.63		6.28		6.28		6.43	
Avg Runtime (secs)			400				2442		**346**		409		410		402	

Table A.5: Late Departure - minor and general disruption: Average RD wrt EA Heuristic

Late Departure	# of Train(s)	Instance	EA Heuristic	Xpress		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
			Delay	Delay	RD	Delay	RD	Delay	RD	Delay	RD	Delay	RD
Timetable		T01	55.67	47.00	**15.57%**	51.17	8.08%	48.17	13.47%	48.17	13.47%	48.17	13.47%
5	1	R01	92.33	66.33	**28.16%**	72.00	22.02%	75.83	17.87%	75.83	17.87%	72.33	21.66%
	2	R02	84.50	60.17	**28.80%**	75.50	10.65%	72.50	14.20%	70.67	16.37%	67.17	20.51%
	5	R03	107.33	73.67	**31.37%**	91.83	14.44%	80.33	25.16%	80.33	25.16%	89.50	16.61%
10	1	R04	85.00	70.67	**16.86%**	97.00	−14.12%	94.00	−10.59%	94.00	−10.59%	85.17	−0.20%
	2	R05	91.50	82.17	**10.20%**	90.00	1.64%	84.83	7.29%	84.83	7.29%	85.83	6.19%
	5	R06	181.17	141.83	**21.71%**	165.67	8.56%	146.00	19.41%	146.00	19.41%	143.17	20.98%
15	1	R07	99.83	67.67	**32.22%**	77.33	22.54%	74.67	25.21%	70.17	29.72%	74.17	25.71%
	2	R08	139.50	123.00	**11.83%**	140.17	−0.48%	129.17	7.41%	129.00	7.53%	130.83	6.21%
	5	R09	173.50	171.33	**1.25%**	182.50	−5.19%	184.50	−6.34%	189.33	−9.13%	189.33	−9.13%
20	6	R10	230.00	182.23	**20.27%**	211.83	7.90%	192.50	16.30%	184.83	19.64%	212.17	7.75%
	9	R11	286.67	240.50	**16.10%**	265.17	7.50%	252.33	11.98%	252.33	11.98%	272.00	5.12%
	12	R12	381.83	347.33	**9.04%**	365.00	4.41%	348.83	8.64%	353.17	7.51%	364.67	4.50%
25	6	R13	270.83	230.67	**14.83%**	253.67	6.34%	245.67	9.29%	240.33	11.26%	252.67	6.71%
	9	R14	310.83	275.67	**11.31%**	303.83	2.25%	287.17	7.61%	298.50	3.97%	295.50	4.93%
	12	R15	434.00	379.33	**12.60%**	420.17	3.19%	397.00	8.53%	397.00	8.53%	393.50	9.33%
	Avg Deviation	All Instances		**17.66%**		6.23%		10.96%		11.25%		10.02%	
		Instances Excl. Unsolved Soln's		**17.66%**		6.23%		10.96%		11.25%		10.02%	

Table A.6: Late Departure - minor and general disruption: Average RD wrt Xpress

Late Departure	# of Train(s)	Instance	Xpress Delay	EA Heuristic Delay	EA Heuristic RD	SB-ATC Delay	SB-ATC RD	SB-ATC-ReOpt-Rej Delay	SB-ATC-ReOpt-Rej RD	SB-ATC-ReOpt Delay	SB-ATC-ReOpt RD	SB-ASG-ReOpt Delay	SB-ASG-ReOpt RD
Timetable		T01	47.00	55.67	-18.44%	51.17	−8.87%	48.17	**−2.48%**	48.17	**−2.48%**	48.17	**−2.48%**
5	1	R01	66.33	92.33	−39.20%	72.00	**−8.54%**	75.83	−14.32%	75.83	−14.32%	72.33	−9.05%
	3	R02	60.17	84.50	−40.44%	75.50	−25.48%	72.50	−20.50%	70.67	−17.45%	67.17	**−11.63%**
	5	R03	73.67	107.33	−45.70%	91.83	−24.66%	80.33	**−9.05%**	80.33	**−9.05%**	89.50	−21.49%
10	1	R04	70.67	85.00	**−20.28%**	97.00	−37.26%	94.00	−33.02%	94.00	−33.02%	85.17	−20.52%
	3	R05	82.17	91.50	−11.36%	90.00	−9.53%	84.83	**−3.25%**	84.83	**−3.25%**	85.83	−4.46%
	5	R06	141.83	181.17	−27.73%	165.67	−16.80%	146.00	−2.94%	146.00	−2.94%	143.17	**−0.94%**
15	1	R07	67.67	99.83	−47.54%	77.33	−14.29%	74.67	−10.34%	70.17	**−3.69%**	74.17	−9.61%
	3	R08	123.00	139.50	−13.41%	140.17	−13.96%	129.17	−5.01%	129.00	**−4.88%**	130.83	−6.37%
	5	R09	171.33	173.50	**−1.26%**	182.50	−6.52%	184.50	−7.69%	189.33	−10.51%	189.33	−10.51%
20	6	R10	182.33	230.00	−26.14%	211.83	−16.18%	192.50	−5.58%	184.83	**−1.37%**	212.17	−16.36%
	9	R11	240.50	286.67	−19.20%	265.17	−10.26%	252.33	**−4.92%**	252.33	**−4.92%**	272.00	−13.10%
	12	R12	347.33	381.83	−9.93%	365.00	−5.09%	348.83	**−0.43%**	353.17	−1.68%	364.67	−4.99%
25	6	R13	230.67	270.83	−17.41%	253.67	−9.97%	245.67	−6.50%	240.33	**−4.19%**	252.67	−9.54%
	9	R14	275.67	310.83	−12.76%	303.83	−10.22%	287.17	**−4.17%**	298.50	−8.28%	295.50	−7.19%
	12	R15	379.33	434.00	−14.41%	420.17	−10.76%	397.00	−4.66%	397.00	−4.66%	393.50	**−3.73%**
	Avg Deviation	All Instances		−22.83%		−14.27%		−8.43%		**−7.92%**		−9.50%	
		Instances Excl. Unsolved Soln's		−22.83%		−14.27%		−8.43%		**−7.92%**		−9.50%	

Table A.7: Longer Running/Dwell Time - major disruption: Delay and Average Delay

Block Delay	# of Block(s)	Instance	Xpress				EA Heuristic		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
			Delay	LR	Avg Delay per Train	Status	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train
Timetable		T01	**47.00**	0.00	1.74	-	55.67	2.06	51.17	1.90	48.17	1.78	48.17	1.78	48.17	1.78
40	4	HP01	8954.67	1249.50	331.65	-	Deadlock	NA	9574.83	354.62	8279.67	306.65	7465.50	**276.50**	Deadlock	NA
	5	HP02	8082.33	1369.50	299.35	-	Deadlock	NA	8473.50	313.83	Deadlock	NA	7913.17	**293.08**	8230.67	304.84
	6	HP03	**8135.17**	1609.50	301.30	-	9600.33	355.57	8456.83	313.22	8514.33	315.35	8423.33	311.98	8535.17	316.12
50	4	HP04	**10260.70**	1569.50	380.03	-	Deadlock	NA	12008.80	444.77	10386.00	384.67	10380.00	384.44	Deadlock	NA
	5	HP05	NA	NA	NA	No Ans	13433.30	497.90	15156.50	561.35	12835.30	475.38	11483.40	425.31	10717.80	**396.96**
	6	HP06	8693.50	1622.50	321.98	-	Deadlock	NA	18494.00	**314.59**	8494.00	**314.59**	8494.00	**314.59**	Deadlock	NA
60	4	HP07	NA	NA	NA	No Ans	Deadlock	NA	14607.70	541.03	12672.50	469.35	12067.80	446.96	11945.80	**442.44**
	5	HP08	NA	NA	NA	No Ans	Deadlock	NA	12023.80	445.33	Deadlock	NA	10900.30	**403.71**	Deadlock	NA
	6	HP09	9602.00	1949.50	355.63	-	10839.30	401.46	9542.67	353.43	9542.67	353.43	9542.17	**353.41**	9558.83	354.03
70	4	HP10	14549.20	2209.50	538.86	-	Deadlock	NAA	16828.80	623.29	Deadlock	NA	14537.20	**538.41**	Deadlock	NA
	5	HP11	NA	NA	NA	No Ans	Deadlock	NAA	12081.10	447.45	12081.10	447.45	12070.60	**447.06**	Deadlock	NA
	6	HP12	14120.30	2489.50	522.97	-	15722.00	582.30	13567.80	502.51	13567.80	502.51	13567.80	502.51	13147.80	**486.96**
80	4	HP13	16363.30	2769.50	606.05	-	Deadlock	NA	20469.60	758.13	17953.20	664.93	16984.50	629.06	16344.90	**605.37**
	5	HP14	15047.80	2449.50	557.33	-	Deadlock	NA	14819.30	548.86	14819.30	548.86	14819.30	548.86	14811.20	**548.56**
	6	HP15	NA	NA	NA	No Ans	Deadlock	NA	19767.30	732.12	17509.30	648.49	17178.00	**636.22**	Deadlock	NA
Avg Delay	All Instances		10350.54				**9932.12**		12245.23		11284.87		10992.20		10371.15	
	All Instances Excl. Unsolved Instances		7976.12				9054.32		7904.62		7918.24		7895.37		**7822.49**	
Avg Delay per Train	All Instances		383.35				**367.86**		453.53		417.96		407.12		384.12	
	All Instances Excl. Unsolved Instances		295.41				335.34		292.76		293.27		292.42		**289.72**	
Avg Runtime (secs)			400				2352		**352**		426		413		415	

TABLE A.8: Longer Running/Dwell Time - major disruption: Average RD wrt EA Heuristic

Late Departure	# of Train(s)	Instance	EA Heuristic	Xpress		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
			Delay	Delay	RD	Delay	RD	Delay	RD	Delay	RD	Delay	RD
Timetable		T01	55.67	47.00	**15.57%**	51.17	8.08%	48.17	13.47%	48.17	13.47%	48.17	13.47%
40	4	HP01	Deadlock	9574.83	NA	9574.83	NA	8279.67	NA	7465.50	NA	Deadlock	NA
	5	HP02	Deadlock	8473.50	NA	8473.50	NA	Deadlock	NA	7913.17	NA	8230.67	NA
	6	HP03	9600.33	8456.83	**15.26%**	8456.83	11.91%	8514.33	11.31%	8423.33	12.26%	8535.17	11.10%
50	4	HP04	Deadlock	12008.80	NA	12008.80	NA	10386.00	NA	10380.00	NA	Deadlock	NA
	5	HP05	13443.30	15156.50	NA	15156.60	−12.74%	12835.30	4.52%	11483.40	14.58%	10717.80	**20.27%**
	6	HP06	Deadlock	8494.00	NA	8494.00	NA	8494.00	NA	8494.00	NA	Deadlock	NA
60	4	HP07	Deadlock	14607.70	NA	14607.70	NA	12672.50	NA	12067.80	NA	11945.80	NA
	5	HP08	Deadlock	12023.80	NA	12023.80	NA	Deadlock	NA	10900.30	NA	Deadlock	NA
	6	HP09	10839.30	9542.67	11.41%	9542.67	11.96%	9542.67	11.96%	9542.17	**11.97%**	9558.83	11.81%
70	4	HP10	Deadlock	16828.80	NA	16828.80	NA	Deadlock	NA	14537.20	NA	Deadlock	NA
	5	HP11	Deadlock	12081.10	NA	12081.10	NA	12081.10	NA	12070.60	NA	Deadlock	NA
	6	HP12	15722.00	13567.80	10.19%	13567.80	13.70%	13567.80	13.70%	13567.80	13.70%	13147.80	**16.37%**
80	6	HP13	Deadlock	20469.60	NA	20469.60	NA	17953.20	NA	16984.50	NA	16344.90	NA
	9	HP14	Deadlock	14819.30	NA	14819.30	NA	14819.30	NA	14819	NA	14811.20	NA
	12	HP15	Deadlock	19767.30	NA	19767.30	NA	17509.30	NA	17178.00	NA	Deadlock	NA
	Avg Deviation	All Instances		13.11%		6.58%		10.99%		13.20%		**14.61%**	
		Instances Excl. Unsolved Soln's		13.11%		11.41%		12.61%		12.85%		**13.19%**	

Table A.9: Longer Running/Dwell Time - major disruption: Average RD wrt Xpress

Late	# of	Instance	Xpress	EA Heuristic		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
Departure	Train(s)		Delay	Delay	RD	Delay	RD	Delay	RD	Delay	RD	Delay	RD
Timetable		T01	47.00	55.67	-18.44%	51.17	−8.87%	48.17	**−2.48%**	48.17	**−2.48%**	48.17	**−2.48%**
40	4	HP01	8594.67	Deadlock	NA	9574.83	−6.93%	8279.67	7.54%	7465.50	**16.63%**	Deadlock	NA
	5	HP02	8082.33	Deadlock	NA	8473.50	−4.84%	Deadlock	NA	7913.17	**2.09%**	8230.67	−1.84%
	6	HP03	8135.17	9600.33	−18.01%	8456.83	−3.95%	8514.33	−4.66%	8423.33	**−3.54%**	8535.17	−4.92%
50	4	HP04	10260.70	Deadlock	NA	12008.80	−17.04%	10386.00	−1.22%	10380.00	**−1.16%**	Deadlock	NA
	5	HP05	NA	13443.30	NA	15156.50	NA	12835.30	NA	11483.40	NA	10717.80	NA
	6	HP06	8693.50	Deadlock	NA	8494.00	**2.29%**	8494.00	**2.29%**	8494.00	**2.29%**	Deadlock	NA
60	4	HP07	NA	Deadlock	NA	14607.70	NA	12672.50	NA	12067.80	NA	11945.80	NA
	5	HP08	NA	Deadlock	NA	12023.80	NA	Deadlock	NA	10900.30	NA	Deadlock	NA
	6	HP09	9602.00	10839.30	−12.89%	9542.67	0.62%	9542.67	0.62%	9542.17	**0.62%**	9558.83	0.45%
70	4	HP10	14549.20	Deadlock	NA	16828.80	−15.67%	Deadlock	NA	14537.20	**0.08%**	Deadlock	NA
	5	HP11	NA	Deadlock	NA	12081.10	NA	12081.10	NA	12070.60	NA	Deadlock	NA
	6	HP12	14120.30	15722.00	−11.34%	13567.80	3.91%	13567.80	3.91%	13567.80	3.91%	13147.80	6.89%
80	4	HP13	16363.30	Deadlock	NA	20469.60	−25.09%	17953.20	−9.72%	16984.50	−3.80%	16344.90	**0.11%**
	5	HP14	15047.80	Deadlock	NA	14819.30	1.52%	14819.30	1.52%	14819.30	1.52%	14811.20	**1.57%**
	6	HP15	NA	Deadlock	NA	19767.30	NA	17509.30	NA	17178.00	NA	Deadlock	NA
	Avg Deviation	All Instances		−15.17%		−6.73%		−0.24%		**−1.47%**		−0.03%	
		Instances Excl. Unsolved Soln's		−15.17%		−2.07%		−0.65%		−0.37%		**−0.02%**	

TABLE A.10: Longer Running/Dwell Time and Late departure - major disruption: Delay and Average Delay

Block Delay	# of Block(s)	Departure Delay	# of Train(s)	Instance	Xpress				EA Heuristic		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
					Delay	LR	Avg Delay per Train	Status	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train
Timetable				T01	**47.00**	0.00	1.74	-	55.67	2.06	51.17	1.90	48.17	1.78	48.17	1.78	48.17	1.78
40	4	10	3	PR01	7611.83	1279.50	281.92	-	Deadlock	NA	9574.83	354.62	8279.67	306.65	**7465.50**	276.50	Deadlock	NA
			5	PR02	8588.00	1299.50	318.07	-	Deadlock	NA	9598.83	355.51	8296.00	307.26	**8290.00**	307.04	Deadlock	NA
		20	9	PR03	NA	NA	NA	No Ans	Deadlock	NA	9646.17	357.27	8432.50	312.31	7997.00	296.19	**7931.67**	293.77
			12	PR04	**8080.33**	1489.50	299.27	-	Deadlock	NA	10206.50	378.02	8879.17	328.86	8248.17	305.49	9859.50	365.17
40	6	10	3	PR05	7278.50	1322.50	269.57	-	Deadlock	NA	**6816.00**	252.44	**6816.00**	252.44	**6816.00**	252.44	Deadlock	NA
			5	PR06	NA	NA	NA	No Ans	9549.33	353.68	8519.33	315.53	8576.83	317.66	8485.83	314.29	**8195.50**	303.54
		20	9	PR07	6751.58	1469.58	250.06	-	7607.00	281.74	**6519.75**	241.47	**6519.75**	241.47	6640.08	245.93	6640.08	245.93
			12	PR08	8641.50	1849.50	320.06	-	10904.70	403.88	8677.33	321.38	8570.83	317.44	**8482.33**	314.16	8959.17	331.82
50	4	10	3	PR09	10245.80	1599.50	379.47	-	Deadlock	NA	12019.30	445.16	10396.50	385.06	**9414.17**	348.67	Deadlock	NA
			5	PR10	10643.40	1769.50	394.20	-	Deadlock	NA	12789.60	473.69	11233.20	416.04	10598.70	392.54	**10207.80**	378.07
		20	9	PR11	9813.50	1749.50	363.46	-	Deadlock	NA	12510.50	463.35	10689.30	395.90	**9574.83**	354.62	Deadlock	NA
			12	PR12	**10204.50**	1859.50	377.94	-	11240.20	416.30	12821.70	474.88	10998.70	407.36	10390.70	384.84	10251.50	379.69
50	6	10	3	PR13	10070.10	1799.50	372.97	-	Deadlock	NA	9667.75	358.06	9667.75	358.06	9667.75	358.06	**9467.75**	350.66
			5	PR14	10878.80	2119.50	402.92	-	Deadlock	NA	12327.30	456.57	10909.30	404.05	**10702.30**	396.38	12373.20	471.75
		20	9	PR15	8777.00	1802.50	325.07	-	9912.83	367.14	**8595.00**	318.33	**8595.00**	318.33	**8595.00**	318.33	**8595.00**	318.33
			12	PR16	8579.83	1859.58	317.77	-	8704.67	322.40	8760.00	324.44	**8556.67**	316.91	8557.83	316.96	8568.17	317.34
		Avg Delay	All Instances		8414.11				8282.06		9358.89		8556.78		**8233.79**		8455.13	
			All Instances Excl. Unsolved Instances		7166.90				8070.84		7570.82		7214.85		**7119.02**		7177.01	
		Avg Delay per Train	All Instances		311.63				306.74		346.62		316.92		**304.95**		313.15	
			All Instances Excl. Unsolved Instances		265.44				298.92		280.40		267.22		**263.67**		265.82	
		Avg Runtime (secs)			400				2501		**367**		417		415		419	

TABLE A.11: Longer Running/Dwell Time and Late departure - major disruption: Average RD wrt EA Heuristic

Block	# of	Departure	# of	Instance	EA Heuristic	Xpress		SB-ATC		SB-ATC-ReOpt-Rej		SB-ATC-ReOpt		SB-ASG-ReOpt	
Delay	Block(s)	Delay	Train(s)		Delay	Delay	RD	Delay	RD	Delay	RD	Delay	RD	Delay	RD
Timetable				T01	55.67	47.00	**15.57%**	51.17	8.08%	48.17	13.47%	48.17	13.47%	48.17	13.47%
40	4	10	3	PR01	Deadlock	7611.83	NA	9574.83	NA	8279.67	NA	7465.50	NA	Deadlock	NA
			5	PR02	Deadlock	8588.00	NA	9598.83	NA	8296.00	NA	8290.00	NA	Deadlock	NA
		20	9	PR03	Deadlock	NA	NA	9646.17	NA	8432.50	NA	7997.00	NA	7931.67	NA
			12	PR04	Deadlock	8080.33	NA	10206.50	NA	8879.17	NA	8248.17	NA	9859.50	NA
40	6	10	3	PR05	Deadlock	7278.50	NA	6816.00	NA	6816.00	NA	6816.00	NA	Deadlock	NA
			5	PR06	9549.33	NA	NA	8519.33	10.79%	8576.83	10.18%	8485.83	11.14%	8195.50	**14.18%**
		20	9	PR07	7607.00	6751.58	11.25%	6519.75	**14.29%**	6519.75	**14.29%**	6640.08	12.71%	6640.08	12.71%
			12	PR08	10904.70	8641.50	20.75%	8677.33	20.43%	8570.83	21.40%	8482.33	**22.21%**	8959.17	17.84%
50	4	10	3	PR09	Deadlock	10245.80	NA	12019.30	NA	10396.50	NA	9414.17	NA	Deadlock	NA
			5	PR10	Deadlock	10643.40	NA	12789.60	NA	11233.20	NA	10598.70	NA	10207.80	NA
		20	9	PR11	Deadlock	9813.50	NA	12510.50	NA	10689.30	NA	9574.83	NA	Deadlock	NA
			12	PR12	11240.20	10204.50	**9.21%**	12821.70	−14.07%	10998.70	2.15%	10390.70	7.56%	10251.50	8.80%
50	6	10	3	PR13	Deadlock	10070.10	NA	9667.75	NA	9667.75	NA	9667.75	NA	9467.75	NA
			5	PR14	Deadlock	10878.80	NA	12327.30	NA	10909.30	NA	10702.30	NA	12737.20	NA
		20	9	PR15	9912.83	8777.00	11.46%	8595.00	**13.29%**	8595.00	**13.29%**	8595.00	**13.29%**	8595.00	**13.29%**
			12	PR16	8704.67	8579.83	1.43%	8760.00	−0.64%	8556.67	**1.70%**	8557.83	1.69%	8568.17	1.57%
			Avg Deviation	All Instances		11.61%		7.45%		10.93%		**11.72%**		11.69%	
				Instances Excl. Unsolved Soln's		11.69%		6.90%		11.05%		**11.82%**		11.28%	

TABLE A.12: Longer Running/Dwell Time and Late departure - major disruption: Average RD wrt Xpress

Block Delay	# of Block(s)	Departure Delay	# of Train(s)	Instance	Xpress Delay	EA Heuristic Delay	EA Heuristic RD	SB-ATC Delay	SB-ATC RD	SB-ATC-ReOpt-Rej Delay	SB-ATC-ReOpt-Rej RD	SB-ATC-ReOpt Delay	SB-ATC-ReOpt RD	SB-ASG-ReOpt Delay	SB-ASG-ReOpt RD
Timetable				T01	47.00	55.67	-18.44%	51.17	−8.87%	48.17	**−2.48%**	48.17	**−2.48%**	48.17	**−2.48%**
40	4	10	3	PR01	7611.83	Deadlock	NA	9574.83	−25.79%	8279.67	−8.77%	7465.50	**1.92%**	Deadlock	NA
			5	PR02	8588.00	Deadlock	NA	9598.83	−11.77%	8296.00	3.40%	8290.00	**3.47%**	Deadlock	NA
		20	9	PR03	NA	Deadlock	NA	9646.17	NA	8432.50	NA	7997.00	NA	7931.67	NA
			12	PR04	8080.33	Deadlock	NA	10206.50	−26.31%	8879.17	−9.89%	8248.17	**−2.08%**	9859.50	−22.02%
40	6	10	3	PR05	7278.50	Deadlock	NA	6816.00	**6.35%**	6816.00	**6.35%**	6816.00	**6.35%**	Deadlock	NA
			5	PR06	NA	9549.33	NA	8519.33	NA	8576.83	NA	8485.83	NA	8195.50	NA
		20	9	PR07	6751.58	7607.00	−12.67%	6519.75	**3.43%**	6519.75	**3.43%**	6640.08	1.65%	6640.08	1.65%
			12	PR08	8641.50	10904.70	−26.19%	8677.33	−0.41%	8570.83	0.82%	8482.33	**1.84%**	8959.17	−3.68%
50	4	10	3	PR09	10245.80	Deadlock	NA	12019.30	−17.31%	10396.50	−1.47%	9414.17	**8.12%**	Deadlock	NA
			5	PR10	10643	Deadlock	NA	12789.60	−20.16%	11233.20	−5.54%	10598.70	0.42%	10207.80	**4.09%**
		20	9	PR11	9813.50	Deadlock	NA	12510.50	−27.48%	10689.30	−8.92%	9574.83	**2.43%**	Deadlock	NA
			12	PR12	10204.50	11240.20	−10.15%	12821.70	−25.65%	10998.70	−7.78%	10390.70	−1.82%	10251.50	**−0.46%**
50	6	10	3	PR13	10070.10	Deadlock	NA	9667.75	4.00%	9667.75	4.00%	9667.75	4.00%	9467.75	**5.98%**
			5	PR14	10878.80	Deadlock	NA	12327.30	−13.31%	10909.30	−0.28%	10702.30	**1.62%**	12737.20	−17.08%
		20	9	PR15	8777.00	9912.83	−12.94	8595.00	**2.07%**	8595.00	**2.07%**	8595.00	**2.07%**	8595.00	**2.07%**
			12	PR16	8579.83	8704.67	−1.46%	8760.00	−2.10%	8556.67	**0.27%**	8557.83	0.26%	8568.17	0.14%
			Avg Deviation	All Instances		−13.64%		−10.89%		−1.65%		**−1.85%**		−3.18%	
				Instances Excl. Unsolved Soln's		−13.64%		−5.25%		−0.61%		**−0.25%**		−0.46%	

Appendix B

Train Routing and Rerouting: Experimental Results

TABLE B.1: Blocked Tracks on a Single Track Section: Delay and Average Delay

Track Segment(s)	# of Blocked Track(s)	# of Parallel Tracks	% of Unavailable Tracks	Instance	FCFS		SB-ATC		SB-ATC-ReOpt	
					Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train
Timetable				RT01	6.00	0.22	**0.50**	0.02	**0.50**	0.02
2	1	2	50	RS01	7.00	0.26	**2.00**	0.07	**2.00**	0.07
4	1	2	50	RS02	7.00	0.26	**2.00**	0.07	**2.00**	0.07
5	1	2	50	RS03	6.00	0.22	**0.50**	0.05	**0.50**	0.02
8	1	2	50	RS04	19.00	0.70	**17.00**	0.63	**17.00**	0.63
11	1	2	50	RS05	7.50	0.28	**3.50**	0.13	40.00	1.48
15	1	2	50	RS06	6.50	0.24	**0.50**	0.02	**0.50**	0.02
17	1	2	50	RS07	6.50	0.24	**0.50**	0.02	**0.50**	0.02
9	1	3	33	RS08	7.00	0.26	**4.50**	0.17	4.50	0.17
	2		67	RS09	Deadlock	NA	**36.50**	1.35	66.50	2.46
10	1	3	33	RS10	6.00	0.22	**0.50**	0.02	**0.50**	0.02
	2		67	RS11	18.00	0.67	**17.50**	0.65	75.00	2.78
1	1	5	20	RS12	6.00	0.22	**0.50**	0.02	**0.50**	0.02
	2		40	RS13	6.00	0.22	**0.50**	0.02	**0.50**	0.02
	3		60	RS14	7.00	0.26	**2.00**	0.07	**2.00**	0.07
	4		80	RS15	39.00	1.44	**35.00**	1.30	**35.00**	1.30
7	1	5	20	RS16	6.00	0.22	**0.50**	0.02	**0.50**	0.02
	2		40	RS17	6.00	0.22	**0.50**	0.02	**0.50**	0.02
	3		60	RS14	6.50	0.24	**0.50**	0.02	**0.50**	0.02
	4		80	RS19	13.00	0.48	**9.50**	0.35	**28.00**	1.04
		Avg Delay	All Instances		9.79		**6.73**		13.85	
			Excluding the Unsolved Instances		9.79		**5.16**		11.08	
		Avg Delay per Train	All Instances		0.36		**0.25**		0.51	
			Excluding the Unsolved Instances		0.36		**0.19**		0.41	
		Avg Runtime (secs)			0.06		7.19		7.51	

TABLE B.2: Blocked Tracks on a Single Track Section: Average RD wrt FCFS

Track Segment(s)	# of Blocked Track(s)	# of Parallel Tracks	% of Unavailable Tracks	Instance	FCFS	SB-ATC		SB-ATC-ReOpt	
					Delay	Delay	RD	Delay	RD
Timetable				RT01	6.00	0.50	91.67%	0.50	91.67%
2	1	2	50	RS01	7.00	2.00	71.43%	2.00	71.43%
4	1	2	50	RS02	7.00	2.00	71.43%	2.00	71.43%
5	1	2	50	RS03	6.00	0.50	91.67%	0.50	91.67%
8	1	2	50	RS04	19.00	17.00	10.53%	17.00	10.53%
11	1	2	50	RB05	7.50	3.50	53.33%	40.00	−433.33%
15	1	2	50	RS06	6.50	0.50	92.31%	0.50	92.31%
17	1	2	50	RS07	6.50	0.50	92.31%	0.50	92.31%
9	1	3	33	RS08	7.00	4.50	35.17%	4.50	35.71%
	2		67	RS09	Deadlock	36.50	NA	66.50	NA
10	1	3	33	RS10	6.00	0.50	91.67%	0.50	91.67%
	2		67	RS11	18.00	17.50	2.78%	75.00	−316.67%
1	1	5	20	RS12	6.00	0.50	91.67%	0.50	91.67%
	2		40	RS13	6.00	0.50	91.67%	0.50	91.67%
	3		60	RS14	7.00	2.00	71.43%	2.00	71.43%
	4		80	RS15	39.00	35.00	10.26%	35.00	10.26%
7	1	5	20	RS16	6.00	0.50	91.67%	0.50	91.67%
	2		40	RS17	6.00	0.50	91.67%	0.50	91.67%
	3		60	RS14	6.50	0.50	92.31%	0.50	92.31%
	4		80	RS19	13.00	9.50	26.92%	28.00	-115.38%
			Avg RD	All Instances		66.97%		17.05%	
				Instances Excl. Unsolved Soln's		66.97%		17.05%	

Table B.3: Blocked Tracks on Multiple Track Sections: Delay and Average Delay

# of Disrupted Track Segment(s)	# of Blocked Track(s)	Spread of Blockage	# of Parallel Tracks	% of Unavailable Tracks	Instance	FCFS		SB-ATC		SB-ATC-ReOpt	
						Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train
Timetable					RT01	6.00	0.22	**0.50**	0.02	**0.50**	0.02
2	2	1,1	2,2	50,50	RB01	6.50	0.24	**0.50**	0.02	**0.50**	0.02
	3	1,2	2,3	50,67	RB02	18.00	0.67	**17.50**	0.65	75.00	2.78
	4	2,2	5,3	40,67	RB03	Deadlock	NA	**36.50**	1.35	66.50	2.46
	5	1,4	2,5	50,80	RB04	14.00	0.52	**11.00**	0.41	29.50	1.09
3	3	1,1,1	2,2,2	50,50,50	RB05	20.00	0.74	**18.50**	0.69	**18.50**	0.69
	4	2,1,1	3,2,2	67,50,50	RB06	Deadlock	NA	**19.50**	0.72	75.00	2.78
	5	1,3,1	2,5,3	50,60,33	RB07	8.00	0.30	**6.00**	0.22	**6.00**	0.22
	6	4,1,1	5,2,2	80,50,50	RB08	40.00	1.48	**36.00**	1.33	41.00	1.52
4	4	1,1,1,1	2,2,2,2	50,50,50,50	RB09	9.00	0.33	**5.00**	0.19	7.50	0.28
	5	2,1,1,1	5,2,2,2	40,50,50,50	RB10	19.50	0.72	**17.00**	0.63	**17.00**	0.63
	6	2,1,1,2	5,2,2,3	40,50,50,67	RB11	Deadlock	NA	**38.00**	1.41	68.00	2.52
	7	1,4,1,1	2,5,2,2	50,80,50,50	RB12	14.50	0.54	**11.00**	0.41	29.50	1.09
5	5	1,1,1,1,1	2,2,2,2,2	50,50,50,50,50	RB13	9.00	0.33	**5.00**	0.19	7.50	0.28
	6	1,1,2,1,1	2,2,5,2,2	50,50,40,50,50	RB14	20.50	0.76	**18.50**	0.69	**18.50**	0.69
	7	3,1,1,1,1	5,2,2,2,2	60,50,50,50,50	RB15	10.00	0.37	**6.50**	0.24	**6.50**	0.24
	8	1,3,2,1,1	2,5,3,2,2	50,60,60,50,50	RB16	Deadlock	NA	**21.00**	0.78	49.50	1.83
			Avg Delay	All Instances		**15.00**		15.76		30.38	
				Excluding the Unsolved Instances		15.00		**11.77**		19.81	
			Avg Delay per Train	All Instances		**0.55**		0.58		1.12	
				Excluding the Unsolved Instances		0.55		**0.44**		0.73	
			Avg Runtime (secs)			0.01		7.16		8.19	

Table B.4: Blocked Tracks on Multiple Track Sections: Average RD wrt FCFS

# of Disrupted Track Segment(s)	# of Blocked Track(s)	Spread of Blockage	# of Parallel Tracks	% of Unavailable Tracks	Instance	FCFS	SB-ATC		SB-ATC-ReOpt	
						Delay	Delay	RD	Delay	RD
Timetable					RT01	6.00	0.50	91.67%	0.50	91.67%
2	2	1,1	2,2	50,50	RB01	6.50	0.50	92.31%	0.50	92.31%
	3	1,2	2,3	50,67	RB02	18.00	17.50	2.78%	75.00	−316.67%
	4	2,2	5,3	40,67	RB03	Deadlock	36.50	NA	66.50	NA
	5	1,4	2,5	50,80	RB04	14.00	11.00	21.43%	29.50	−110.71%
3	3	1,1,1	2,2,2	50,50,50	RB05	20.00	18.50	7.50%	18.50	7.50%
	4	2,1,1	3,2,2	67,50,50	RB06	Deadlock	19.50	NA	75.00	NA
	5	1,3,1	2,5,3	50,60,33	RB07	8.00	6.00	25.00%	6.00	25.00%
	6	4,1,1	5,2,2	80,50,50	RB08	40.00	36.00	10.00%	41.00	−2.50%
4	4	1,1,1,1	2,2,2,2	50,50,50,50	RB09	9.00	5.00	44.44%	7.50	16.67%
	5	2,1,1,1	5,2,2,2	40,50,50,50	RB10	19.50	17.00	12.82%	17.00	12.82%
	6	2,1,1,2	5,2,2,3	40,50,50,67	RB11	Deadlock	38.00	NA	68.00	NA
	7	1,4,1,1	2,5,2,2	50,80,50,50	RB12	14.50	11.00	24.14%	29.50	−103.45%
5	5	1,1,1,1,1	2,2,2,2,2	50,50,50,50,50	RB13	9.00	5.00	44.44%	7.50	16.67%
	6	1,1,2,1,1	2,2,5,2,2	50,50,40,50,50	RB14	20.50	18.50	9.76%	18.50	9.76%
	7	3,1,1,1,1	5,2,2,2,2	60,50,50,50,50	RB15	10.00	6.50	35.00%	6.50	35.00%
	8	1,3,2,1,1	2,5,3,2,2	50,60,60,50,50	RB16	Deadlock	21.00	NA	49.50	NA
			Avg RD	All Instances			32.41%		−17.38%	
				Instances Excl. Unsolved Soln’s			32.41%		−17.38%	

Table B.5: Longer Running/Dwell times: Delay and Average Delay

Block Delay	# of Block(s)	# of Trains	Instance	FCFS		SB-ATC		SB-ATC-ReOpt	
				Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train
Timetable			RT01	6.00	0.22	**0.50**	0.02	**0.50**	0.02
5	1	3	RP01	**19.50**	0.72	20.00	0.74	85.00	3.15
	2	5	RP02	28.50	1.06	**24.00**	0.89	29.00	1.07
10	1	3	RP03	146.50	5.43	**134.00**	4.96	134.00	4.96
	2	5	RP04	**79.00**	2.93	152.00	5.63	152.00	5.63
15	1	3	RP05	168.00	6.22	**152.00**	5.63	**152.00**	5.63
	2	5	RP06	211.00	7.81	**192.00**	7.11	229.50	8.50
20	2	9	RP07	**274.00**	10.15	340.50	12.61	416.00	15.41
	3	12	RP08	637.50	23.61	**613.50**	22.72	**613.50**	22.72
25	2	9	RP09	Deadlock	NA	**626.00**	23.19	760.00	28.15
	3	12	RP10	1248.50	46.24	**1246.00**	46.15	**1246.00**	46.15
30	2	9	RP11	Deadlock	NA	**889.50**	32.94	1013.00	37.52
	3	12	RP12	1054.00	39.04	**1013.00**	37.52	1117.00	41.37
40	4	18	RP13	1685.50	62.43	**1462.50**	54.17	1718.50	63.65
	6	23	RP14	Deadlock	NA	**2349.50**	87.02	2395.50	88.72
50	4	18	RP15	2557.00	94.70	**2497.50**	92.50	2630.00	97.41
	6	23	RP16	Deadlock	NA	**2847.00**	105.44	3350.00	124.07
60	4	18	RP17	2200.00	81.48	**2006.50**	74.31	2321.00	85.96
	6	23	RP18	Deadlock	NA	**4919.50**	182.20	Deadlock	NA
	Avg Delay	All Instances		**736.79**		1130.82		1020.14	
		Excluding the Unsolved Instances		736.79		**703.86**		774.57	
	Avg Delay per Train	All Instances		**27.29**		41.88		37.78	
		Excluding the Unsolved Instances		27.29		**26.06**		28.69	
	Avg Runtime (secs)			0.01		7.21		9.90	

Table B.6: Longer Running/Dwell times: Average RD wrt FCFS

Block Delay	# of Block(s)	# of Trains	Instance	FCFS	SB-ATC		SB-ATC-ReOpt	
				Delay	Delay	RD	Delay	RD
Timetable			RT01	6.00	0.50	91.67%	0.50	91.67%
5	1	3	RP01	19.50	20.00	−2.56%	85.00	-335.90%
	2	5	RP02	28.50	24.00	15.79%	29.00	−1.75%
10	1	3	RP03	146.50	134.00	8.53%	134.00	8.53%
	2	5	RP04	79.00	152.00	−92.41%	152.00	−92.41%
15	1	3	RP05	168.00	152.00	9.52%	152.00	9.52%
	2	5	RP06	211.00	192.00	9.00%	229.50	−8.77%
20	2	9	RP07	274.00	340.50	−24.27%	416.00	−51.82%
	3	12	RP08	637.50	613.50	3.76%	613.50	3.76%
25	2	9	RP09	Deadlock	626.00	NA	760.00	NA
	3	12	RP10	1248.50	1246.00	0.20%	1246.00	0.20%
30	2	9	RP11	Deadlock	889.50	NA	1013.00	NA
	3	12	RP12	1054.00	1013.00	3.89%	1117.00	−5.89%
40	4	18	RP13	1685.50	1462.50	13.23%	1718.50	−1.96%
	6	23	RP14	Deadlock	2349.50	NA	2395.50	NA
50	4	18	RP15	2557.00	2497.50	2.33%	2630.00	−2.85%
	6	23	RP16	Deadlock	2847.00	NA	3350.00	NA
60	4	18	RP17	2200.00	2006.50	8.80%	2321.00	−5.50%
	6	23	RP18	Deadlock	4919.50	NA	Deadlock	NA
		Avg RD	All Instances		3.39%		−28.09%	
			Instances Excl. Unsolved Soln's		3.39%		−28.09%	

Table B.7: Late Departure: Delay and Average Delay

Late Departure	# of Trains	Instance	FCFS		SB-ATC		SB-ATC-ReOpt	
			Delay	Avg Delay per Train	Delay	Avg Delay per Train	Delay	Avg Delay per Train
Timetable		RT01	6.00	0.22	**0.50**	0.02	**0.50**	0.02
5	3	RR01	18.50	0.69	**13.00**	0.48	**13.00**	0.48
	5	RR02	31.00	1.15	**23.00**	0.85	**23.00**	0.85
10	3	RR03	43.00	1.59	**35.00**	1.30	**35.00**	1.30
	5	RR04	**57.00**	2.11	59.20	2.20	92.00	3.41
15	3	RR05	55.00	2.04	**47.00**	1.74	**47.00**	1.74
	5	RR06	**80.50**	2.98	**80.50**	2.98	108.00	4.00
20	9	RR07	185.50	6.87	**182.50**	6.76	445.00	16.48
	12	RR08	244.00	9.04	**242.50**	8.98	298.00	11.04
25	9	RR09	231.00	8.56	**220.00**	8.15	247.50	9.17
	12	RR10	307.00	11.37	**294.50**	10.91	367.00	13.59
Avg Delay	All Instances		114.41		**108.91**		152.36	
	Excluding the Unsolved Instances		114.41		**108.91**		152.36	
Avg Delay per Train	All Instances		4.24		**4.03**		5.64	
	Excluding the Unsolved Instances		4.24		**4.03**		5.64	
Avg Runtime (secs)			0.01		7.19		8.36	

TABLE B.8: Late Departure: Average RD wrt FCFS

Late Departure	# of Trains	Instance	FCFS	SB-ATC		SB-ATC-ReOpt	
			Delay	Delay	RD	Delay	RD
Timetable		RT01	6.00	0.50	91.67%	0.50	91.67%
5	3	RR01	18.50	13.00	29.73%	13.00	29.73%
	5	RR02	31.00	23.00	25.81%	23.00	25.81%
10	3	RR03	43.00	35.00	18.60%	35.00	18.60%
	5	RR04	57.00	59.50	−4.39%	92.00	−61.40%
15	3	RR05	55.00	47.00	14.55%	47.00	14.55%
	5	RR06	80.50	80.50	0.00%	108.00	−34.16%
20	9	RR07	185.50	182.50	1.62%	445.00	−139.89%
	12	RR08	244.00	242.50	0.61%	298.00	−22.13%
25	9	RR09	231.00	220.00	4.76%	247.50	−7.14%
	12	RR10	307.00	294.50	4.07%	367.00	−19.54%
	Avg RD	All Instances		17.00%		−9.45%	
		Instances Excl. Unsolved Soln's		17.00%		−9.45%	

Table B.9: Late and On-time Trains: Delay and Relative Delay

Late Departure	# of Trains	Instances	Delay						Relative Delay			
			FCFS			SB-ATC			FCFS		SB-ATC	
			Total	Late Trains	On-time Trains	Total	Late Trains	On-time Trains	Late Trains	On-time Trains	Late Trains	On-time Trains
Timetable		RT01	6.00	0.00	6.00	0.50	0.00	0.50	0.00%	100.00%	0.00%	100.00%
5	1	RR01	11.00	5.00	6.00	6.50	5.00	1.50	45.45%	54.55%	76.92%	23.08%
	3	RR02	18.50	15.00	3.50	13.00	13.00	0.00	81.08%	18.92%	100.00%	0.00%
	5	RR03	31.00	25.50	5.50	23.00	18.00	5.00	82.26%	17.74%	78.26%	21.74%
10	1	RR04	14.00	8.00	6.00	8.50	8.00	0.50	57.14%	42.86%	94.12%	5.88%
	3	RR05	43.00	31.00	12.00	35.00	29.00	6.00	72.09%	27.91%	82.86%	17.14%
	5	RR06	57.00	44.50	12.50	59.50	45.00	14.50	78.07%	21.93%	75.63%	24.37%
15	1	RR07	21.00	14.50	6.70	14.50	12.50	2.00	69.05%	31.90%	86.21%	13.79%
	3	RR08	55.00	44.50	10.50	47.00	42.50	4.50	80.91%	19.09%	90.43%	9.57%
	5	RR09	80.50	70.50	10.00	80.50	76.00	4.50	87.58%	12.42%	94.41%	5.59%
20	6	RR10	120.50	120.00	0.50	119.00	119.00	0.00	99.59%	0.41%	100.00%	0.00%
	9	RR11	185.50	179.50	6.00	182.50	180.00	2.50	96.77%	3.23%	98.63%	1.37%
	12	RR12	244.00	241.00	3.00	242.50	237.50	5.00	98.77%	1.23%	97.94%	2.06%
25	6	RR13	151.50	144.50	7.00	142.50	142.00	0.50	95.38%	4.62%	99.65%	0.35%
	9	RR14	231.00	224.50	6.50	220.00	217.00	3.00	97.19%	2.81%	98.64%	1.36%
	12	RR15	307.00	300.50	6.50	294.50	291.00	3.50	97.88%	2.12%	98.81%	1.19%
		Avg Delay	98.53	91.78	6.76	93.06	89.72	3.34	77.45%	22.61%	85.78%	14.22%
		Avg Delay per Train	3.65	3.40	0.25	0.45	3.32	0.12				

Bibliography

Aarts, E. H. L. and Lenstra, J. K., editors (1997). *Local Search and Combinatorial Optimization*. J. Wiley, New York.

Aarts, E. H. L., van Laarhoven, P. J. M., Lenstra, J. K., and Ulder, N. L. J. (1994). A computational study of local search algorithms for job shop scheduling. *ORSA Journal on Computing*, 6(2):118–125.

Adams, J., Balas, E., and Zawack, D. (1988). The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, 34(3):391–401.

Ahuja, R. K., Cunha, C. B., and Sahin, G. (2005). Network Models in Railroad Planning and Scheduling. *TutORials in Operations Research*, 1:54–101.

Albrecht, T. (2009). Automated Timetable Design for Demand-Oriented Service on Suburban Railways. *Public Transport*, 1(1):5–20.

Alvarez-Valdes, R., Fuertes, A., Tamarit, J. M., Gimenez, G., and Ramos, R. (2005). A heuristic to schedule flexible job-shop in a glass factory. *European Journal of Operational Research*, 165:525–534.

Applegate, D. and Cook, W. (1991). A Computational Study of the Job-shop Scheduling Problem. *ORSA Journal on Computing*, 3:149–156.

Assad, A. A. (1980). Modelling of Rail Networks: Toward a Routing/Makeup Model. *Transportation Research Part B: Methodological*, 14(1):101–114.

Balas, E., Lenstra, J. K., and Vazacopoulos, A. (1995). The One-Machine Problem with Delayed Precedence Constraints and Its Use in Job Shop Scheduling. *Management Science*, 41(1):94–109.

Balas, E., Simonetti, N., and Vazacopoulos, A. (2008). Job Shop Scheduling with Setup Times, Deadlines and Precedence Constraints. *Journal of Scheduling*, 11(4):253–262.

Balas, E. and Vazacopoulos, A. (1998). Guided Local Search with Shifting Bottleneck for Job Shop Scheduling. *Management Science*, 44(2):262–275.

Baptiste, P. (2000). Scheduling Equal-length Jobs on Identical Parallel Machines. *Discrete Applied Mathematics*, 103(1):21–32.

Barnhart, C., Jin, H., and Vance, P. H. (2000). Railroad Blocking: A Network Design Application. *Operations Research*, 48(4):603–614.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3):316–329.

Bellman, R. (2003). *Dynamic Programming.* Princeton University Press.

Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models.* Prentice Hall, New Jersey.

Borndorfer, R., Grtschel, M., Lukac, S., Mitusch, M., Schlechte, T., Schultz, S., and Tanner, A. (2005). An Auctioning Approach to Railway Slot Allocation. Technical report, 0545, Konrad-Zuse-Zentrum fr Informationstechnik Berlin.

Borndorfer, R. and Schlechte, T. (2007a). Models for Railway Track Allocation. Technical report, 0702, Konrad-Zuse-Zentrum fr Informationstechnik Berlin.

Borndorfer, R. and Schlechte, T. (2007b). Solving Railway Track Allocation Problems. Technical report, 0720, Konrad-Zuse-Zentrum fr Informationstechnik Berlin.

Brannlund, U., Lindberg, P. O., Nou, A., and Nilsson, J. E. (1998). Railway Timetabling using Lagrangian Relaxation. *Transportation Science*, 32(4):358–369.

Bratley, P., Florian, M., and Robillard, P. (1973). On Sequencing with Earliest Starts and Due Dates with Application to Computing Bounds for the (n—m—G—Fmax) Problem. *Naval Research Logistics Quarterly*, 20:57–67.

Brizuela, C. A., Zhao, Y., and Sannomiya, N. (2001). No-wait and blocking job-shops: Challenging problems for GA's. In *IEEE 0-7803-77-2/ 01*, pages 2349 – 2354.

Brooks, G. H. and White, C. R. (1965). An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem. *Journal of Industrial Engineering*, 16:34–40.

Brucker, P. (1988). An Efficient Algorithm for the Job-shop Problem with Two Jobs. *Computing*, 40:353–359.

Brucker, P. (2007). *Scheduling Algorithms.* Springer.

Brucker, P., Jurisch, B., and Kramer, A. (1994a). The Job-shop Problem and Immediate Selection. *Annals of Operations Research*, 50(1):73–114.

Brucker, P., Jurisch, B., and Sievers, B. (1994b). A Branch and Bound Algorithm for the Job-shop Scheduling Problem. *Discrete Applied Mathematics*, 49:107–127.

Brucker, P. and Knust, S. (1994). Complexity results for scheduling problems. Retrieved April 29, 2013, from http://www.informatik.uni-osnabrueck.de/knust/class/.

Brucker, P. and Kravchenko, S. A. (2005). Scheduling Jobs with Release Times on Parallel Machines to Minimize Total Tardiness. Technical report, Universität Osnabrück, Fachbereich Mathematik/Informatik.

Bulbul, K. (2011). A Hybrid Shifting Bottleneck-Tabu Search Heuristic for the Job Shop Total Weighted Tardiness Problem. *Computers & Operations Research*, 38(6):967–983.

Burke, E. and Kendall, G., editors (2005). *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques.* Springer.

Bussieck, M. R. (1997). *Optimal Lines in Public Rail Transport.* PhD thesis, TU Braunschweig.

Cacchiani, V., Caprara, A., and Toth, P. (2008). A Column Generation Approach to Train Timetabling on a Corridor. *4OR*, 6(2):125–142.

Cacchiani, V. and Toth, P. (2012). Nominal and Robust Train Timetabling Problems. *European Journal of Operational Research*, 219(3):727–737.

Cai, X. and Goh, C. J. (1994). A Fast Heuristic for the Train Scheduling Problem. *Computers and Operations Research*, 21:499510.

Cai, X., Goh, C. J., and Mees, A. I. (1998). Greedy Heuristics for Rapid Scheduling of Trains on a Single Track. *IIE Transaction*, 30:481493.

Caimi, G. (2009). *Algorithmic Decision Suport for Train Scheduling in a Large and Highly Utilised Railway Network*. PhD thesis, ETH Zurich.

Caimi, G., Burkolter, D., and Herrmann, T. (2005). *Operations Research Proceedings 2004*, chapter Finding Delay-tolerant Train Routings through Stations, pages 136–143. Springer Berlin Heidelberg.

Caimi, G., Burkolter, D., Herrmann, T., Chudak, F., and Laumanns, M. (2007). Design of a New Railway Scheduling Model for Dense Services. In *Proceedings of the 2nd International Seminar on Railway Operations Modelling and Analysis. Hannover, Germany.*

Camm, J. D., Raturi, A. S., and Tsubakitani, S. (1990). Cutting Big M Down to Size. *Interfaces*, 20(5):61–66.

Caprara, A., Fischetti, M., and Toth, P. (2002). Modeling and Solving the Train Timetabling Problem. *Operations Research*, 50(5):851–861.

Caprara, A., Kroon, L., Monaci, M., Peeters, M., and Toth, P. (2007). *Passenger Railway Optimization*, chapter 3, pages 129–187. Elsevier.

Carey, M. (1994a). A Model and Strategy for Train Pathing with Choice of Lines, Platforms, and Routes. *Transportation Research Part B: Methodological*, 28(5):333–353.

Carey, M. (1994b). Extending a Train Pathing Model from One-way to Two-way Track. *Transportation Research Part B: Methodological*, 28(5):395–400.

Carey, M. and Carville, S. (2003). Scheduling and Platforming Trains at Busy Complex Stations. *Transportation Research, Part A*, 37(3):194224.

Carey, M. and Crawford, I. (2007). Scheduling Trains on a Network of Busy Complex Stations. *Transportation Research, Part B*, 41(2):159178.

Carey, M. and Lockwood, D. (1995). A Model, Algorithms and Strategy for Train Pathing. *Journal of the Operational Research Society*, 46(8):988–1005.

Carlier, J. (1982). The One-machine Sequencing Problem. *European Journal of Operational Research*, 11:42–47.

Carlier, J. and Pinson, E. (1989). An Algorithm for Solving the Job-shop Problem. *Management Science*, 35(2):165–176.

Carlier, J. and Pinson, E. (1994). Adjustments of Heads and Tails for the Job-shop Problem. *European Journal of Operational Research*, 78:146–161.

Cartwright, H. M. and Cattell, J. R. (1995). *Applications of Modern Heuristic Methods*, chapter Studies of continuous-flow chemical synthesis using genetic algorithms, page 129144. Alfred Waller.

Charlton, J. M. and Death, C. C. (1970). A Method of Solution for General Machine-Scheduling Problems. *Operations Research*, 18:689–707.

Chen, H. and Luh, P. B. (2003). An alternative framework to Lagrangian relaxation approach for job shop scheduling. *European Journal of Operational Research*, 149(3):499–512.

Cook, S. A. (1971). The Complexity of Theorem Proving Procedures. In *STOC '71 Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158.

Cordeau, J. F., Toth, P., and Vigo, D. (1998). A Survey of Optimization Models for Train Routing and Scheduling. *Transportation science*, 32(4):380–404.

Corman, F., D'Ariano, A., Pacciarelli, D., and Pranzo, M. (2009). Evaluation of green wave policy in real-time railway traffic management. *Transportation Research Part C*, 17:607–616.

Corman, F., D'Ariano, A., Pacciarelli, D., and Pranzo, M. (2010). A Tabu Search Algorithm for Rerouting Trains During Rail Operations. *Transportation Research Part B-Methodological*, 44:175–192.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press and McGraw-Hill.

Crainic, T. G. (2003). *Handbook of Transportation Science*, chapter Long-haul Freight Transportation, pages 451–516. Springer US.

Crowder, H., Johnson, E. L., and Padberg, M. (1983). Solving Large-Scale Zero-One Linear Programming Problems. *Operations Research*, 31(5):803–834.

Dantzig, G. B. (1951). *Activity Analysis of Production and Allocation*, chapter Maximization of a linear function of variables subject to linear inequalities, pages 339–347. John Wiley & Sons Inc., New York, N. Y.

D'Ariano, A. (2008). *Improving Real-time Train Dispatching: Models, Algorithms and Applications*. PhD thesis, TUDelft.

D'Ariano, A., Corman, F., Pacciarelli, D., and Pranzo, M. (2008). Reordering and local rerouting strategies to manage train traffic in real-time. *Transportation Science*, 42(4):405–419.

D'Ariano, A., Pacclarelli, D., and Pranzo, M. (2007). A Branch and Bound Algorithm for Scheduling Trains in a Railway Network. *European Journal of Operational Research*, 183(2):643–657.

Dauzre-Prs, S. and Lasserre, J. B. (1993). A Modified Shifting Bottleneck Procedure for Job Shop Scheduling. *International Journal of Production Research*, 31:923–932.

Dauzre-Prs, S. and Lasserre, J. B. (1994). *An Integrated Approach in Production Planning and Scheduling*. Lecture Notes in Economics and Mathematical Systems, No. 411, Springer Verlag, Berlin.

Dauzre-Prs, S. and Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70:281–306.

Davis, L., editor (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

De Bontridder, K. (2005). Minimizing Total Weighted Tardiness in a Generalized Job Shop. *Journal of Scheduling*, 8(6):479–496.

Della Croce, F., Tadei, R., and Volta, G. (1995). A Genetic Algorithm for the Job Shop Problem. *Computers and Operations Research*, 22:15–24.

DellAmico, M. and Trubian, M. (1993). Applying tabu search to the jobshop scheduling problem. *Annals of Operations Research*, 41(1-4):231–252.

Denardo, E. V. (1982). *Dynamic Programming: Models and Applications.* Prentice-Hall, New Jersey.

Dorfman, M. J. and Medanic, J. (2004). Scheduling Trains on a Railway Network using a Discrete Event Model of Railway Traffic. *Transportation Research Part B: Methodological*, 38(1):81–98.

Dorndorf, U. and Pesch, E. (1995). Evolution Based Learning in a Job Shop Scheduling Environment. *Computers and Operations Research*, 22(1):25–40.

Dowsland, K. (1993). *Modern Heuristic Techniques for Combinatorial Problems*, chapter Simulated annealing. Blackwell Scientific Publications, Oxford.

Du, J. and Leung, J. Y.-T. (1990). Minimizing Total Tardiness on One Machine is NP-hard. *Mathematics of Operations Research*, 15(3):483–495.

Eglese, R. W. (1990). Simulated annealing: a tool for operational research. *European Journal of Operational Research*, 46:271–281.

Enquiries, N. R. (2013). National Rail Enquiries official website. Retrieved April 29, 2013, from http://www.nationalrail.co.uk/ .

Essafi, I., Mati, Y., and Dauzere-Peres, S. (2008). Agenetic Local Search Algorithm for Minimizing Total Weighted Tardiness in the Job-shop Scheduling Problem. *Computers and Operations Research*, 35(8):2599–2616.

Fattahi, P., Mehrabad, M. S., and Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18:331–342.

FICO (2013). FICO Xpress Optimimization Suite's official website. Retrieved February 01, 2012, from http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx.

Fisher, H. and Thompson, G. L. (1963). *Industrial Scheduling, Probabilistic Learning Combinations of Local Job-shop Scheduling Rules.* Prentice Hall.

Flamini, M. and Pacciarelli, D. (2008). Real time management of a metro rail terminus. *European Journal of Operational Research*, 189(3):746–761.

Fogarty, T. C., Ireson, N. S., and Bull, L. (1995). *Applications of Modern Heuristic Methods*, chapter Genetics based machine learning applications in industry and commerce, page 91110. Alfred Waller.

French, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop.* Ellis Horwood, Chichester, UK.

Gao, J., Sun, L., and Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35(9):28922907.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability, a Guide to the Theory of NP-completeness.* W. H. Freeman and Company.

Glover, F. (1989). Tabu Search - Part I. *ORSA Journal of Computing*, 1:190–206.

Glover, F. and Kochenberger, G. A., editors (2003). *Handbook of Metaheuristics.* Springer.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison- Wesley, Reading, MA.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278.

Gorman, M. F. (1998). An Application of Genetic and Tabu Searches to the Freight Railroad Operating Plan Problem. *Annals of Operations Research*, 78:51–69.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. H. G. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, 5:287–326.

Groeflin, H. and Klinkert, A. (2009). A New Neighborhood and Tabu Search for the Blocking Job Shop. *Discrete Applied Mathematics*, 157(17):3643–3655.

Hall, N. G. and Sriskandarajah, C. (1996). A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Operations Research*, 44(3):510–525.

Hansen, I. and Pachl, J. (2008). *Railway Timatbale & Traffic.* Eurail Press.

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

Hoos, H. and Stutzle, T. (2005). *Stochastic Local Search - Foundations and Applications.* Morgan Kaufmann Publishers, Elsevier, Amsterdam.

Huisman, D., Kroon, L. G., Lentink, R. M., and Vromans, M. J. (2005). Operations Research in Passenger Railway Transportation. *Statistica Neerlandica*, 59(4):467–497.

Ivens, P. and Lambrecht, M. (1996). Extending the Shifting Bottleneck procedure to real-life applications. *European Journal of Operational Research*, 90:252–268.

Jackson, J. R. (1955). Scheduling a Production Line to Minimize Maximum Tardiness. Technical report, Technical Report 43, University of California, Los Angeles.

Johnson, D. S., Papadimitriou, C. H., and Yannakakis, M. (1988). How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100.

Johnson, S. M. (1954). Optimal Two- and Three-stage Production Schedules with Setup Times Included. *Naval Research Logistics Quarterly*, 1:61–68.

Jovanovic, D. and Harker, P. T. (1991). Tactical Scheduling of Rail Operations: The SCAN I System. *Transportation Science*, 25(1):46–64.

Kachiyan, L. G. (1979). A polynomial time algorithm for linear programming. *Soviet Mathematics Dolady*, 20:191–194.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395.

Khosravi, B., Bennell, J. A., and Potts, C. N. (2010). Robust Train Scheduling based on the UK Network. In *24th European Conference on Operational Research EURO XXIV LISBON, Lisbon, Portugal.*

Khosravi, B., Bennell, J. A., and Potts, C. N. (2011a). A Modified Shifting Bottleneck Procedure for Train Scheduling in the UK. In *Workshop on Mathematical Optimization of Railway-Systems (CPAIOR2011), Berlin, Germany.*

Khosravi, B., Bennell, J. A., and Potts, C. N. (2011b). Train Scheduling in the UK using a Job Shop Scheduling Approach. In *Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2011), Phoenix, Arizona, USA.*

Khosravi, B., Bennell, J. A., and Potts, C. N. (2012a). Modified Shifting Bottleneck Algorithms for Train Scheduling and Rescheduling in the UK. In *Proceedings of the Vehicle Routing and Logistics Optimization (VeRoLog 2012), Bologna, Italy.*

Khosravi, B., Bennell, J. A., and Potts, C. N. (2012b). Train Scheduling and Rescheduling in the UK with a Modified Shifting Bottleneck Procedure. In *Proceedings of the 12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2012), Ljubljana, Slovenia.*

Kirkpatrick, S., Gellat, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220:671–680.

Klemenz, M. and Schultz, S. (2007). Modelling Aspects of a Railway Slot Allocation. In *2nd International Seminar on Railway Operations Modelling and Analysis.*

Kraay, D., Harker, P. T., and Chen, B. (1991). Optimal Pacing of Trains in Freight Railroads: Model Formulation and Solution. *Operations Research*, 39(1):82–99.

Kreipl, S. (2000). A Large Step Random Walk for Minimizing Total Weighted Tardiness in a Jobshop. *Journal of Scheduling*, 3(3):12538.

Kroon, L. G., Romeijn, H., and Zwaneweld, P. J. (1997). Routing Trains through Railway Networks: Complexity Issues. *European Journal of Operational Research*, 98:485–498.

Kubiak, W., Sethi, S., and Sriskandarajah, C. (1996). An efficient algorithm for a job shop problem. *Mathematical Industrial Systems*, 1:203–216.

Laguna, M. and Glover, F. (1993). Integrating Target Analysis and Tabu Search for Improved Scheduling Systems. *Expert Systems with Applications*, 6:287–297.

Land, A. H. and Doig, A. G. (1960). An Automatic Method for Solving Discrete Programming Problems. *Econometrica*, 28:97–520.

Lawler, E. L.; Wood, D. E. (1966). Branch-And-Bound Methods: A Survey. *Operations Research*, 14(4):699–719.

Lawler, E. L. (1977). A "Pseudopolynomial" Algorithm for Sequencing Jobs to Minimize Total Tardiness. *Annals of Discrete Mathematics*, 1:331–342.

Lee, Y. H. and Pinedo, M. (1997). Scheduling Jobs on Parallel Machines with Sequence-Dependent Setup Times. *European Journal of Operational Research*, 100(3):464–474.

Lenstra, J. K. (1977). *Sequencing by enumerative methods.* Mathematisch Centrum, Amsterdam.

Leung, J. Y.-T., editor (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis.* Chapman & Hall/CRC Press.

Leung, J. Y.-T. and Young, G. H. (1990). Minimizing Total Tardiness on a Single Machine with Precedence Constraints. *ORSA Journal on Computing*, 2(4):346–352.

Liebchen, C. and Mohring, R. H. (2008). *Computer-aided Systems in Public Transport*, chapter The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetablesand beyond, pages 117–150. Springer Berlin Heidelberg.

Liu, S. Q. and Kozan, E. (2009). Scheduling Trains as a Blocking Parallel-machine Job shop Scheduling Problem. *Computers & Operations Research*, 36(10):2840–2852.

Liu, S. Q. and Kozan, E. (2011). Scheduling Trains with Priorities: A No-Wait Blocking Parallel-Machine Job-Shop Scheduling Model. *Transportation Science*, 45(2):175–198.

Liu, S. Q. and Kozan, E. (2012). A Hybrid Shifting Bottleneck Procedure Algorithm for the Parallel-Machine Job-shop Scheduling Problem. *Journal of the Operational Research Society*, 63:168–182.

Liu, S. Q. and Ong, H. L. (2002). A Comparative Study of Algorithms for the Flowshop Scheduling Problem. *Asia - Pacific Journal of Operational Research*, 19(2):205–222.

Luethi, M. (2009). *Improving the Efficiency of Heavily used Railway Networks through Integrated Real-time Rescheduling.* PhD thesis, ETH Zurich.

Lusby, R., Larsen, J., Ehrgott, M., and Ryan, D. (2011). Railway Track Allocation: Models and Methods. *Or Spectrum*, 33(4):843–883.

Lusby, R., Larsen, J., Ryan, D., and Ehrgott, M. (2006). Routing Trains Through Railway Junctions: A New Set Packing Approach. Technical report, 21, Technical University of Denmark.

Mascis, A. and Pacciarelli, D. (2000). Machine scheduling via alternative graphs. Technical report, RT-DIA-46-2000, Italy.

Mascis, A. and Pacciarelli, D. (2002). Job-shop Scheduling with Blocking and No-wait Constraints. *European Journal of Operational Research*, 143(3):498–517.

Mason, S. J., Fowler, J. W., and Carlyle, W. M. (2002). A Modified Shifting Bottleneck Heuristic for Minimizing Total Weighted Tardiness in Complex Job Shops. *Journal of Scheduling*, 5(3):247–262.

Mason, S. J., Fowler, J. W., Carlyle, W. M., and Montgomery, D. C. (2005). Heuristics for minimizing total weighted tardiness in complex job shops. *International Journal of Production Research*, 43(10):1943–1963.

McMahon, G. B. and Florian, M. (1975). On scheduling with Ready Times and Due dates to Minimize Maximum Lateness. *Operations Research*, 23(3):475–482.

Meloni, C., Pacciarelli, D., and Pranzo, M. (2004). A Rollout Metaheuristic for Job Shop Scheduling Problems. *Annals of Operations Research*, 131(1):215–235.

Metropolis, N., Rosenbluth, A. W., Teller, A. H., and Teller, E. (1953). Equation of State Calculation by Fast Computing Machines. *Journal of Chemistry Physics*, 21:1087–1091.

Monch, L. and Drieel, R. (2005). A distributed shifting bottleneck heuristic for complex job shops. *Computers & Industrial Engineering*, 49:363–380.

Monch, L., Schabacker, R., Pabst, D., and Fowler, J. W. (2007). Genetic algorithm-based subproblem solution procedures for a modified shifting bottleneck heuristic for complex job shops. *European Journal of Operational Research*, 177:2100–2118.

Nemeti, L. (1964). Das Reihenfolgeproblem in der Fertigungsprogrammierung und Linear-planung mit logischen Bedingungen. *Mathematica (Cluj)*, 6:87–99.

NetworkRail (2010). Route Plans 2012, Route Plan A Kent, published on NetworkRail's official website. Retrieved April 29, 2013, from http://www.networkrail.co.uk/RoutePlans/PDF/RouteA-Kent.pdf.

NetworkRail (2013). Performance 2013, published on NetworkRail's official website. Retrieved April 29, 2013, from http://www.networkrail.co.uk/about/performance/.

Newton, H. N., Barnhart, C., and Vance, P. H. (1998). Constructing Railroad Blocking Plans to Minimize Handling Costs. *Transportation Science*, 32(4):330–345.

Nowicki, E. and Smutnicki, C. (1996). A fast tabu search algorithm for the job shop problem. *Management Science*, 42(6):797–813.

Odijk, M. A. (1996). A Constraint Generation Algorithm for the Construction of Periodic Railway Timetables. *Transportation Research Part B: Methodological*, 30(6):455–464.

Oliveira, E. (2001). *Solving Single-track Railway Scheduling Problem using Constraint Programming.* PhD thesis, The University of Leeds.

Oliveira, E. and Smith, B. M. (2000). A Job-shop Scheduling Model for the Single-track Railway Scheduling Problem. Technical report, Researcg Report Series - University of Leeds School of Computer Studies LU SCS RR (21).

Ovacik, I. M. and Uzsoy, R. (1997). *Decomposition Methods for Complex Factory Scheduling Problems.* Kluwer Academic Publishers, Boston.

Pachl, J. (2009). *Railway Operation and Control.* Mountlake Terrace: VTD Rail Publishing.

Papadimitriou, C. and Kanellakis, P. (1980). Flowshop scheduling with limited temporary storage. *Journal of Associated Computer Machinery*, 27:533–549.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity.* Upper Saddle River, NJ: Prentice-Hall, Inc.

Paschos, V. T., editor (2010a). *Concepts of Combinatorial Optimization.* Wiley-ISTE.

Paschos, V. T., editor (2010b). *Paradigms of Combinatorial Optimization.* Wiley-ISTE.

Peeters, L. (2003). *Cyclic Railway Timetable Optimization.* PhD thesis, Erasmus Research Institute of Management.

Pfund, M. E., Balasubramanian, H., Fowler, J. W., Mason, S. J., and Rose, O. (2008). A multi-criteria approach for scheduling semiconductor wafer fabrication facilities. *Journal of Scheduling*, 11(1):29–47.

Pinedo, M. and Singer, M. (1999). A Shifting Bottleneck Heuristic for Minimizing the Total Weighted Tardiness in a Job Shop. *Naval Research Logistics*, 46(1):1–17.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems.* New York, NY, Springer.

Pinson, E. (1995). *The Job Shop Scheduling Problem: A Concise Survey and Some Recent Developments*, chapter 3, Scheduling Theory and its Applications, pages 277–293. John Wiley & Sons Ltd.

Plastria, F. (2002). Formulating logical implications in combinatorial optimization. *European Journal of Operational Research*, 14(2):338–353.

Potts, C. N. and Strusevich, V. A. (2009). Fifty years of scheduling: a survey of milestones. *Journal of the Operational Research Society*, 60:41–68.

Rayward-Smith, V. J., Osman, I. H., Reeves, C. R., and Smith, G. D., editors (1996). *Modern Heuristic Search Methods.* John Wiley & Sons.

Reeves, C. R. (1995). *Applications of Modern Heuristic Methods*, chapter Genetic algorithms and combinatorial optimisation, page 111126. Alfred Waller.

Rodriguez, J. (2007). A Constraint Programming Model for Real-time Train Scheduling at Junctions. *Transportation Research Part B-Methodological*, 41(2):231–245.

Ross, S. M. (1983). *Introduction to Stochastic Dynamic Programming.* Academic Press, New York.

Roy, B. and Sussman, R. (1964). Les Problmes dordonnancement Avec Contraintes Disjonctives. Technical report, , 9, SEMA, Paris.

Sadeh, N., Sycara, K., and Xiong, Y. (1995). Backtracking techniques for the job shop scheduling constraint satisfaction problem. *Artificial Intelligence*, 76:455–480.

Sahin, I. (1999). Railway Traffic Control and Train Scheduling based on Inter-train Conflict Management. *Transportation Research Part B-Methodological*, 33(7):511–534.

Sauder, R. L. and Westerman, W. M. (1983). Computer Aided Train Dispatching: Decision Support through Optimization. *Interfaces*, 13(6):24–37.

Schlechte, T. (2012). *Railway Track Allocation: Models and Algorithms.* PhD thesis, Technical University of Berlin.

Schrage, L. (1970). Solving Resource-constrained Network Problems by Implicit Enumeration-Nonpreemptive Case. *Operations Research*, 18:253–278.

Schrijver, A. (1998). *Theory of Linear and Integer Programming.* Wiley, New York.

Schrijver, A. and Steenbeek, A. (1994). Timetable Construction for Railned. Technical report, C.W.I. Center for Mathematics and Computer Science, Amsterdam.

Schutten, M. (1998). Practical job shop scheduling. *Annals of Operational Research*, 83:161–177.

Serafini, P. and Ukovich, W. (1989). A Mathematical for Periodic Scheduling Problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581.

Smith, S. P. (1992). Experiment on using Genetic Algorithms to Learn Scheduling Heuristics. In Biswas, G., editor, *Proceedings of SPIE, Volume 1707, Applications of Artificial Intelligence X: Knowledge-Based Systems, The International Society for Optical Engineering: Bellingham, WA*, page 378386.

Szpigel, B. (1973). Optimal Train Scheduling on a Single Track Railway. In Ross, M., editor, *Operational Research 72, Amsterdam, The Netherlands*, pages 343–352.

Taha, H. A. (2002). *Operations Research: An Introduction.* Prentice Hall.

Taillard, E. (1994). Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *ORSA Journal on Computing*, 6(2):108–117.

Timkovsky, V. G. (1985). On the complexity of scheduling an arbitrary system. *Soviet Journal of Computer and System Sciences*, 5:46–52.

Tornquist, J. and Persson, J. A. (2005). Train Traffic Deviation Handling using Tabu Search and Simulated Annealing. In *Proceedings of the 38th Annual Hawaii International Conference on. IEEE, 2005.*

Tornquist, J. and Persson, J. A. (2007). N-tracked Railway Traffic Rescheduling During Disturbances. *Transportation Research Part B: Methodological*, 41(3):342–362.

Ullman, J. (1975). NP-complete Scheduling Problems. *Journal of Computer and System Science*, 10 (3):384–393.

van Laarhoven, P. and Aarts, E. (1987). *Simulated annealing: theory and applications*. Kluwer, Dordrecht.

Vepsalainen, A. and Morton, T. (1987). Priority Rules for Job Shops with Weighted Tardiness Cost. *Management Science*, 33(8):1035–1047.

Williams, H. P. (1990). *Model Building in Mathematical Programming*. Wiley, Chichester.

Wolsey, L. A. and Nemhauser, G. L. (1998). *Integer and Combinatorial Optimization*. Wiley, New York.

Wong, R. C., Yuen, T. W., Fung, K. W., and Leung, J. M. (2008). Optimizing Timetable Synchronization for Rail Mass Transit. *Transportation Science*, 42(1):57–69.

Xia, W. and Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computer & Industrail Engineering*, 48:409–425.

Yamada, T., Rosen, B. E., and R., N. (1994). A Simulated Annealing Approach to job Shop Scheduling using Critical Block Transition Operators. In *IEEE International Conference on Neural Networks, ICNN94. IEEE: New York*, pages 4687–4692.

Yan, H. and Hooker, J. N. (1999). Tight representation of logical constraints as cardinality rules. *Mathematical Programming*, 85(2):363–378.

Zhou, X. and Zhong, M. (2007). Single-track Train Timetabling with Guaranteed Optimality: Branch-and-bound Algorithms with Enhanced Lower Bounds. *Transportation Research Part B: Methodological*, 41(3):320–341.