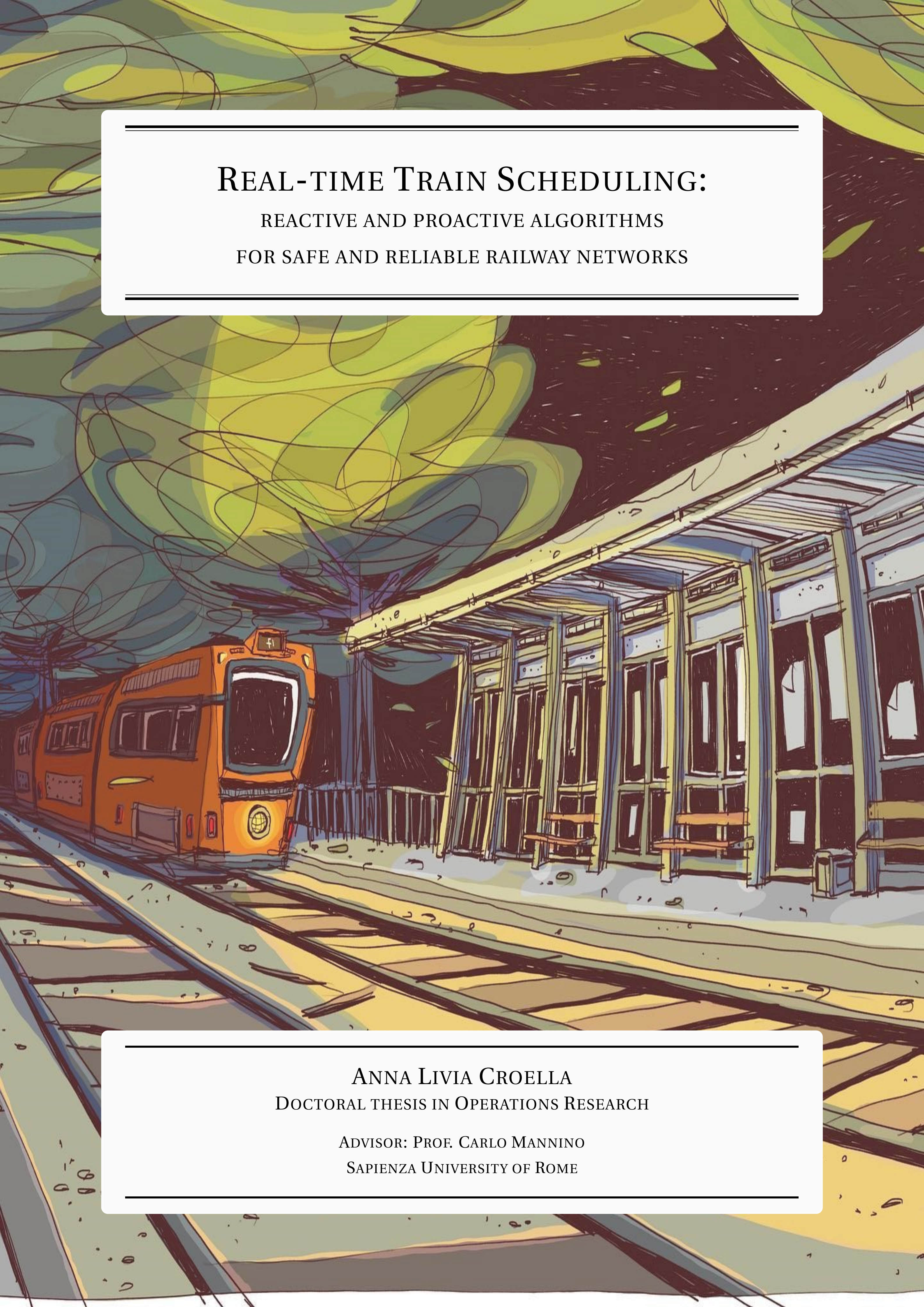


Real-time Train Scheduling:

reactive and proactive algorithms

for safe and reliable railway networks

Anna Livia Croella

Doctoral thesis in Operations Research

Advisor: Prof. Carlo Mannino

Sapienza University of Rome

SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Department of Computer, Control and Management Engineering

Doctoral thesis in Operations Research

Real-time Train Scheduling: reactive and proactive algorithms for safe and reliable railway networks

Thesis Advisor
Prof. Carlo Mannino

Co-Advisor
Ph.D. Paolo Ventura

Candidate
Anna Livia Croella
1544694

Academic Year MMXIIX-MMXXI (XXXIV cycle)

Real-time Train Scheduling: reactive and proactive algorithms for safe and reliable railway networks

Ph.D. thesis Sapienza University of Rome

This thesis has been typeset by LaTeX.

Author's email: annalivia.croella@uniroma1.it

La prima volta vai un po' a caso, e questa è cosa nota.
Vale praticamente per tutto.
L'amore, la pasta al forno, il katun a Mirabilandia.
Uno non sa come si fanno le cose finchè non le fa.

Stelle o sparo - Nova

Abstract

Train scheduling is a critical activity in rail traffic management, both off-line (timetabling) and on-line/in real-time (dispatching). This research aims at the design and development of advanced optimization models and methods for the real-time re-scheduling. The study covers two overlapping areas: managing the train service plans in an increasingly dynamic management scenario; and ensuring the safety and reliability of the railway system in case of major disruptions.

Although the *real-time re-scheduling problem* is relatively easy to depict, due to the interdependent nature of trains moving along the networks lines, a huge challenge looms in the search for an optimal schedule. A great variety of approaches have been proposed over the years, but here we are interested in those based on Mixed Integer Linear Programming, which are the most widely adopted in the literature. The main issue that such approaches must face is how to represent the fact that two trains cannot occupy simultaneously the same track (or other pairs of incompatible railway resources). Typically, Time-Indexed (TI) formulations for scheduling problems are stronger than other classical formulations, like big-M models. Moreover, they can be easily adapted to cope with complicating constraints and non-linear objectives. Unfortunately, their size grows usually very large with the size of the scheduling instance. As a consequence, TI formulations are not suitable for attacking real-life instances, especially when fast, on-line responses are required. Further, the approximation introduced by time discretization often leads to solutions which cannot be realized in practice.
The *Dynamic Discretization Discovery* (DDD), introduced by Boland for the continuous-time service network design problem, is a novel technique to keep at bay the growth of TI formulations (and thus their response times) and, at the same time, ensure the necessary modelling precision. Exploiting and extending the DDD paradigm, we develop a primal-dual exact approach to train dispatching. The result is a restricted TI formulation and a procedure with running times comparable with the best alternatives presented in the literature on our real-life instances of train dispatching. Furthermore, the method implemented does not suffer by the approximation error introduced by a standard time discretization. In our comparisons the big-M approach maintains the lead on average, but the distance is getting smaller. Even though the method developed in this research is at its early stage, it is very promising not only in a railway context but, more generally, for the broader field of job-shop scheduling. In addition, it offers many hints and opportunities for enhancements that will be investigated in future works.

On the other hand, when major disruptions occur in a rail network a simple rescheduling is not sufficient to re-ensure the viability of the network. Indeed, parts of the network may become unavailable for inbound trains, and decisions must be taken to mitigate the impact on the overall traffic. Arguably the most critical point is to avoid creating *deadlocks*, a situation that arises when a group of trains is positioned in such a way that none can move due to other trains in the group blocking their path. The infrastructure manager and train operating companies in these cases may be forced to stop trains until the normal status is recovered. A crucial aspect is thus to identify, for each train, a location where the train can hold during the disruption, avoiding to disconnect the network and allowing for a quick recovering of the original plan, at restart. A good location for holding a train is called a *safe place* and it additionally serves the purpose of preventing trains to drive past the last location where they can be safely parked, which could otherwise lead to further blockages and deadlocks. We outline some necessary and sufficient conditions to achieve the desired safe places properties. We then translate such conditions into constraints for a binary formulation of the problem, which we named *Safe Place Assignment Problem* (SPAP).
The SPAP finds an application in the current usage of Advanced Traffic Management Systems (ATMSs). This digital train management solutions are called to solve instances of a hard optimization problem in very limited computational time and they may fail to return a plan for different reasons. Finding a solution to the SPAP in these cases adds an additional level of safety, as dispatchers are provided with the last safe location for a train to drive within the plan's horizon. Computational results on a set of instances provided by a Class I U.S. railroad company show how the approach can be used effectively in the real-life setting that motivates the study, by returning optimal assignments in fractions of second.

Both research outputs are very innovative and off the beaten track. The achievements regards both the theoretical and methodological point of view and present in addition a practical relevance.

Keywords: *Real-Time Train Re-scheduling*, *Dynamic Discretization Discovery*, *Disruption Management*, *Safe Place Assignment*, *Binary Linear Programming*

Contents

List of Figures

List of Tables

Notation

Chapter 3

$\bar{p}$	Number of discretization interval periods
$\sigma(u)$	Subsequent operation of a generic elementary event u
τ	Schedule vector
$\tau(S)$	Schedule vector associated with the selection S
τ^*	Optimal schedule vector
$c(\tau)$	Cost function of a schedule vector
D	Set of disjunctive pairs of edges of a disjunctive graph
d	Disjunctive pair index
E	Set of fixed edges of a disjunctive graph
$G(N, E, D)$	Disjunctive graph
$G(S)$	Disjunctive graph associated with the selection S
I	Set of incompatible events
l	Length vector of the edges of a disjunctive graph
M	Very large number
N	Set of nodes of a disjunctive graph
n	End node of a disjunctive graph
o	Start node of a disjunctive graph
p	Discretization interval period index
S	Selection of directed edges on a disjunctive graph
$S(\tau)$	Selection of directed edges on a disjunctive graph associated with the schedule τ

S^*	Optimal selection of directed edges on a disjunctive graph
t	Continuous starting time variable vector for elementary events
u, v, q, m	Nodes of a disjunctive graph / Elementary events indices
x	Time-indexed variable vector for elementary events
y	Binary variable vector for alternative edge selection

Chapter 4

$\bar{w}$	Weight function vector associated with the interval Λ
Λ	Set of all partitions Λ^{ir}
Λ^{ir}	Partition set of the time intervals associated with the event "train i traverses track segment r"
λ_p^{ir}	p-th interval of the partition set Λ^{ir}
Λ_k	Set of all partitions of the $k-th$ iteration of the DDD-ALG
Φ	Function assigning to each interval λ_p^{ir} its lower bound h_p^{ir}
Ψ	Function assigning to each scheduled departure time t^{ir} a unique interval λ_p^{ir}
τ	Schedule vector
$\tau(S)$	Schedule vector associated with the set S
τ^*	Optimal schedule vector associated with the set S^*
τ_k	Schedule vector of the $k-th$ iteration of the DDD-ALG
$\underline{t}$	Earliest departure time vector
D_k	DDD problem solved at the $k-th$ iteration of the DDD-ALG
$G_k(V_k, E_k)$	DDD graph associated with the $k-th$ iteration of the DDD-ALG
h_p^{ir}	Lower bound of the interval λ_p^{ir}
I	Set of trains
l	Amount of time needed by train i to traverse track segment r
l_{rq}^{ij}	Amount of time that train j has to wait before using the track segment q, once train i traversed track segment r
M	Very large number
n_{ir}	Index of the last interval of the partition set Λ^{ir}

R	Set of all track segments
r	Track segment/block section index
R_i	Fixed route of train i
S	Set of distinct intervals $\subseteq \Lambda$
S^*	Λ-feasible set of intervals of minimum weight
S_k	Λ-feasible set of intervals of the DDD graph $G_k(V_k, E_k)$
t	Scheduled departure time vector
w	Weight function associated with the schedule τ
x	Binary variable vector for interval selection
$\mathcal{D}$	Set of couples of conflicting/non-shareable track segments traversed by trains, example: $\{(i,r),(j,q)\}$

Chapter 5

$\bar{\varsigma}$	Cost function for the halted trains
Δ	Node representing the end of a disruption in a $G(\phi)$ graph
κ	Maximum number of available safe place positions among all trains
ϕ	Safe place assignment function
ς	Cost function for the safe place assignment
b	Network junction or bifurcation point
$G(\phi)$	Disjunctive graph associated with a safe place assignment
h	Binary variable vector for halted trains
I	Set of trains
P	Blocking path/safe place
$P(i,s)$	Blocking path of train i when positioned with his head facing signal point s
P_i^0	Starting position of train i
Q	Bypassing blocking path
R	Set of all track segments
R_i	Fixed route of train i
s	Network signal point

U^P	Set of paths bypassing the blocking path/safe place P
y	Binary variable vector for the activation of bypassing paths
z	Binary variable vector for the assignment of trains to safe place positions
$\mathcal{P}(i)$	Set of blocking paths/(potential) safe places for train i
$\mathcal{U}$	Set of paths that can be used to bypass the trains parked in the safe places

Acronyms

AI	Artificial Intelligence
ATMS	Advanced Traffic Management System
ATP	Automatic Train Protection
CLP	Continuous Linear Program
DDD	Dynamic Discretization Discovery
DSS	Decision Support System
ERTMS	European Rail Traffic Management System
FSFS	First Scheduled First Served
ILP	Integer Linear Program
JSP	Job-shop Scheduling Problem
LP	Linear Program
LPP	Longest Path Problem
MILP	Mixed Integer Linear Program
MIP	Mixed Integer Program
MSSP	Maximal Stable Set Problem
MWSSP	Maximum Weight Stable Set Problem
OR	Operations Research
RTSP	Real-time Train re-Scheduling Problem
SNDP	Service Network Design Problem
SPAP	Safe Place Assignment Problem
SPP	Shortest Path Problem
TI	Time-Indexed
TMS	Traffic Management System

TSPTW	Travel Salesman Problem with time windows
TTRP	Train Traffic Re-scheduling Problem

Chapter 1

Introduction

1.1 Research motivation

Railway transports represent the choice of the majority of passenger and freight movements. In general, to increase the railway service quality means for many countries a big step in solving the mobility issue and therefore is part of most government priority lists. Railways are also the greenest mode of motorised transport in terms of greenhouse gas emissions. Rail uses the more fuel-efficient vehicles and causes less noise, accidents and infrastructure deterioration of car transport. Nowadays, with the environmental stewardship on the top of the community agenda, a smarter (more technological) use of the railways can make significant changes in reducing the human footprint while operating in a sustainable way.

One of the most challenging aspects in managing a railway systems regards the coordination of all its inherent processes, sectors and stakeholders. From the rolling stocks and the energy management to the control command signalling, all involved areas should cooperate in order to provide to the final customers a service that is safe and reliable. Since the deregulation of the European railway sector, the search for a balance that favours both sides of the industry has even become more difficult. On the one hand, the railway infrastructure managers want to achieve the higher utilization for the network and promote its maximum accessibility and availability, increasing trains speed and infrastructures capacities. Simultaneously, the other side sees the railway operating companies trying to fulfill the market demand in the best way possible and increasing their private market shares. Note that an increased capacity exploitation leads to a more sensitive system where even a small delay can seriously compromise the proper functioning of the whole railway traffic.
Of course, all the railway actors and organizations wish for the most efficient service that

guarantees the most reliable and secure movement of passengers and goods. This objective is pursued at all levels of service planning (strategic, tactical, operational) which involve different decision-making levels and span different time horizons. Nevertheless the service quality is often compromised by unexpected events that can be endogenous (infrastructure damages, maintenance, drivers behaviours, etc.) or exogenous (larger passenger boarding times, adverse weather conditions, accidents, animals on the tracks, etc.). In addition to an initial delay, these events can cause knock-on delays and/or even make trains to miss scheduled connections and/or create conflicts among them. A conflict arises when two or more trains require access to the same resource at the same time. We refer to an event that causes minor deviations from the scheduled times as *disturbance* and to an incident causing the interruption of the railway system as *disruption.* Usually, ideal time buffers and margins are inserted in railways timetable scheduled offline to absorb minor delays and dampen the knock-on (or domino) effect. However, in case of major disturbances and local/global infrastructure interruptions, such an expedient is never sufficient. Therefore an on-line monitoring and re-scheduling becomes essential to deal with unforeseen delays and ensure an adequate level of service and adherence to the original schedule. The *real-time train re-scheduling* aims at it. The goal is to restore a feasible situation for the network by resolving conflicts among trains, adjusting their times and routes in real-time while adhering as much as possible to the (off-line) scheduled plan.

The generational shift of the railway network to an automated digital control system is also part of the scenario. In the last decades a vast portion of the literature has focused on formulating models based on the availability of data on the fly. In addition, with the use of a single and unified European system for signalling and speed control, the European Rail Traffic Management System (ERTMS)[1], the papers concerning the European railway context may finally have a practical value. In general, a Traffic Management System (TMS) allows to long-term and short-term planning and to control the network by detecting potential conflicts and accordingly adjusting trains routes. There are of course benefits, both in terms of reduced operation cost and customer service improvement, and challenges with the introduction of these new automatic systems. Further, although TMSs already implement real-time control functions, the optimization models exploitation still has little room to breathe within them. This can be attributed both to the lack of real-time data from the field (and the possibility to share them) and the disregard of infrastructure managers. The latter are rather skeptical and/or not fully aware of the power of an optimization-based

[1] The ERTMS promotes the interoperability of different national railway systems; it incorporates Automatic Train Protection (ATP) systems and enables for a faster data communication ([63]).

approach, but also disillusioned by various failures and unfulfilled promises from models and algorithms. While the existing Decision Support Systems (DSSs) implement simple dispatching rules (like the *first come first served* rule), now researcher are able to generate very large and tailored solutions for all trains paths in a few minutes thanks to optimization algorithms. Of course, dispatchers can always decide to follow the solutions proposed or not. In the first case, we highlight that the most recent automated traffic control systems are able to proceed according to the plan provided since they are directly connected with the fields. On the other hand, when dispatchers choose to implement their own solutions the algorithm should adapt in order to be ready for optimization in its successive calls. Note that, when monitoring an extended and dense area of the network, it is difficult for traffic controllers (and in general for a human mind) to fully understand and calculate all consequences brought by a local decision. The use of enhanced algorithms can be in this case the only way to cross the barrier. We believe that in the future the infrastructure managers will be directly involved in the development of optimal real-time control systems, funding research groups or even taking part of them (as in the case of the cutting-edge multinational Alstom [2] and the large company Bombardier Transportation [106]). This collaboration will bring advantages to both the service, improving punctuality and coordination within lines, and the actors involved, for example reducing the dispatchers workload. An interested reader can refer to [24] for more details.
Among other things, in the next years we can also expect an increasing number of disruptions due to system failures as well as climate changes and heat-related damages. Since disturbances and disruptions can severely alter the circulation of the train traffic, in order to increase the reliability of the system, future control systems are required to implement new protocols to ensure efficient recovery. Most retrieval practices still follow predetermined procedures, contingency plans, or dispatcher instincts, and the literature lacks mathematical modelling for innovative protocols and procedures.

1.2 Research questions and goals

When trains move through a railway network, they occupy a sequence of rail resources, such as track sections, platforms, stations, etc.. Roughly speaking, train scheduling is about determining the time in which each train enters and exits the resources encountered on its path. Such a task is central in various phases of traffic planning. In the operational phase, when trains finally move through the railway, the original schedule must be adjusted in real-time to factor erratic train delays, small or large network disruptions, missing crews and other unplanned events. The work carried out in this research project focuses on this

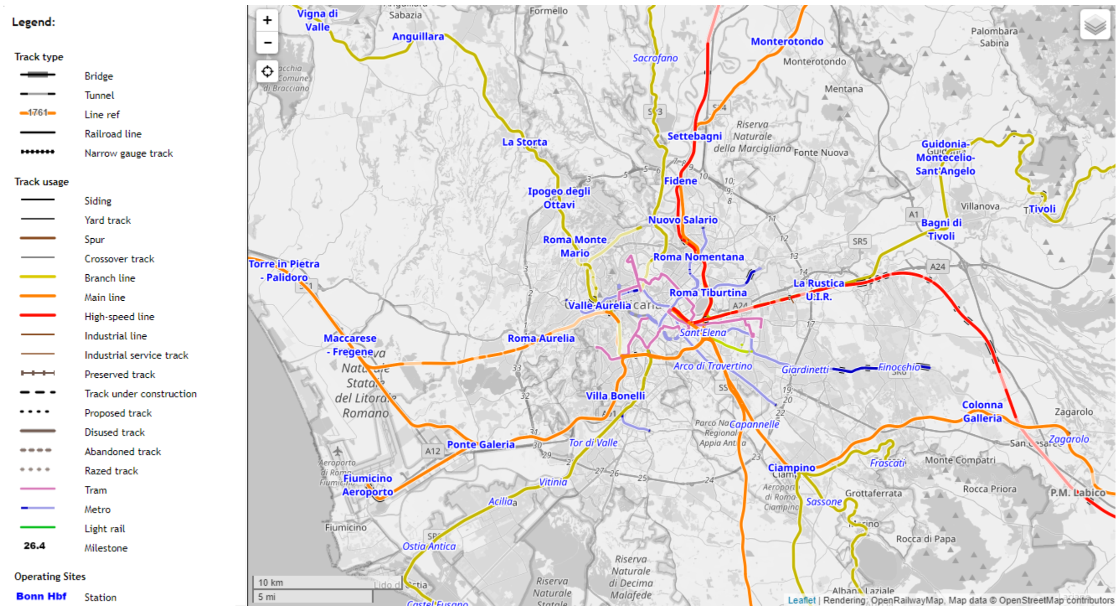


Figure 1.1: A map section showing the railway network area surrounding the city of Rome, including track types, station names, track numbers and milestones.

Source: *https://www.openrailwaymap.org*

on-line activity.
The research deals with the control of secure and reliable networks in an increasingly dynamic management scenario. The models developed tackle two different, but in a sense complementary, aspects of the same problem: managing the system after an unforeseen event that makes the planned activities no more feasible, and doing it as quickly and safely as possible. These requirements directly derive from the need for a more rigorous coordination between all the different railway players and, in general, generates from the possibility of adopting automated prototypes protocols in the real-life implementations for a new system-changing railway framework. We can formalize them into two main research objectives answering two different questions:

- in case of disturbance, how can we generate a new feasible plan quickly?
- in case of disruption (or partial re-scheduling information), how can we add an additional level of safety to the system?

Research Objective 1. *Propose a new efficient method for the Real-time Train re-Scheduling Problem (RTSP) that is capable to overcome the classical approaches drawbacks and pitfalls. To push the limits on the achievable computational times of the re-scheduling phase.*

From the optimization theory standpoint, train scheduling is a generalization of the job-shop scheduling (with blocking and no-wait constraints - [113]) and thus of the machine scheduling, and as such it can benefit from a vast body of literature. We search for a more efficient dynamic model that can quickly react to the occurrence of some delays and disturbances. We investigate the exploitation of an innovative technique that has been recently introduced in the context of the network design problem and that leaves for a wide application area. In the present work, we start to design its effective and efficient implementation and lay the foundation for its application in the train scheduling field. As one of the most delicate phases of the deployment, the algorithm engineering is deeply investigated focusing on both feasibility and optimality (while most of the works give a predominant rule to the first aspect). The performance of our proposed procedure is thus compared with the benchmark approaches of the case. In particular, we are interested in a comparison with the models based on Mixed Integer Linear Programming, that are the most adopted in the literature for the real-time train re-scheduling (and, in particular, for the train traffic re-scheduling).

Research Objective 2. *Outline an intelligent procedure to achieve a higher system safety in case of a disruption occurrence. To ease its application into real-life protocols through an efficient mathematical formulation of the model.*

Existing decision support systems cannot cope with huge disruptions, thus, in the majority of the cases, when a disruption occurs a list of predetermined rules is applied. However, very little has been done for their real-life implementations. In this research project we reach the formal definition of some common key objectives to mitigate the impact of unexpected events and make the network more secure. We formulate an optimization model capable of assigning trains to locations where they can be safely parked (hereafter simply referred as *safe place*). The purpose of this second research goal is itself twofold, showing a *proactive* aspect when used in combination with a movement planner (that estimate the future state of the net) and a *reactive* one when used locally after a disruption event.

1.3 Research approach and contributions

The research approach is model-based. We first identify the train re-scheduling framework and design two different problems. Thereafter we develop mathematical models to deal with them. In particular, our focus is on pure binary optimization programs and their formulation. We devise two linear models that are ideally suited to real-world applications and requirements. Going into more detail, we can identify a common structure in the formulations of the two designed models. Indeed, both present families of packing constraints. While the first model can be completely conducted to the integer programming formulation of the minimum weight stable set problem, the second formulation presented shares with it all but one of the constraints groups. In general, these type of problems are hard to solve, but they have been much studied in the literature and most commercial solvers can exploit the special structure of their constraint matrix.

Research Contribution 1. When modelling a railway system, each train service can be seen as a set of arrivals and departure events. In a re-scheduling problem, given the current status of the network, we want to determine new times for these event to take place since the original one are not compatible anymore and, in doing so, we aim to minimize some train-related delay measures. Although the (real-time re-scheduling) problem structure is relatively easy to depict, due to the interdependent nature of trains moving along the networks lines, a huge challenge looms in the search for an optimal schedule. Exploiting and extending a newly introduced technique, called Dynamic Discretization Discovery (DDD), we develop a primal-dual approach to tackle this challenging aspect of the train dispatching. The result of the study is a restricted time-indexed formulation and a procedure with running times comparable with the best alternatives presented in the literature on our real-life instances. Furthermore, the newly introduced method does not suffer from the approximation error introduced by a standard time discretization. In our comparisons the big-M approach, the most used in practise, retains the advantage on average, but the distance is getting smaller. Even though the method developed in the research is at its early stage, it is very promising not only in a railway context but, more generally, for the broader field of job-shop scheduling. In addition, it offers many hints and opportunities for enhancements that will be investigated in future works.

Research Contribution 2. When a severe disruption occurs, to stress the optimality and feasibility of (re)scheduling outputs, great attention must be paid to the safety and reliability of the entire railway system. In these cases a primary measure to mitigate its impact is a timely/anticipatory control of trains moving towards the disrupted area. A major contribution of this thesis is the formalization of a new mathematical model for a *train stop deployment protocol*, an issue that very few previous papers have pointed out in the literature (only three for the best of our knowledge). This type of protocols define how to move or where to stop the trains around the affected region during a disruption, with some (fairly basic) key objectives. Firstly, to leave a viable path in the network with no obstructions, so as to allow possible work/maintenance trains to reach the disruption site(s). Secondly, to quickly return to the normal traffic regime once the disruption is over. To this end, an heuristic idea suggests to push trains as far as possible along their original routes before stopping them in a safe place. We define the necessary and sufficient conditions for a *safe place assignment* to have the desired property and end up with a pure binary integer linear formulation that present a remarkably low resolution time. The practical and theoretical complexity of the underlying problem, named *Safe Place Assignment Problem*, is hence discussed. We remark that such a procedure can also be exploited when an ATMS fails to solve, in a short time, a particularly difficult re-scheduling problem. Moreover, even though we trace a specific real-life protocol adopted by a large Class I North-American railroad company, the model formulated is highly flexible and can be adapted to the different needs of railway operators and/or infrastructure managers.

Both research outputs are very innovative and off the beaten track. The achievements regards both the theoretical and methodological point of view (DDD method) and present in addition a practical relevance (Safe place assignment problem).

1.4 Outline

The thesis is structured as follows.

Chapter 2 presents the underlying theoretical framework and the mathematical foundations necessary to interpret and understand the results discussed in the following chapters. We provide the basic definitions and results of the optimization and the fundamental elements of complexity and graph theories.

In Chapter 3 the RTSP is presented in detail. We give the state-of-the-art of the scientific literature on the subject and report some background information on railway systems and its mathematical modelling. The concept of disjunctive graph and the mathematical rendering of the conflicts constraints among shared resources is then addressed. A discussion on the formulations mainly used in the field, with their strengths and pitfalls, closes the chapter.

The next two chapters generate from articles already submitted for publication and in conference proceedings. They contain the main findings of the related research projects.
Chapter 4 focuses on a particular facet of the RTSP, the Train Traffic Re-scheduling Problem. In this chapter we exploit a recently introduced DDD technique to implement an

algorithm that can overcome the drawbacks of the classical formulations (identified in the previous chapter). We give an overview of the related works on the DDD paradigm and then formally define our problem and the resolution steps engineering. The application of the algorithm on a set of real-life instances and the resulting performance is thus analyzed and compared with the benchmark approaches results.

Chapter 5 discusses a different aspect of train dispatching: the disruption management. Following a review of research on the topic, we outline a real-life protocol that aims to make the railway network more reliable and safe. We report the modelling choices that allow us to microscopically represent the railroad system and formally state the objectives pursued by the protocol. We thus derive a (binary linear) programming formulation for it. A computational study on two data set of realistic instances, highlighting different features of the formulated model, completes the work and shows the time efficiency of the program.

Finally, Chapter 6 summarizes the achievements and provides further ideas for extending the research study. New possible solution procedures for the train traffic re-scheduling algorithm are exposed, as well as new insights for tailor-made protocols in the field of disturbance management. Additional implications and recommendations for future work conclude the thesis.

Chapter 2

Preliminaries

2.1 Introduction

This chapter is devoted to the exposure of the theoretical framework necessary to understand the research contributions shown in the present thesis.

Section 2.2 gives the basic definitions and relevant results about optimization problems and, in particular, about linear programming. In Section 2.2.1 we introduce some basic notions of complexity theory to distinguish between problems that are easily solvable and problems that are classified as "hard" to handle. Then, we discuss two algorithms that are at the foundation of *integer linear programming*. Two general techniques, *row* and *column generation methods*, that can be applied to deal with large problems are hence introduced in Section 2.2.4. Finally, Section 2.3 exposes some relevant elements of graph theory and formally define two important problems that we will address in the rest of the thesis and that are also widely used as subroutines in many resolution algorithms, namely the *shortest path problem* and the *maximum stable set problem*. We refer the interested reader to [34–36, 120, 141] for comprehensive overviews on the discussed topics.

2.2 Optimization problems

An optimization problem is a generic problem in which one has to choose among a set of distinct and non-mixable alternatives, denoted by $\mathcal{S}$ and called *feasible region*. The choice is made according to a quantitative criterion, namely according to a given *objective function* $f(\cdot) : \mathcal{S} \rightarrow R$ that has to be maximized or minimized. Without loss of generality, we can write a generic mathematical program as $min\{f(x) : x \in \mathcal{S}\}$, where x is the vector of the

variables to be optimized.
Three distinct scenarios may arise:

- the problem is *infeasible*, i.e. $\mathcal{S} = \emptyset$ and no solution exists;

- the problem is *unbounded*, that is given a value M there always exists a vector $\bar{x}$ such that $f(\bar{x}) < -M$;

- the problem admits an *optimal solution* and there exists a vector $x^* \in \mathcal{S}$ such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{S}$; we refer to the value $f(x^*)$ as *optimum.*

Real world optimization problems admit a finite number (however large it may be) of solutions, i.e. the set $\mathcal{S}$ has a finite cardinality. The field of the applied mathematics addressing optimization problem over a discrete set is called *combinatorial optimization.*
In general. the characteristics of the set $\mathcal{S}$ and the objective function properties determine the computational complexity of the problem. In this thesis we will focus only on the so called *linear problems*, that are problems showing a linear objective function and whose feasible region can be expressed by a finite set of linear constraints. Among these we can distinguish between pure Continuous Linear Programs (CLPs), Integer Linear Programs (ILPs) or Mixed Integer Linear Programs (MILPs). Formally, a generic linear program can be expressed in the form:

$$\begin{aligned} \min \quad & cx \\ s.t. \quad & Ax \leq b \\ & x \geq 0 \quad \text{integral or real} \end{aligned} \tag{2.1}$$

with A being a $m \times n$ matrix of (usually rational) elements, c a row vector in $\mathbb{R}^n$ and b a column vector in $\mathbb{R}^m$. We define a set of the type $\mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b\}$, i.e. a set of points satisfying a finite number of linear inequalities, as a *polyhedron.* A bounded polyhedron is also called a *polytope.*
When talking about a CLP problem the variable vector x can only assume real values, $x \in \mathbb{R}^n$, while for an ILP the domain is the set of integer numbers and $x \in \mathbb{Z}^n$. One can also identify a subclass of problems for the case of integer programming: a *pure binary problems* is denoted by the presence of only binary variables, that is $x \in \{0,1\}^n \subseteq \mathbb{Z}^n$. On the other hand, for a MILP we can identify two distinct variables vectors with their own domain ($x \in \mathbb{Z}^{n_x}$ and $y \in \mathbb{R}^{n_y}$) and separate two row vectors for the objective function (c and h) and two constraint matrices (A and G) respectively. A MILP problem can be thus

formulated as follows:

$$\begin{aligned} \min \quad & cx + hy \\ s.t. \quad & Ax + Gy \le b \\ & x \ge 0 \quad \text{integral} \\ & y \ge 0 \quad \text{real} \ . \end{aligned} \tag{2.2}$$

Figure 2.1 shows the representation in $\mathbb{R}^2$ of a pure integer set (2.1b), a continuous set (2.1a) and a mixed integer linear set (2.1c).

2.2.1 Elements of complexity theory

An optimization problem can be always formulated as a *decision* problem that allows only two possible solution: "yes" or "no". Indeed, searching for a solution with certain properties is equivalent to ask if a solution having that properties exists. Once formulated a decision problem, one of the crucial question is how much time we need to compute a solution for it. Basically, we want to answer this question for different sets of inputs, namely *instances*, that we may provide to the model. Of course, different instances may be less or more complex to solve. Moreover, the time needed to solve them strictly depends by the algorithm used. Given a problem, an *algorithm* is a procedure that is able to find in a finite number of steps the solution to a generic instance of it. In general, we can identify the dimension (or *size*) $\phi(x)$ of a given instance x with the space needed to store the input data. We hence define the *algorithm complexity* $f(\phi)$ as the number of steps (or arithmetic operation) it requires to find the solution associated with an instance. Now, we can determine a bound for the algorithm complexity solving the *worst case* instances of a problem and, in most of the cases, it is convenient to use an upper approximation, that is easy to calculate, to bound the size of the instance. We recall that, given two functions $f(x)$ and $g(x)$, we say that $f(x) = \mathcal{O}(g(x))$, if it exists a value x_0 such that $f(x) \le g(x)$ for all $x > x_0$, that is after a certain x_0 the function $f(x)$ definitely lies under $g(x)$. Next we give two fundamental notion of the complexity theory.

Definition 2.1. *An algorithm has a polynomial complexity if and only if its complexity $f(\phi)$ in the worst case instances is such that $f = \mathcal{O}(g(\phi))$, with $g(\phi)$ being a polynomial. We call it a polynomial algorithm.*

Definition 2.2. *A decision problem belongs to the complexity class $\mathcal{P}$ if and only if there exists a polynomial algorithm to solve it.*

Obviously, when it comes to polynomial algorithms the degree of the polynomial is a measure of its efficiency: the lower the degree the better the procedure. We can also have

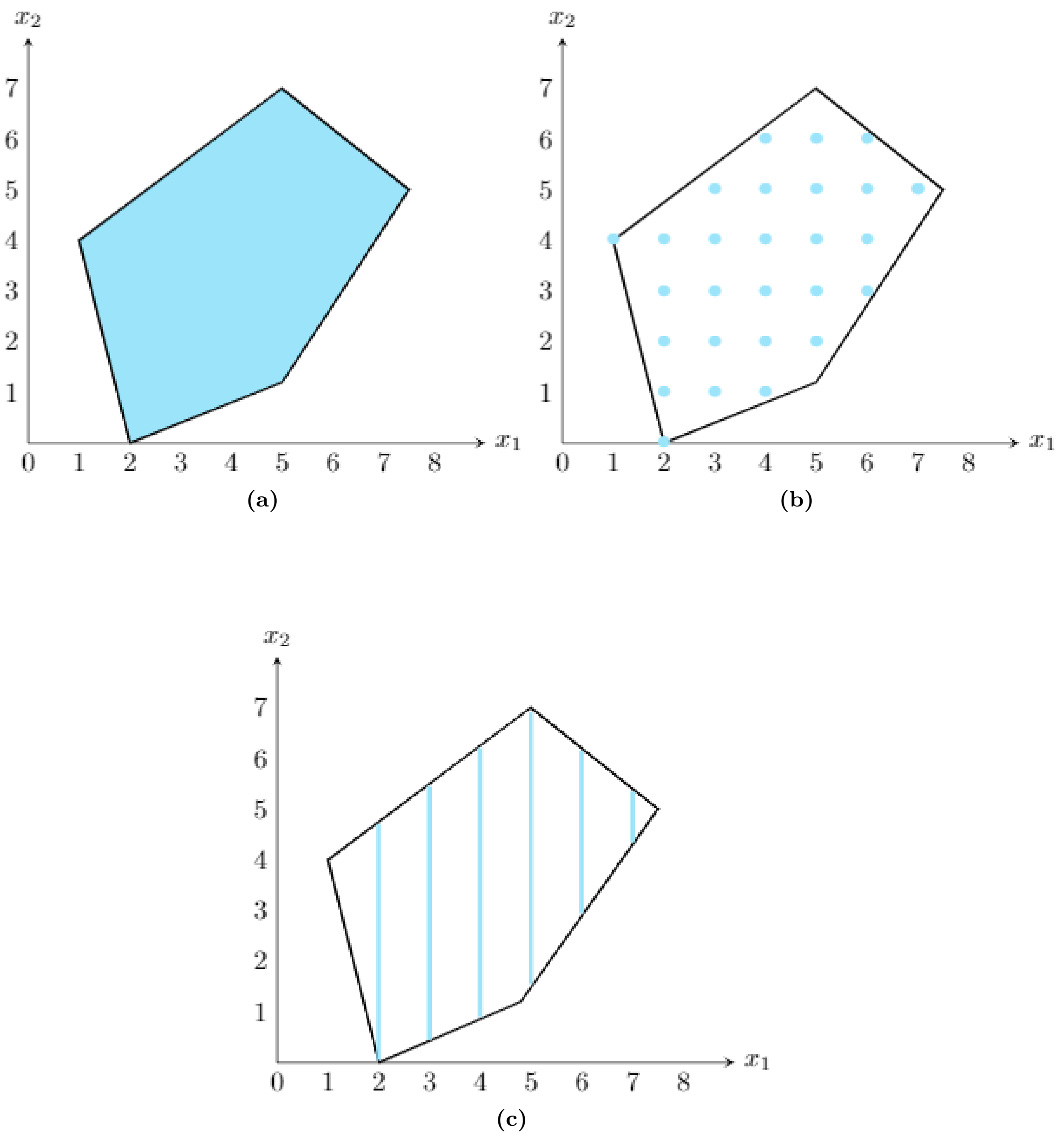


Figure 2.1: Example of a continuous (2.1b), a pure integer (2.1a) and a mixed integer (2.1c) linear set defined by a finite number of linear inequalities in $\mathbb{R}^2$.

different trends for complexity, such as exponential or hyper-exponential. The latter, from a certain value on, grows faster than a polynomial and are therefore considered inefficient.

Another important class of problems is the *non-deterministic polynomial-time class*, or $\mathcal{NP}$ class.

Definition 2.3. *A decision problem belong to the complexity class $\mathcal{NP}$ if there exists a polynomial algorithm able to prove the validity of all the instances with a "yes"-certificate.*

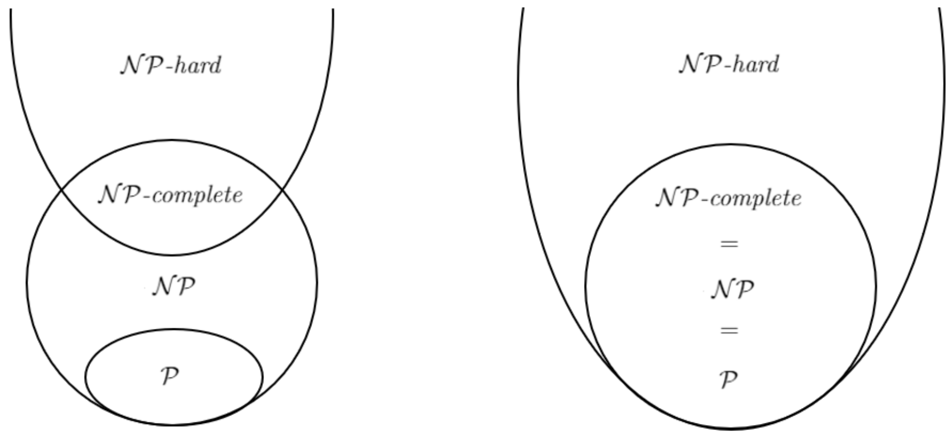
Roughly speaking, given a problem, each instance with a positive answer can be certified being so in polynomial time. This implies that the "yes"-certificate itself shows a size bounded by a polynomial. On the other hand, for a "no"-answer instance we are not able to certify whether the solution is correct or not in polynomial time. Mirror-like, to the class co-$\mathcal{NP}$ belong all decision problems such that their "no"-certificate can be proved polynomially while we cannot claim the same for its "yes"-answers instances.
Clearly, class $\mathcal{NP}$ includes all problem in $\mathcal{P}$ ($\mathcal{P} \subseteq \mathcal{NP}$), but we still do not know if the two classes coincide ($\mathcal{P} = \mathcal{NP}$). Besides, we have that $\mathcal{P} \subseteq$ co-$\mathcal{NP}$ and therefore $\mathcal{P} \subseteq \mathcal{NP} \cap$ co-$\mathcal{NP}$. In particular, the problems belonging to $\mathcal{NP} \cap$ co-$\mathcal{NP}$ are those having a certification for both the positive and negative answer case.
The $\mathcal{NP}$ class also includes $\mathcal{NP}$*-complete* problems, known to be among the hardest problems.

Definition 2.4. *A decision problem $\Pi \in \mathcal{NP}$ is said to be $\mathcal{NP}$-complete if every problem in $\mathcal{NP}$ can be reduced to Π in polynomial time.*

Analogously, we have that Π is co-$\mathcal{NP}$*-complete* if all other decision problems in co-$\mathcal{NP}$ can be reduced to Π in polynomial time. Bear in mind that, when reducing a problem Π_1 in a different one Π_2, each instance with a "yes" answer in Π_1 must generate a "yes" answer in Π_2. Further, if we consider a generic problem (not necessarily in $\mathcal{NP}$) we can define the following subset of problems:

Definition 2.5. *A problem Π is said to be $\mathcal{NP}$-hard if all other decision problems in $\mathcal{NP}$ can be reduced to Π in polynomial time.*

Hence we call $\mathcal{NP}$*-hard* the problems whose complexity is considered at least as hard of any other problem in $\mathcal{NP}$. The basic example of an $\mathcal{NP}$*-hard* problem is integer programming. We remark that proving that a $\mathcal{NP}$*-complete* problem can be solved by a polynomial algorithm implies that all problems in $\mathcal{NP}$ are solvable in polynomial time, thus classes $\mathcal{P}$ and $\mathcal{NP}$ are essentially the same class. Figure 2.2 shows the Euler diagrams for the above



(a) $\mathcal{P} \neq \mathcal{NP}$ assumption. **(b)** $\mathcal{P} = \mathcal{NP}$ assumption.

Figure 2.2: Euler diagram for the complexity classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-*complete* and $\mathcal{NP}$-*hard*.

mentioned complexity classes assuming that $\mathcal{P} \neq \mathcal{NP}$ (Figure 2.2a) and $\mathcal{P} = \mathcal{NP}$ (Figure 2.2b).

2.2.2 Branch&Bound algorithms

The resolution of an linear program can be a non trivial task. In particular, solving an ILP can be an hard mission since integer programming belongs to the $\mathcal{NP}$-*hard* class. In general, the most used approach is to exploit a relaxation of the original problem, hence a set $\mathcal{P} \supset \mathcal{S}$, that is able to provide a good approximate solution easily. Such a solution gives a *lower bound* (LB) (*lower bound* (UB)) for a minimization (maximization) problem since it provides a lower (upper) limit for the optimal value reachable by the original integer problem. By definition, the set $\mathcal{P} \supset \mathcal{S}$ contains all points of $\mathcal{S}$ and it can hence provide a solution at least (at most) equal to the one of the optimum value $x^* \in \mathcal{S}$. Moreover, a LB (UB) allows to certify the quality of an integer solution, the shorter the distance within the LB (UB) and a solution value, the closer the optimal integer value.

Now, given a generic MILP, a *linear relaxation* of $\mathcal{S}$ can be defined by a polyhedron of the type $\mathcal{P} = \{(x, y) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \; : \; Dx + Qy \leq u\} \supseteq \mathcal{S}$. When $D = A$, $Q = G$ and $u = b$ we call such polyhedron a *natural linear relaxation* of $\mathcal{S}$. Note that $\mathcal{P}$ is simply obtained by changing the domain of the x variables, from natural to real number. The corresponding natural Linear Program (LP) relaxation is hence formulated as $min\{(x, y) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \; : \; Ax + By \leq b\}$.

one of the most adopted approach using this basic concept is the *Branch&Bound algorithm* (B&B). The algorithm proposes the exploration of the feasible region through an implicit enumeration and the bounding of the objective value. In fact, the explicit enumeration of all eligible solutions is in most of the cases computationally expensive and basically an impracticable way. Given sufficient time, the B&B guarantees to find, if it exists, the optimum of a ILP model and is hence classified as an *exact algorithm*.
Now, said $\Pi = min\{cx : x \in \mathcal{S}\}$, with $\mathcal{S} = \{x \in \mathbb{Z}^n : Ax \leq b\}$, and being $\mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b\}$ its relaxed polyhedron, we can rewrite Π as $min\{cx : x \in \mathcal{P} \cap \mathbb{Z}^n\}$ and define its natural relaxation as $min\{cx : x \in \mathcal{P}\}$. The B&B proceeds partitioning the feasible region into smaller subsets P_i and inserting them in a list $\mathcal{L}$. By definition, all subsets must satisfy the following properties:

- $\bigcup_i P_i = \mathcal{P}$, i.e. the union of all subsets is the polyhedron $\mathcal{P}$;
- $P_i \cap P_j = \emptyset$ for all pair-wise subsets with $i \neq j$, i.e. the intersection of any pair of different subsets must be null.

Then, taken a subset P_i in the list $\mathcal{L}$ of all partitions, the algorithm solves the associated problem $min\{cx : x \in P_i\}$ and finds an optimal solution x^i. We say that $z^i = cx^i$ is its relative optimum value. In particular, said z^* the optimum value obtained so far, when selecting (and removing) a subset P_i from the list $\mathcal{L}$ to solve it, the following two situations can occur:

- $z^i = cx^i < z^*$, then the algorithm checks if the solution is integer. If $x^i \in \mathbb{Z}^n$ the current value z^* is updated and set equal to cx^i, otherwise the current subset is further partitioned into subsets that are easier to analyze and that will be added to the list $\mathcal{L}$. these new subsets must contain all the integer solutions of P_i but x^i;
- $z^i = cx^i \geq z^*$ and, since the considered subset cannot generate a solution that is better than the current one, we can simply discard t.

We remark that if $P_i = \emptyset$ then the subset does not need to be considered any further and in this case we say that is has been *pruned*. The algorithm terminates when all the sets in the list $\mathcal{L}$ have been analyzed and an optimal solution of value z^* is returned. The evolution of the algorithm can be represented with a tree, called *branching* (or *enumeration*) *tree*. A flowchart resuming the B&B steps for a minimization ILP is presented in Figure 2.3.

As the algorithm name suggests, the two main components of the algorithms are the *branching* technique, which allows to divide the problem in simpler sub-problems, and the *bounding*

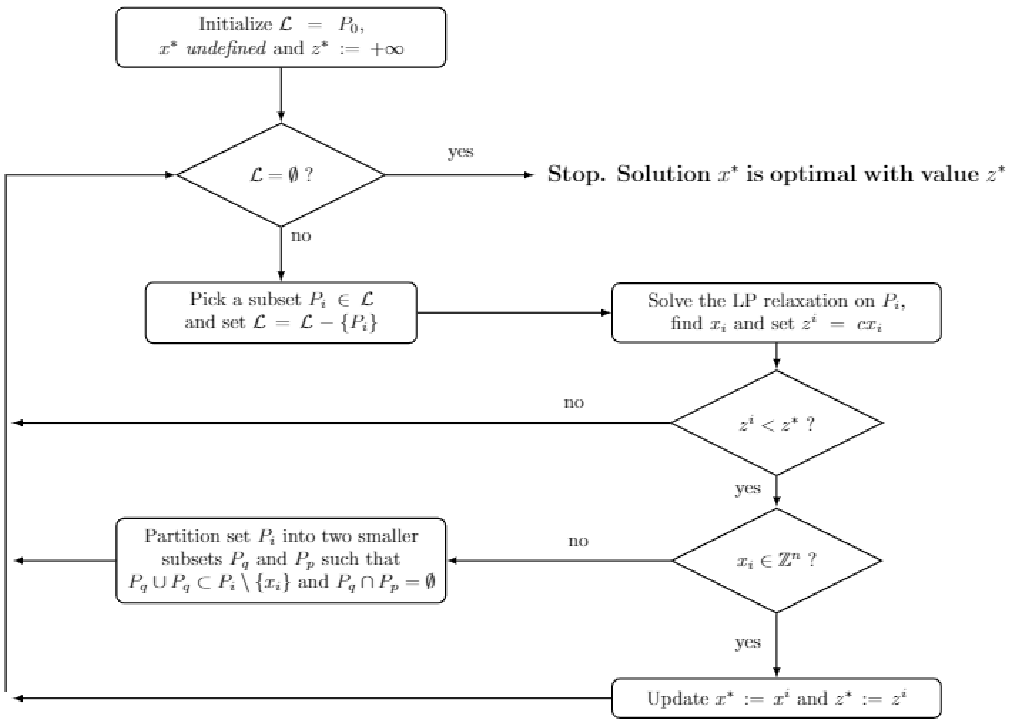

Initialize $\mathcal{L} = P_0$, x^* *undefined* and $z^* := +\infty$
$\mathcal{L} = \emptyset$?
yes
Stop. Solution x^* is optimal with value z^*
no
Pick a subset $P_i \in \mathcal{L}$ and set $\mathcal{L} = \mathcal{L} - \{P_i\}$
Solve the LP relaxation on P_i, find x_i and set $z^i = cx_i$
$z^i < z^*$?
no
yes
$x_i \in \mathbb{Z}^n$?
no
Partition set P_i into two smaller subsets P_q and P_p such that $P_q \cup P_q \subset P_i \setminus \{x_i\}$ and $P_q \cap P_p = \emptyset$
yes
Update $x^* := x^i$ and $z^* := z^i$

Figure 2.3: Flowchart of the steps performed by the Branch&Bound algorithm to solve a minimization problem.

technique, which allows to limit the search space by the evaluation of the solutions found. The idea beyond the B&B algorithm is very simple, but of course there are several ways to implement it. From the choice of the relaxed formulation[1] to the criteria of selection for the problems and branching variables, a number of factor influences the computational time taken by the algorithm. One can also modify the objective function or settle for producing only a good LB for each problem in order to speed up the resolution phase. Furthermore, alternatives and variants of the algorithm can include additional phases before the branching steps. An example is given by the so called *Branch&Cut* algorithm that strengths the LP formulation thanks to the addition of constraints limiting the feasible region of the relaxation. This constraints are called *cuts* and are based on the idea to find an inequality of type $\alpha x \leq \beta$ that "cuts out" and separate a current solution $x_i \notin \mathcal{S}$, obtained solving the problem in P_i. One can thus define a new polyhedron $P_i' := P_i \cap \{x : \alpha x \leq \beta\}$ that is stronger than the original formulation P_i. In fact, since $S \subseteq P_i' \subset P_i$, the optimal value provided by P_i' will be at least equal to the one given by x_i. We refer to the application of this method as *cutting plane.* Of course the number of cuts satisfying this type of criterion is infinite and some choices may be more effective than others in paving the way through the integer optimum. However, even though the selection is made empirically and there is "no free lunch", the Branch&Cut method represents still the foundation of the best state-of-art commercial solver for integer programming.

2.2.3 Primal-dual algorithms

We now introduce the *duality theory.* This theory provides extremely relevant results both from the theoretical and algorithmic perspectives. In particular, the duality theory allows to associate to an optimization problem its *dual problem* which, under determined hypotheses, may have a simpler structure and may be more prone to be exploited from a theoretical and/or computational point of view.
In the following we give the definition of the dual problem and present one of the most important theorem of linear programming. W.l.o.g with A, b, c we indicate respectively a matrix in $\mathbb{R}^{m\times n}$, a vector in $\mathbb{R}^m$, a vector in $\mathbb{R}^n$, and with x and y we denote two variable vectors belonging respectively to $\mathbb{R}^n$ and $\mathbb{R}^m$.

Definition 2.6. *Given a linear program of the type* $\max\{\text{cx: Ax} \leq \text{b}\}$*, its dual problem is defined as* $\min\{\text{yb: yA} = \text{c, y} \geq 0\}$*. We refer to* $\max\{\text{cx: Ax} \leq \text{b}\}$ *as primal problem.*

[1] For example, said $x_i \in P_i$ a feasible vector with x_j being a fractional component, we can divide P_i into two sets such that $P_p = P_i \cap \{x_j \leq \lfloor x_j^i \rfloor\}$ and $P_q = P_i \cap \{x_j \geq \lceil x_j^i \rceil\}$. Such a base partition method is called *binary branching.*

Theorem 2.1. *Said $\mathcal{P} := \{x : Ax \leq b\}$ and $D := \{y : yA = c, y \geq 0\}$ the polyhedra relative to a primal problem and its dual, with both $\mathcal{P} \neq \emptyset$ and $\mathcal{D} \neq \emptyset$, then there exists a solution $x^* \in \mathcal{P}$ and a solution $y^* \in \mathcal{D}$ and for them it applies that $max\{cx : Ax \leq b\} = min\{yb : yA = c, y \geq 0\}$.*

In other words, Theorem 2.1 claims that the optimality of a solution of a (primal) problem can be demonstrated by giving for its dual a solution of equivalent value. This property is called *strong duality.* It further follows that:

Corollary 2.1. *Given a couple of primal and dual problems, $max\{cx : Ax \leq b\}$ and $min\{yb : yA = c, y \geq 0\}$, and said $\bar{x}$ and $\bar{y}$ two respectively feasible solutions, the following statements are equivalent:*

i) *$\bar{x}$ and $\bar{y}$ are optimal respectively for $max\{cx : Ax \leq b\}$ and $min\{yb : yA = c, y \geq 0\}$;*

ii) *$c\bar{x} = \bar{y}b$;*

iii) *$\bar{y}(A\bar{x} - b) = 0$.*

The condition in *iii*) of the Corollary 2.1 is called *primary complementary slackness condition* and is at the foundations of the primal-dual approach. It states that for every non-zero component of the variable vector y, the corresponding constraint of the primal problem is satisfied at equality, i.e. it is active. We can also rewrite the Corollary 2.1 in a different form:

Corollary 2.2. *Given a couple of primal and dual problems $min\{cx : Ax \geq b, x \geq 0\}$ and $max\{yb : yA \leq c, y \geq 0\}$, and said $\bar{x}$ and $\bar{y}$ two respectively feasible solutions, the following statements are equivalent:*

i) *$\bar{x}$ and $\bar{y}$ are optimal respectively for $min\{cx : Ax \geq b, x \geq 0\}$ and $max\{yb : yA \leq c, y \geq 0\}$;*

ii) *$c\bar{x} = \bar{y}b$;*

iii) *$(c - \bar{y}A)\bar{x} = 0$ and $\bar{y}(A\bar{x} - b) = 0$.*

We call the first equality of *iii*) *dual complementary slackness condition.*

Primal and dual relations allow us to prove whether a linear program is unbounded of infeasible. In particular, next theorem clears this point.

		Dual problem		
		Optimal solution	*Unbounded*	*Infeasible*
Primal problem	*Optimal solution*	✓	-	-
	Unbounded	-	-	✓
	Infeasible	-	✓	✓

Table 2.1: Combinations that can occur in linear programming for a couple of primal and dual problems.

Theorem 2.2. *Given a couple of primal and dual problems min*$\{cx : Ax \leq b\}$ *and max*$\{yb : yA = c, y \geq 0\}$*, the primal problem is unbounded if and only if its dual is infeasible. If the primal problem admits an optimal solution, then also its dual admits an optimal solution.*

Table 2.1 shows all possible combinations that can occur for a couple of primal and dual problems. Observe that, the dual of a dual problem coincides with the primal problem.
We remark that the majority of the problem presented in practice admits a (finite number of) solution(s), as in the case of the real-time train traffic rescheduling applications (see Chapter 4). We therefore limit the discussion to this scenario.

Suppose now that we have a solution $\bar{y}$ of the dual problem. A classical *primal-dual method* proceeds searching for a solution of the primal problem $\bar{x}$ that satisfies the complementary slackness conditions. If none exists, a new feasible solution $\hat{y}$ with a greater value can be found solving the dual and the same procedure repeated until all the conditions are met (until the optimality of both solutions is proved). Specifically, given feasible vector $\bar{y}$, we can certify its optimality by solving an associated primal problem (*restricted primal problem*) that minimizes the violation of the complementary slackness conditions while searching for a solution in the feasible region. This approach leads to efficient algorithms when applied to "weighted" optimization problems, since it allows to exploit the formulation of its unweighted counterparts, which is an easier problem in most of the cases. Moreover, many of these type of problems can be solved combinatorially instead of using linear programming techniques. It is especially in these cases that the check on the slackness condition and the search for a new feasible dual solution can be efficiently computed. This is why primal-dual methods are also often devised to design good approximation algorithms for a (wide) variety of difficult combinatorial problems.

2.2.4 Row and Column generation methods

LPs may come with a large number of constraints defining the feasible region or may present a huge number of variables, in this cases we can make use of two general techniques called respectively *row generation* and *column generation* (or *delayed column generation*). The word "generation" expresses the fact that a number of rows or columns are embodied in the problem only as necessary. The crucial aspects of these approach retain in defining the starting sub-problem (through *warm-start techniques*) and detecting, among all, the best row and column candidates to integrate in it. Note that, as in the worst case scenario we end up building the entire problem, both methods always require a finite number of steps.

The row generation method rests on the fact that only a subset of linear constraints is actually needed to bound the area where good solutions can be found (think for example at those problem whose feasible region is defined by all subset of a given ground set). Given a generic optimization problem $min\{f(x) : x \in \mathcal{P}\}$, this method starts selecting a basic subset of constraints $\mathcal{P}' \neq \emptyset$, such that $\mathcal{P} \subset \mathcal{P}$, and then proceeds by adding a row of the matrix defining $\mathcal{P}$ every time we find a constraint violated by the current solution found in $\mathcal{P}'$. The constraints added at each iteration define the region of the polyhedron where the optimal solution is most likely to be achieved. Note that claiming the feasibility for a solution x in $\mathcal{P}$ is equivalent to state its optimality. Of course, in some cases one can also settle for a feasible solution with an acceptable level of approximation. In case of an unbounded problem the procedure terminates adding all constraints of the original polyhedron $\mathcal{P}$.
Ideally, when adding a constraint we want to form the boundaries of the original polyhedron and at the same time exclude a region of the polyhedron that is as large as possible. Besides, we want to find a trade-off between the running time of the constraint identification phase (*separation phase*) and the quality of the constraint added. We remark that the speed of the algorithm is also decided by both the tractability of the problem solved at each iteration and by the number of problems solved. The row generation method is usually used to solve mixed integer programs, starting from the linear relaxation of the model and adding rows to cut off the continuous values of integer variables, and is in general embodied in Branch&Bound algorithms (see Section 2.2.2).

On the other hand, the idea under the column generation method is to solve a manageable problem, called *restricted master problem*, defined only by a subset of variables and then check whether the optimal solution of the restricted problem corresponds to the optimal one of the whole problem, also called *master problem.* If the optimality check fails then

new variables are added to the model in order to improve the "partial" solution. In formal terms, the variables added represents the columns of the constraint matrix that defines a linear program feasible region. Note that to add a column in a primal problem implies to add a (row) constraints in its dual and, from the point of view of the latter, the procedure can be seen as a cutting plane method. Thus, to choose among the *candidate variables* that will enter the restricted master problem we can once again exploit the duality theory.
Going into details, given a problem of the type $min\{cx : Ax \geq b, x \geq 0^n\}$ the best candidate variables are the ones with the higher potential to decrease the value of the objective function. Now, said $max\{yb : yA \leq c, y \geq 0^m\}$ the dual problem and x_K the optimal solution found for the restricted problem on the variable subset K (with $|K| < n$), we can compute the dual cost vector y^*. Then, to discover the most promising variable(s) we solve the following problem, know as *pricing problem*:

$$\max_{i=1,..,n} \quad \sigma_i := c_i - y^* A_i \tag{2.3}$$

where with A_i we indicate the $i-th$ row of matrix A. The objective function σ is the *reduced cost vector* and its components represent the amount by which each variable' s objective function coefficient needs to improve before the relative variable could assume a positive value. In other words, we calculate how much the objective function can change when making a zero-valued variable positive. It follows that if no variable presents a negative reduced cost, i.e. $\sigma^* = 0$, the solution of the restricted problem is optimal for the master problem and, in this case, the primal complementary slackness condition are also satisfied. Otherwise, if $\sigma^* < 0$ we add the variable(s) x_i satisfying $\sigma^* = c_i - y^* A_i$ to the restricted master problem and iterate the procedure. The flowchart in Figure 2.4 summarizes all the steps performed by a generic column generation algorithm.
In many cases the condition $\sigma^* \nless 0$ is satisfied before the main problem becomes computationally intractable. Clearly, the column generation approach is particularly efficient when most of the variables in the final solution are equal to 0. A classical example of its efficient use is the *Cutting Stock Problem*[2]. Other problems, typically belonging to the combinatorial field, often addressed with this technique concern *Crew Scheduling*[3] and *Vehicle Routing*[4]. Moreover, such a procedure can be exploited in a B&B approach to compute the bound at each node of the branching tree; this hybrid techniques is known as *Branch&Price.*

[2] The Cutting Stock Problem is the problem of cutting pieces of stock material with a fixed width into smaller pieces of specified size, minimizing the amount of wasted material.

[3] The Crew Scheduling Problem is the problem of assigning a group of crew members to a set of scheduled flight or railway routes.

[4] The Vehicle Routing Problem is the problem of finding an optimal route for a fleet of vehicles visiting a set of locations.

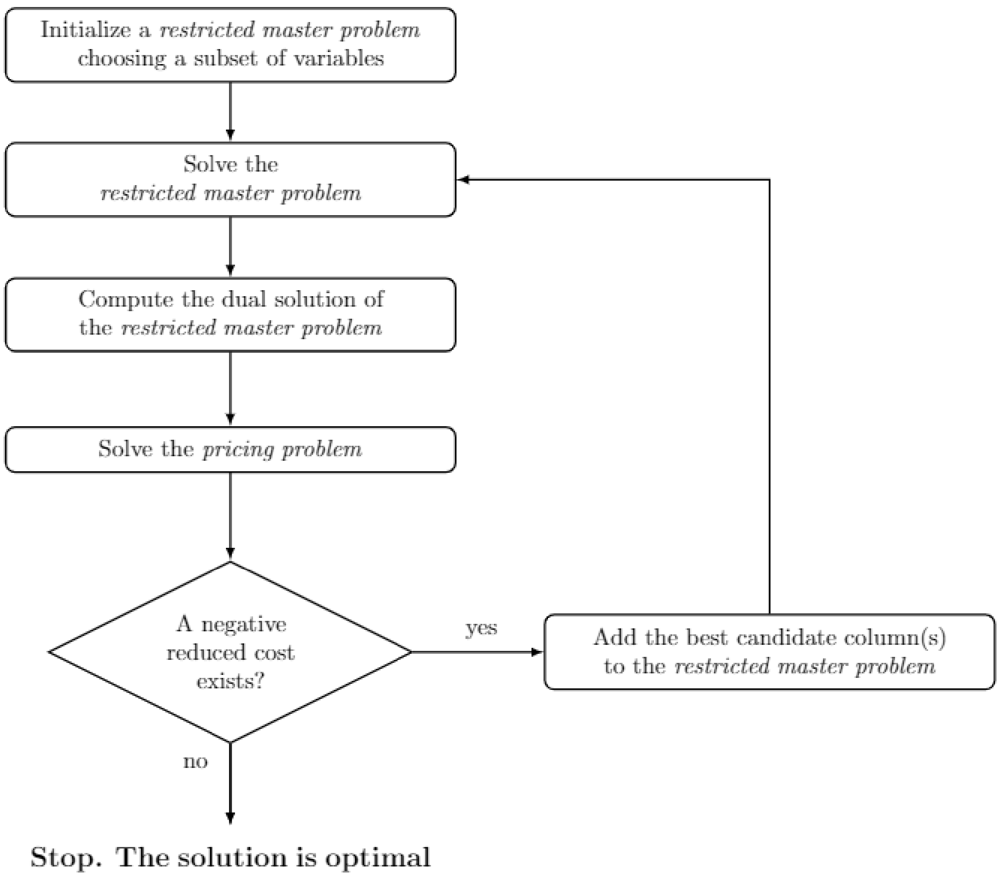

Initialize a *restricted master problem* choosing a subset of variables
Solve the *restricted master problem*
Compute the dual solution of the *restricted master problem*
Solve the *pricing problem*
A negative reduced cost exists?
yes
Add the best candidate column(s) to the *restricted master problem*
no
Stop. The solution is optimal

Figure 2.4: Flowchart of the steps performed by a generic column generation method.

2.3 Elements of graph theory

Graphs are one of the most fundamental combinatorial structure. In this section we report some essential definitions and present the basic elements that characterize the graph theory: circuits, paths and trees. We then expose two well-known combinatorial problems that have numerous and diverse applications.

2.3.1 Basic definitions

The graph theory studies the metric and topological properties of binary relations between objects. A graph is a collection of *vertices* (also referred as *node*) and *edges* (also called *arcs* or *links*) connecting them pairwise. More formally:

Definition 2.7. *An (undirected) graph G is a triple (V, E, Φ) with V and E being disjoint finite sets of vertices and edges and Φ a function that associate with each edge a pair of vertices.*

Usually, we simply write $G(V, E)$ and implicitly imply the *incidence function* Φ. For the sake of completeness we also report the notion of subgraph.

Definition 2.8. *A subgraph $H(V', E')$ of $G(V, E)$ is a graph defined on $V' \subseteq V$ and $E' \subseteq E$. Given a subset of vertices N, a subgraph $H(V', E')$ with $V' = V \setminus N$ is said induced by N if $E' = \{\{u, v\} \in E : u, v \in V'\}$. We refer to it as $G(N)$. A subgraph $H(V', E')$ of $G(V, E)$ is spanning if $V' = V$.*

In a directed graph we consider oriented edges connecting an *head* node to a *tail* node, while an undirected graph is made up of non-ordered pairs of *end nodes*. Two vertices connected by an edge are said *adjacent* to each other (or *neighbors*) and the edge is said to be *incident* to them. Given a vertex v, the set of all neighbors vertices is given by $\Gamma(v) = \{V \ni u : \{u, v\} \in E\}$. Besides, we denote with $\delta(v)$ the set of edges incident with v, and with $|\delta(v)|$ the degree of node v. A vertex with degree equal to 0 is called *isolated.* In particular, for a directed graph we have $\delta^+(v)$ is the set of edges with head node in v and $\delta^-(v)$ the set of edges with v as tail node. By extension, given a set $N \subseteq V$, $\delta(N)$ is the set of edges with an end node in N and the other in $V \setminus N$; we call $\delta(N)$ a *cut.* An edge connecting a vertex to itself is called *loop*, while two edges that are incident to the same ends are said *parallel.* A graph is called *simple* if there are no loops and parallel edges. In the remaining we will consider only simple graphs. Moreover, given a graph $G(V, E)$, the cardinality of the set V gives the *order* of the graph and is usually indicated with n, while the number of elements in the set E denotes the *size* of the graph and is commonly

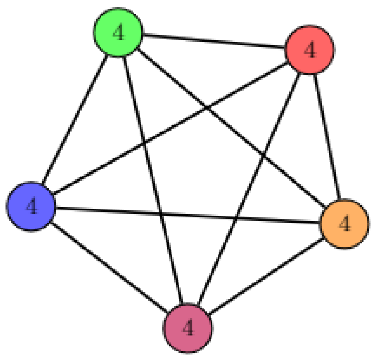


Figure 2.5: Example of a complete graph having order equal to 5. The vertex label reports the vertex degree.

indicated with m. A graph is said *complete* if it is simple and any pair of vertices in V is connected by an edge. Its degree is thus equal to $m = n \cdot (n-1) \setminus 2$ and each vertex has a degree of $(n-1)$. An example of a complete graph with 5 vertices (with degree equal to 4) is shown in Figure 2.5. Lastly, given a graph $G(V, E)$, a *complement graph* $\bar{G}(V, \bar{E})$ is such that $\hat{G}(V, E \cup \bar{E})$ is a complete graph.

Special structures.

Definition 2.9. *A walk is an alternating sequence of vertices and edges in a graph. It is said* open *if the first and last vertex differ,* closed *otherwise.*

Said $\{v_1, \{v_1, v_2\}, v_2, ..., v_{n-1}, \{v_{n-1}, v_n\}, v_n\}$ a walk, we call $\{v_1, v_n\}$ *terminal vertices* and $\{v_2, ..., v_{n-1}\}$ internal vertices. The number of edges in the walk determines the length of the walk. Note that some vertices and/or edges can be repeated. In the following we report the definitions of some basic graph structures.

Definition 2.10. *A* trial *is a walk with all distinct edges.*

Definition 2.11. *A* circuit *is a closed trial.*

Definition 2.12. *A* path *is a trial with all distinct vertices.*

Definition 2.13. *A* cycle *is a closed path.*

Depending on whether we consider an undirected graph or a directed one, walks, trials, circuits, paths and cycles may be oriented or not. We call a directed acyclic graph a *DAG*.

Connectivity.

An undirected graph is *connected* if it exists a path connecting any two pairs of vertices. Each connected subgraph is called a *connected component* or simply a *component*. A directed graph is said *strongly connected* if, for any pair of vertices u and v, it exists a path connecting u to v and a path connecting v to u. The *strongly connected components* of a directed graph are represented by its strongly connected maximal subgraphs. We highlight that:

Theorem 2.3. *A graph $G(V, E)$ is connected if and only if each subset N of vertices, such that $\emptyset \neq N \neq V$, with $\delta(N) \neq \emptyset$.*

Definition 2.14. *A forest is an acyclic graph, i.e. a graph that does not contain any cycle.*

Definition 2.15. *A tree is an acyclic connected graph.*

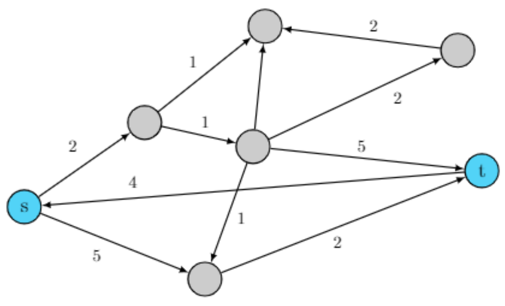
We can also say that a forest is a collection of trees. Observe that, given a tree, there is always a unique path between any pair of vertices.

2.3.2 Shortest Path problem

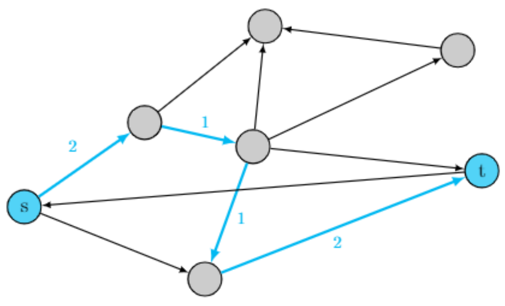
Given a graph $G(V, E)$, we can associate a real cost (*weight* or *length*) c_e to each edge $e \in e$. Said P a path in G its cost can be hence calculated as $c(P) = \sum_{e \in P} c_e$. We can now formulate the Shortest Path Problem (SPP) as follows:

Definition 2.16. *Given a graph $G(V, E)$, a source node $s \in V$ and a destination node $t \in V$, and said $c \in I\!R^{|E|}$ the cost vector of edges, find a path P from s to t with minimum cost $c(P)$.*

The SPP can be also formulated considering different variations. For example, we may have only a source vector and we aim to find the shortest path from it to all other vertices (*single-source shortest path problem*) or we may want to find the shortest paths between every pair of nodes (*all-pairs shortest path problem*). Moreover, by reversing the direction of each edge in the graph, the problem of a *single-destination shortest path problem* can be reduces to a single-source shortest path problem. Figure 2.6a shows an example of a directed connected graph and reports the costs associated with all edges. For In Figure 2.6b we highlight in blue the shortest path between its nodes s and t, such a path has an optimal value equal to 6.
By definition, if the graph is connected we know that there exists a path P connecting any two pairs of nodes. In this case we can hence exclude the infeasibility of a SPP. Next theorem, that applies both for directed and undirected graphs, states the condition under which the problem is also bounded (and admits an optimal solution).



(a) A directed and connected simple graph.



(b) Shortest path from node s to node t.

Figure 2.6: Example of a shortest path from s to t for a directed and connected graph.

Theorem 2.4. *Given a connected graph $G(V, E)$, a source node $s \in V$ and a destination node $t \in V$, and said $c \in \mathbb{R}^{|E|}$ the edges cost vector, it exists a shortest path from s to t if and only if G does not contain negative cost cycles.*

Proof. Note that, since $G(V, E)$ is connected, there exists at least a path P connecting node s to node t.
If. If no negative cost cycles exist, we can remove all cycles without increasing the cost of all possible paths. Then, we can compute a finite number of paths between s and t and choose the one with lower cost.
Only if. Suppose now that it exists a cycle going from u to v having a negative cost. Since the graph is connected, we can construct a path P' going from s to u, traversing the cycle an arbitrary number of times and then connecting v to t. Thus, we can always produce a path with a lower (negative) cost and this proves that no solution is optimal and that the problem is instead unbounded.

□

Form the mathematical standpoint, a path P can be defined by an incidence vector $x \in \{0,1\}^{|E|}$ of the type:

$$x_e = \begin{cases} 1 & \text{if edge } e \text{ belong to } P \\ 0 & \text{otherwise} \end{cases}$$

Moreover, for a path P going from vertex s to vertex t we have that:

(i) $\sum_{e\in\delta^-(s)} x_e = 1$ for the source vertex s;

(ii) $\sum_{e\in\delta^+(t)} x_e = 1$ for the destination vertex t;

(iii) $\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e$ for all other vertices such that $v \neq \{s,t\}$.

Hence, the SPP can be simply stated as an ILP of the form: $min\{cx : x$ s.t. *(i)-(iii)*, $x \in \{0,1\}^{|E|}\}$. When no negative cost cycles exists the SPP can be easily solved and the optimal path found efficiently (see *Dijkstra's algorithm*, *Floyd-Warshall's algorithm*, *Bellman− Ford algorithm* etc.). The SPP resolution turns harder when the cycles do exist and the problem belongs, in this case, to the class of $\mathcal{NP}$*-hard* problems (see Section 2.2.1).
However, such a particularly complex issue goes beyond the scope of this work and we refer an interested reader to [36] for a more in-depth discussion. We instead conclude the section by presenting a problem related to the SPP that we will use in the future chapters, namely the (single-pair) Longest Path Problem (LPP) on a DAG:

Definition 2.17. *Given a DAG $G(V,E)$, a source node $s \in V$ and a destination node $t \in V$, and said $l \in I\!R^{|E|}$ the edges length vector, find a path P with maximal length $l(P)$.*

We underline that the LPP can be seen as a SPP obtained by inverting the sign of the cost vector. From the Theorem 2.4 and the assumption of a DAG, it immediately follows that there always exists an optimal solution for the problem. Moreover, a crucial result for all scheduling problems is that such a solution can be computed in linear time thanks to a *topological ordering*.

2.3.3 Maximal Stable Set problem

Said $G(V,E)$ a graph, a *stable set* (*independent set* or *vertex packing*) S is a set of nodes no two of which are adjacent. We call a *clique* a set C of pairwise adjacent nodes. Therefore, given a stable set S there is no edge in $G(V,E)$ connecting two vertices of S. On the other hand, given a clique C there is always one connecting two node in it. The size of a stable set (clique) is given by the number of vertices in the set and is referred as *stability number*. We call a stable set (clique) *maximal* if it is not properly contained in any other stable set (clique) of the graph. That is, no other vertex $v \in V \setminus S$ ($v \in V \setminus C$) can join the

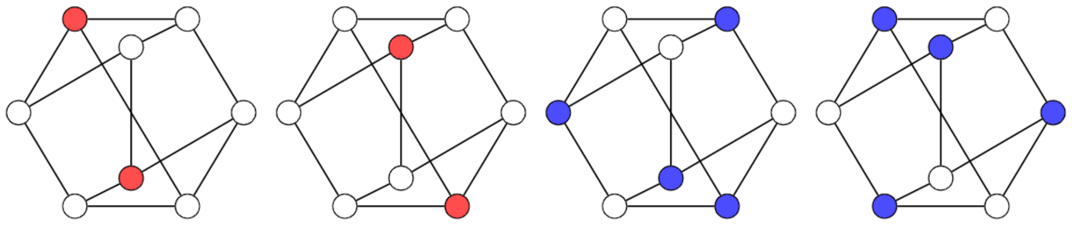


Figure 2.7: Examples of stable sets (colored nodes) for a graph. The blue nodes belong to maximal stable sets.

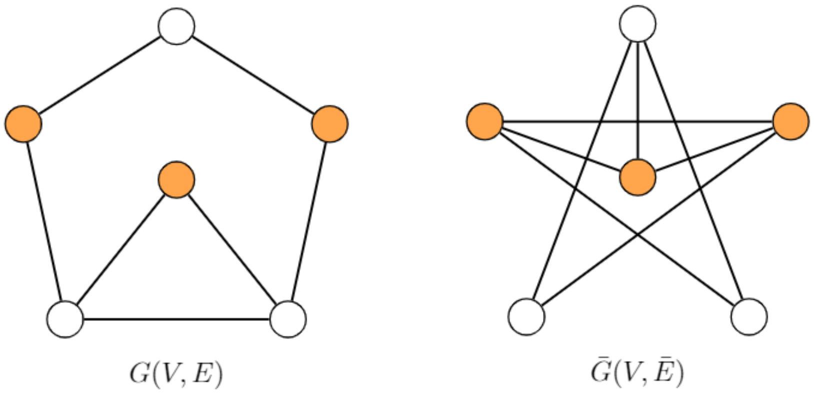


Figure 2.8: A set of nodes defines a maximal stable in a graph if and only if it defines a maximal clique in its complement graph.

stable set S (clique C) and increase its size. We remark that the subgraph $G(C)$ induced by a maximal clique C is always complete. In Figure 2.7 we mark in red two different examples of stable sets for a graph and color the two maximal stable sets of the graph in blue.

The Maximal Stable Set Problem (MSSP) is a typical example of an $\mathcal{NP}$-*hard* combinatorial problem that has many real-life applications. It can be formally defined as follows:

Definition 2.18. *Given a graph $G(V, E)$, the Maximal Stable Set Problem (MSSP) is the problem of finding a stable set that is maximal in it.*

Note that the MSSP in $G(V, E)$ is equivalent to the problem of finding a clique of maximum cardinality in its complement graph $\bar{G}(V, \bar{E})$ (see Figure 2.8 for an example). Moreover, said $G(V, E)$ a weighted graph and $W(\cdot)$ the function that associates to each vertex v a (rational) weight w_v, we can define the weight of a stable set S as $W(S) = \sum_{v \in S} w_v$. Then:

Definition 2.19. *Given a graph $G(V, E)$ and a weight function $W(\cdot)$, the Maximum Weight Stable Set Problem (MWSSP) is the problem of finding a stable set S of maximum weight*

$W(S)$.

The MSSP can be seen as a particular case of the Maximum Weight Stable Set Problem (MWSSP), where all the vertices have a weight equal to 1. Besides, as for the MSSP, searching for a maximum weight stable set in $G(V, E)$ is equivalent to search for a maximum weighted clique in $\bar{G}(V, \bar{E})$.

The M(W)SSP can be mathematically defined through many different and equivalent formulations of integer programming. Before providing a pure binary model for it, we need to introduce the definition of a *packing constraint.* Indeed, all the integer programming solvers in commerce rely on the graph representations of this type of constraints to formulate the logical condition involving two (ore more) binary variables in an ILP.

Definition 2.20. *Given a finite set E and a family $\mathcal{F} := \{F_1, ..., F_n\}$ of subsets of E, a subset $S \subseteq E$ is a packing of $\mathcal{F}$ if S intersects each member of the family $\mathcal{F}$ at most once. We have that:*

$$S := \{x \in \{0,1\}^{|E|} \ : \ \textstyle\sum_{v \in F_i} x_v \leq 1 \ , \ \forall \ F_i \in \mathcal{F}\} \ .$$

Next, we define the incidence vector $x \in \{0,1\}^{|V|}$ of a set S as:

$$x_v = \begin{cases} 1 & \text{if node } v \text{ belong to } S \\ 0 & \text{otherwise} \end{cases}$$

Now, since the set of edges $E \subseteq V \times V$ of a graph $G(V, E)$ is a family of subsets of V, the MWSSP can be therefore formulated through a group of packing constraints. We have the following *edge formulation*:

$$\begin{array}{lll} \max & \sum_{v \in V} w_v x_v & \\ s.t. & x_u + x_v \leq 1 & \forall \ \{u, v\} \in E \\ & x_v \in \{0,1\} & \forall \ v \in V \quad . \end{array} \tag{2.4}$$

In general, such a formulation does not provide thigh bound, that is the value of the optimal solution of its LP relaxation is not close to the optimal integer solution of (2.4). However, the formulation can be strengthened by exploiting some families of inequalities, like those related to cliques. In fact, one can replace the *edge constraints* of all pairs of nodes joining a clique with a single condition of the type $\sum_{v \in C} x_v \leq 1$. This inequality allows to both reduce the number of constraints and tighten the formulation. The strongest formulation is then obtained by defining the packing constraints relative to all maximal cliques. Note

that the inequality generated by a clique C' such that $C' \subset C$ is always dominated by the inequality defined by the clique C. Unfortunately, the number of clique inequalities of a graph can be exponential and the related separation problem is also $\mathcal{NP}$-*hard*.
A wide variety of heuristic and exact methods, and even reinforcement learning algorithms, have been proposed over the years to solve the M(W)SSP, since the problem becomes very difficult as the size of the graph increases. Naturally, one can also exploit the special structure of the problems under analysis to add valid inequalities and provide tighter constraints. Besides, many families of graphs exhibit properties that make the MWSSP solvable in polynomial time. In the present work we make use of MWSSP as a subroutine of the implemented algorithm for real-time train re-scheduling (see Chapter 4).

Chapter 3

The Real-time Train re-Scheduling Problem

3.1 Introduction

Real-time train re-Scheduling is one of the most challenging tasks of the railway systems management. Due to the limited solution time, the high interaction between processes, and the different and sometimes conflicting needs of the agents involved in the process, the search for a global (and optimal) new schedule can be hard and laborious. Moreover, an initial delay on a train route can propagate as a secondary delay on other portions of the network causing severe knock-on effects on the whole system. The wider the disturbance taking place in the net, the harder is the recovering of the timetable planned off-line. The dispatchers have the arduous task of supervising the correct traffic execution and intervene, according to aforethought control sequences and dispatching rules, to re-establish a regular flow. In addition, the uncertainty on the future state of the network contributes to increase the inherent complexity of the problem. Indeed, the solution just reached through a re-scheduling may become impractical in the near future as a result of additional system disturbances and disruptions.

Over the past 50 years, the literature focused on the mathematical representation and resolution of the Real-time Train re-Scheduling Problem (RTSP) while its implementation in the field still leaves key players in traffic management skeptical. As a matter of facts, the Decision Support Systems (DSSs) and Advanced Traffic Management Systems (ATMSs) practically implemented in the recent years have been limited to small sub-systems and resulted to be capable to tackle only "smooth" disturbances. Commonly, the structure of RTSP reduces to a Job-shop Scheduling Problem (JSP): the jobs are trains, while the sequences of operations performed on the different machines are represented by the track segments traversed on the train service routes. Scheduled arrival and departure times are seen as due dates, whilst additional constraints may establish precedence orders between operations and other system requirements. In general, the railway system modelling can be done using a macroscopic or a microscopic approach, depending on the level of detail considered for the network. Usually, the primary goal is the minimization of the deviation from the off-line timetable that, in most of the cases, is translated in terms of primary and/or secondary delays. On the other hand, the plans cost-effectiveness is often marginally affected by rescheduling due to disturbances and/or disruptions and, for this reason, is poorly addressed in the literature. The resolution of the RTSP is commonly addressed to the use of *disjunctive graph* formulations and the problem is easily translated into Mixed Integer Linear Programs (MILPs) (the so called *big-M formulations*). Alternative formulations for train scheduling, and for the scheduling problems in general ([131]), come from the use of pure binary models, as for the *time-indexed* formulation. However, the search for a feasible, and/or optimal, schedule is not so trivial as its modelling and different approaches and algorithms, heuristics and meta-heuristics have been proposed through the years.

The aim of this chapter is firstly to define in a structured way the RTSP, and secondly to present the basic mathematical concepts and the main approaches adopted in the literature to solve it (Section 3.2 and Section 3.3). In Section 3.4 we introduce the notions required to understand the models and techniques we implemented to tackle the train dispatching in a reactive and proactive way (see Chapter 4 and Chapter 5). In particular, we show how the RTSP can be formulated as a JSP and solved through the use of disjunctive programming and linear models. A brief discussion on the motivations that led us to the model presented in Chapter 4 closes the chapter.

3.2 State-of-the-art

The scientific literature on train scheduling is very extensive and dates back almost half a century. From the first example of linear programming in the train scheduling problem in the early 70's (see [145]), similar approaches exploiting Branch&Bound techniques were adopted in the next decades for the timetable construction of single-track lines ([84, 85, 88, 115, 123, 126]). In 2004, the *blocking time theory*, which allows a blocking time stairway to be reserved for each train and avoids any time overlapping between two different rolling

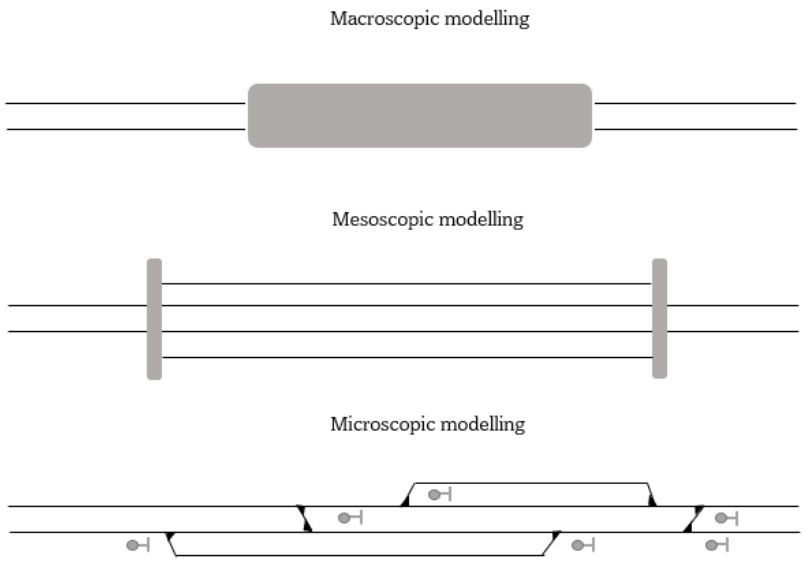


Figure 3.1: Example of macroscopic, mesoscopic and microscopic representation of a portion of a railway network made by two main lines and a station.

stocks (see [150]), is finally recognized as a standard. Since its successfully introduction in the re-scheduling models (thanks to the use of the disjunctive graph formulations [113]), a number of applications exploiting its possible translation into (mixed) integer programs have been proposed in the literature making it the most widely used approach ([135]).
Several models have been proposed to mathematically represent the train scheduling problem through the years. According to the level of detail they can be "roughly" categorized into *macroscopic* or *microscopic* models. Note that there is no exact definition for the characteristics of each classification and the distinction is often blurry. We can also find models that combine macro and micro aspects, called *mesoscopic models*, as in [31] and [98]. Figure 3.1 shows an example of the three modelling approaches. Naturally, a trade-off between the fine modelling and the formulation efficiency should be considered. Work [93] reports the performance, in terms of solution quality and computation times, of different modelling choices. Most of the models proposed so far represent the time horizon as time-continuous, while the discrete-time representation turns out to be less addressed. A rigorous definition of the these two methods, with their strengths and weaknesses, is presented in [107] where the authors analyze the use of integer programming to formulate and solve train dispatching problems. Moreover, in [80] Harrod and Schlechte compare the *physical block occupancy* and the *timed block occupancy* representation on a set of instances and establish a relevant

baseline comparison, concluding that the two models provide comparable solutions in the aggregate.

Extensive surveys tackling different aspects of the RTSP can be found in [16], [27], [38], [46], [64], [91], [100], [96] and [159].
Paper [38] conducts a preliminary wide investigation of rail optimization in all its aspects, from strategic planning to routing and scheduling.
In [27], in addition to providing a clear distinction between the terms disturbance and disruption (see next Section for more details), Cacchiani at al. classify the academic literature into two categories, dealing with macroscopic or microscopic approach. They point out that the number of models treating a macroscopic level of detail far exceeds the number of microscopic models studied. Besides, they show how the number of works addressing the customer perspective is quite limited.
In [64] Fang et al. review 128 papers and, like previously published surveys [118] and [148], divide the management of a railway system in four levels (strategic, tactical, operational and re-scheduling) that differ for the decision level considered and the time horizon spanned. The articles are classified based on the mathematical/non-mathematical strategies adopted, the solution approaches, the problem types and the chosen objective functions.
Corman and Meng in [46] give a specific focus on the dynamic and stochastic aspects of the railway traffic re-scheduling problems. Two main distinctions are explored: static approaches (executed once and for all) versus dynamic approaches (acquiring information over time); and reactive approaches (considering no future) versus proactive approaches (looking probabilistically at the next states of the system).
On the other hand, the work presented by Lamorgese et al. in [100] provides an overview on different train dispatching aspects and summarizes the solution methods adopted over the years to solve them. The authors push for a greater exploitation of the theoretical results achieved in the practical development of automated re-scheduling systems.
Besides, passenger-oriented re-scheduling methods and models are investigated in [91], where the authors suggest a more judicious choice of performance metrics to maximize customers satisfaction.
Recent survey [159] categorizes 153 works after defining three different types of data-driven models: stochastic methods, graphic models and machine leaning methods. Further, comprehensive reviews and relevant insights for the recent application of big data in the context of railway engineering can be found in [74] and [95].
Paper [96] explores instead challenges and limitations of the Artificial Intelligence (AI) applications in the rail industry. Work [16] gives a comprehensive review of research projects

and published articles after presenting the state-of-the-art of AI in the field. Currently, AI is used to predict delays, avoid disruptions, regulate rolling stocks speeds, and more (see [144] and [55]), and represents an effective lever to increase the market share of railroads. The downside, however, is represented by a very real threat of cyber-attacks and by the blurred line traced between passenger privacy and acquired information.

Nevertheless, most of the presented works are based on abstracted models and simplified assumptions and none of the methods and algorithms proposed so far is able to solve real-world or continental-scale train dispatching problems in a competitive time. There have been only a handful of real-life implementations, and in any case rather small in size or restricted to special cases. The implemented solution approaches are indeed typically limited to simple local heuristics, and most research advances remain on paper or, at best, at the prototype level. Thankfully, the landscape has changed quite rapidly over the past few years. Over the past two decades, the interest in the development of DSSs and large scale Traffic Management Systems (TMSs) to automatize the dispatching process has grown remarkably. Through the use of information technology systems, advanced software and remote equipment, a TMS is able to provide a permanent control over the entire rail network and enables long and short-term planning. A TMS may require human interaction to approve, validate, and implement the solution provided by the software (semi-automation) or may directly forward the instructions to a signaling system (full automation). The potential of such support tools, and especially those embedding optimization-based algorithms, is increasingly recognized by infrastructure managers and operators.
In [24] the authors discuss three success stories on integrated optimization in railway systems and present a real-world successful implementations for each of them. As for the real-time train dispatching area, they describe two real-time traffic control applications (reported in [106] and [108]) on a Mass Transit (generally dedicated to urban public transport) and on a Main Line (one of the main arteries of a railway system), which are capable to provide a real-time timetable solution through a closed loop control and the use of MILP. The optimization-based models were implemented for the metro system of the city of Milan and for the Italian single-track line Trento-Bassano and were fully operative by the end of 2012. Several European projects have also been launched in the practical field. Examples include: ARRIVAL (Algorithms for Robust and on-line Railway optimisation) [61], whose aim was to develop the foundation algorithmic research for robust and on-line train service planning; ON-TIME (Optimal Networks for Train Integration Management across Europe) [62], carried out by a multidisciplinary team of 19 partners in 4 countries (Sweden, UK, Netherlands and Italy), the project developed new methods and processes to make the European

rail network more reliable and resilient (see also [130]); and COMBINE (enhanced COntrol center for a Moving Block signalling systEm) [60], for the implementation of innovative optimization tools in the *moving block traffic management systems* (see also [114]).
In [49] a DSS called ROMA (Railway traffic Optimization by Means of Alternative graphs) is applied in two densely used areas of the Dutch railways and proved to be able to successfully handle real-time schedule disturbances. [58] presents ICONIS (Integrated CONtrol and Information System), an advanced system implementing AGLIBRARY, a tailored optimization system that is capable to quickly compute near-optimal conflict-free train schedules and that has been applied to a large area of the British railway. A new operational traffic control system whit a user-centred design process, called STEG (from Swedish "Steering of Train Traffic with Electronic Graph"), has also been developed and operationally tested in Sweden (see [149]). Furthermore, since 2011 INFORMS has issued a "Railroad Problem Solving Competition" to promote the interest of railway operators in implementing automated decision-making systems. This year's challenge focuses on a possible rule of the RFID reader technology in producing a more accurate estimate for trains arrivals times (see [132]). The Chinese project "Safety, Reliability, and Disruption Management of High-Speed Rail (HSR) and Metro Systems" ([151]) aims to create the fundamental science and methodologies for rail safety, monitoring, and control by involving an international consortium of experts.

In addition, some infrastructure managers in Europe, such as Bane NOR in Norway (see [9]), DSB in Denmark (see [56]), UP in North America (see [87]), and others, have issued tenders for a renewal of their TMSs including some explicit requirements for autonomous and optimized real-time scheduling functions.
In North-America, Class I railroads[1] Norfolk Southern and General Electric Company (GE) deployed an automated movement planner which incorporates some optimization capabilities. In [23] Bollapragada et al. show how the software-based optimization algorithm is able to dispatch thousands of trains by considering, among other things, additional tasks such as crew expiration and relief crews transportation. Using a Branch&Bound approach with a beam-search branching strategy the algorithm delivers, in less than two minutes, an eight-hour plan that minimizes the weighted sum of several objectives (formulated as economic measurements). The GE movement planner was recently rolled out also by Aurizon, the largest rail freight operator in Australia (see [4]).

[1] The United States classifies the railroad carriers into Class I, II, or III according to size criteria established by the Surface Transportation Board (STB). Class I includes the seven largest railroads in North America.

Similarly, the North America's largest freight rail network BNSF Railway is directly involved in advanced research on dispatch optimization. The work presented in [25] addresses the railway multiterritory dispatch planning problem on this Class I railroad.
Far from being exhaustive, the above examples show that the interest in ATMSs is on the rise and that the penetration of such tools, or at least the awareness of their potential, is growing in the industry.

Besides, as highlighted by the authors of [92], an evaluation framework to access the real capability and performances of the various algorithms and approaches proposed in the literature is still missing. They hence propose a tool that can be exploited by practitioners to select the most appropriate strategy to adopt, considering the characteristics of the railway system under analysis.

3.3 Problem description

A railway network is a set of basic elements, such as lines, stations, tracks, junctions and signals (see Figure 3.2). In the first place, a railway can be decomposed in sub-networks called *lines*. Lines can be traversed by trains in two directions, conventionally called *east-bound* and *west-bound*. Each line can be thus defined as the ordered set of the *track segments* (or *block sections*) that make up its route and that can accommodate only a train at a time. The beginning and the end of each track segment is denoted by the presence of a *signal*. According to the color of the signal (red, green or yellow in a standard *fixed block signaling system*) some train movements can be denied. Signals, in combination with interlocking and Automatic Train Protection (ATP)[2] systems, enable compliance with the minimum safety distance among trains. In a fixed block system only a train at time is allowed to occupy a track segment, while in a moving block system the safety is maintained thanks to speed regulation and the continuous control on train positions. We refer the reader to [124] and [76] for further information on the argument.

With the term *stations* we identify railway facilities (or areas) where trains can stop, possibly load/unload passengers and/or freight and perform service operations. A station can be reached directly through segments on adjacent tracks, or from different tracks thanks to *junctions* (or *switches*) that allow the connection.
A train visits a sequence of stations running on track segments in two possible directions.

[2] The ATP is a type of train safety system that continuously monitors the speed of trains and is able to force the rolling stocks braking in some circumstances.

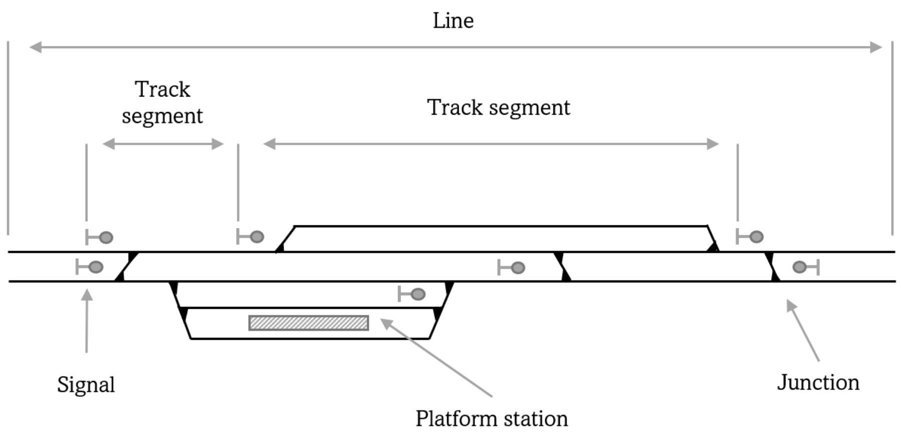


Figure 3.2: Schematic microscopic representation of the main elements of a railway system.

Trains running in opposite directions are called *crossing trains* and may share some tracks on the network; they will cross them in opposite orders. Trains running in the same direction are instead called *followers*. Each train has its own speed profile depending on the rolling stock and infrastructure characteristics, and on the actions taken by the driver. The train speed profile influences the *running time* (or *occupation time*) needed by each train to traverse a segment. We can distinguish between a theoretic (nominal/scheduled) running time and the running time truly occurred when observing a railway system or sub-system. Usually, the theoretic running time is given as the sum of the time needed by the train to traverse a segment at a theoretical speed plus a possible dwell time at a platform and some buffer times inserted to avoid/reduce potential delays.

A *timetable* is a list of all established trains routes and their scheduled times planned off-line to meet the railway system traffic constraints and requirements. A sequence of track segments to be visited is given for each train, along with the expected arrival and departure times. One can visualize the trains movements on a net using a *train graph*, a tool displaying, as a function of time, all trains running across the main rail segments. Figure 3.3 shows an example. On the y-axis we can see a portion of a railway line, while the time is plotted on the x-axis. Each piece-wise line on the graph depicts a trains, moving from A to B with a positive slope, and from B to A when showing a negative slope. The slopes of the segmented lines make it possible also to represent trains speeds: the higher the slope the faster the train. Note that a train waiting at a track segment is represented by an horizontal line. The points were two line intersect indicate instead meet-and-pass events, i.e. locations where trains moving in opposite directions may meet. It is here that the priorities and the timing for the blocks occupancy must be decided.

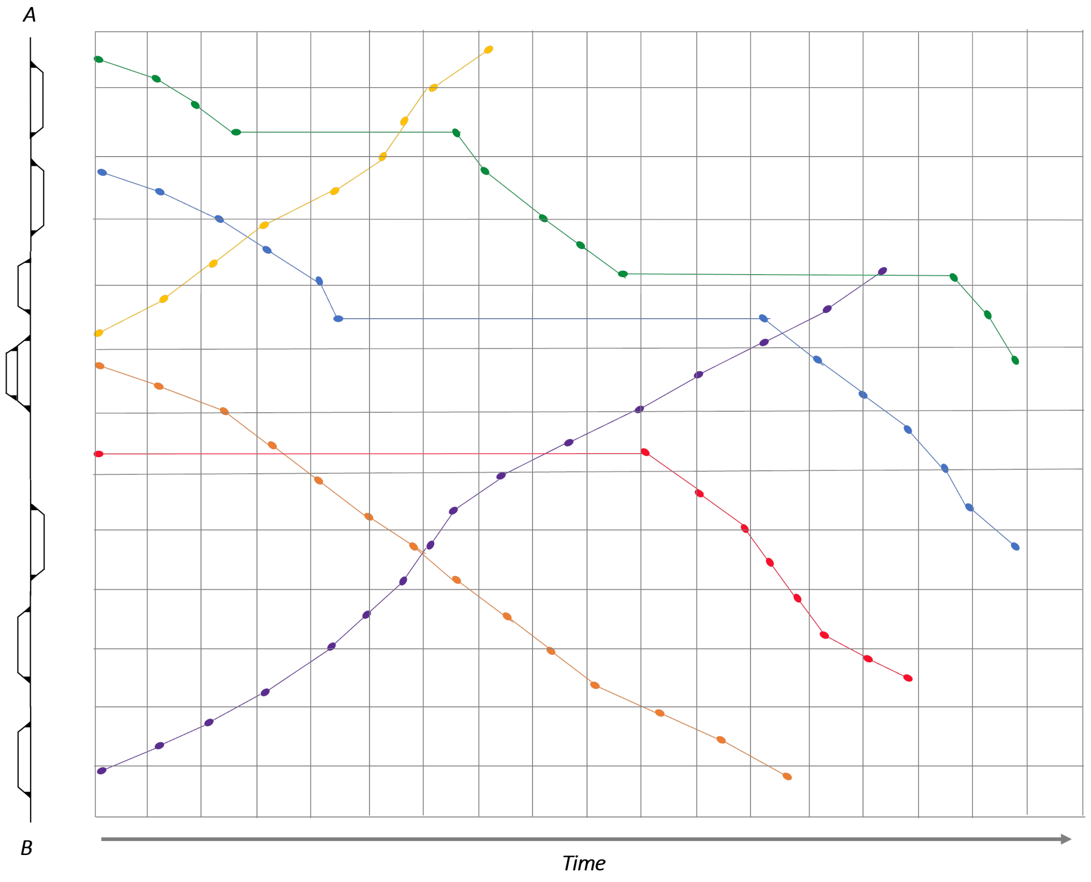
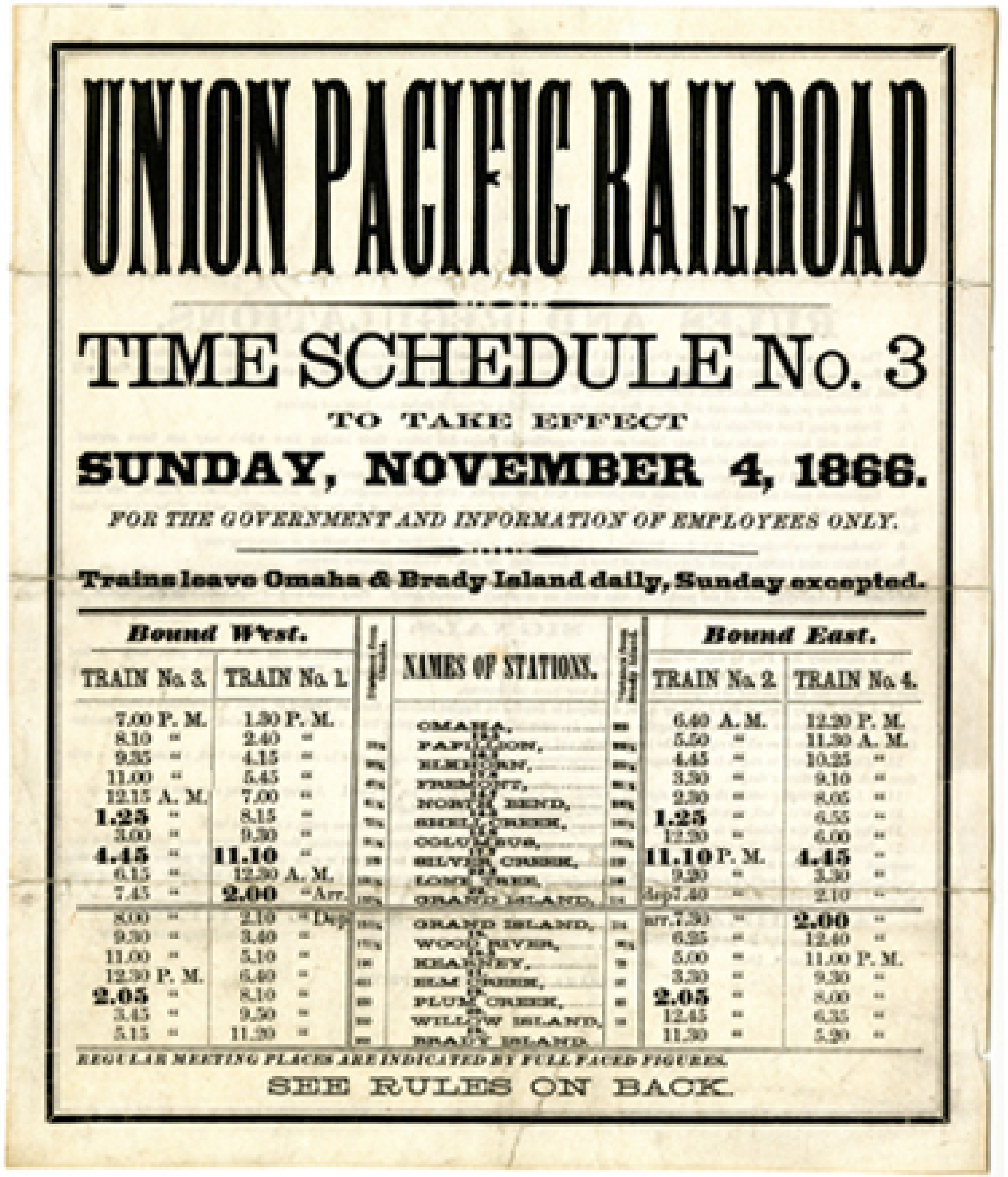


Figure 3.3: Example of a train graph that allows to visualize all trains movements on a portion of a railway line going from station A to station B.

The formulation of a timetable is typically done off-line by means of efficient mathematical programming and optimization techniques. With the work [29] the authors present the state-of-the-art methods for obtaining robust timetables, i.e. plans capable to avoid the delay propagation as much as possible, and show their main advantages and drawbacks. While paper [28] reviews the literature on nominal and robust versions of the problem, highlighting the differences between several models and methods developed over the years. Still, the real-time application of this planned schedules remains non-trivial. A large set of endogenous and exogenous factors and unexpected events can influence the trains adherence to the established times. Weather conditions, over-crowded stations platforms, drivers behaviors, rolling stocks breakdowns, damages at the physical infrastructure or signal malfunctions, can determine delays and deviations from the planned activity. Work [27] makes a distinction between unexpected events that cause minor deviations from the scheduled times, referred in the literature as *disturbance*, and incidents causing the interruption of the railway system, denoted with the term *disruption*.
We can also classify the delays in *primary* (or *initial*) delays and *secondary* (or *knock-on*) delays. The first refers to the delays affecting the track segment(s) or the network area where the disturbance/disruption occurred, while the secondary delays are generated by the propagation of the primary ones. Goverde ([77]) analyzes the delay propagation behavior through an algorithm capable of handling large-scale networks. Paper [137] investigates instead on how to determine, once given the amount of delay registered, its primary and secondary causes through some Machine Learning techniques.
The occurrence of delays can make a timetable infeasible and produce *conflicts* and *deadlocks*. We say that there is a conflict between two, or more, trains if they are trying to access a track segment at the same time. On the other hand, in a deadlock configuration trains cannot move ahead due to the positioning of other trains; in this case at least one rolling stock must be moved backwards to restore a feasible situation. A set of trains is called *bound-to-deadlock* if, starting from their current positions and time, they will inevitably end up in a deadlock (see [48] for a more in-depth discussion).

The *real-time train re-scheduling*, also referred as *train dispatching*, aims at restoring a feasible and reliable situation for the railway system. We can separate two different facets: the *traffic re-scheduling* from the *transport service re-scheduling*. The aim of the traffic re-scheduling is to establish new scheduled times for the trains affected by a delay acting on service re-timing and re-ordering. On the other hand, the transport service re-scheduling concerns the possibility to define new trains routes, the cancellation of some courses and the



UNION PACIFIC RAILROAD

TIME SCHEDULE No. 3

TO TAKE EFFECT

SUNDAY, NOVEMBER 4, 1866.

FOR THE GOVERNMENT AND INFORMATION OF EMPLOYEES ONLY.

Trains leave Omaha & Brady Island daily, Sunday excepted.

Bound West.					*Bound East.*	
TRAIN No. 3.	TRAIN No. 1.	[illegible]	NAMES OF STATIONS.	[illegible]	TRAIN No. 2.	TRAIN No. 4.
7.00 P. M.	1.30 P. M.		OMAHA,	[illegible]	6.40 A. M.	12.20 P. M.
8.10 "	2.40 "	[illegible]	PAPILLION,	[illegible]	5.50 "	11.30 A. M.
9.35 "	4.15 "	[illegible]	ELKHORN,	[illegible]	4.45 "	10.25 "
11.00 "	5.45 "	[illegible]	FREMONT,	[illegible]	3.30 "	9.10 "
12.15 A. M.	7.00 "	[illegible]	NORTH BEND,	[illegible]	2.30 "	8.05 "
1.25 "	8.15 "	[illegible]	SHELL CREEK,	[illegible]	**1.25** "	6.55 "
3.00 "	9.30 "	[illegible]	COLUMBUS,	[illegible]	12.30 "	6.00 "
4.45 "	**11.10** "	[illegible]	SILVER CREEK,	[illegible]	**11.10** P. M.	**4.45** "
6.15 "	12.30 A. M.	[illegible]	LONE TREE,	[illegible]	9.20 "	3.30 "
7.45 "	**2.00** " Arr.	[illegible]	GRAND ISLAND,	[illegible]	Dep. 7.40 "	2.10 "
8.00 "	2.10 " Dep.	[illegible]	GRAND ISLAND,	[illegible]	Arr. 7.30 "	**2.00** "
9.30 "	3.40 "	[illegible]	WOOD RIVER,	[illegible]	6.25 "	12.40 "
11.00 "	5.10 "	[illegible]	KEARNEY,	[illegible]	5.00 "	11.00 P. M.
12.30 P. M.	6.40 "	[illegible]	ELM CREEK,	[illegible]	3.30 "	9.30 "
2.05 "	8.10 "	[illegible]	PLUM CREEK,	[illegible]	**2.05** "	8.00 "
3.45 "	9.50 "	[illegible]	WILLOW ISLAND,	[illegible]	12.45 "	6.35 "
5.15 "	11.20 "	[illegible]	BRADY ISLAND.		11.30 "	5.20 "

REGULAR MEETING PLACES ARE INDICATED BY FULL FACED FIGURES.

SEE RULES ON BACK.

Figure 3.4: Historic timetables of the Union Pacific Railroad dating back to 1866.

Source: *https://www.uprrmuseum.org/uprrm/exhibits/curators-corner/employee-timetables/index.htm* short-turning of the rolling stocks. Generally speaking, the main challenges of the real-time re-scheduling lie in:

- the difficulty of acquiring accurate and up-to-date data from the field,
- the limited time available to take effective and efficient corrective actions,
- the complex implementation of an automated DSS that is required to be also fast and reliable.

Moreover, depending on the network congestion and the amount of the primary and secondary delays, the generation of a new plan can be more or less difficult. In a high-traffic system, most planners fail to provide a new global feasible schedule in a short time frame. This can be due to both the time spent acquiring data from the system and the inherent complexity of the solved model. The expertise of dispatchers is therefore critical to ensuring continuity of service. Besides, since train operations are assumed to be stochastic processes, the traffic controllers are called upon to choose between various strategies, such as adjusting running speeds, altering dwell times, and modifying trains overtakes in order to absorb the delays. We remark that, when a severe disruption occurs, they can make use of emergency timetables (called *contingency plans*) that are computed off-line according to transport operating companies and infrastructure managers guidelines.
When attempting to restore a timetable feasibility, the key goal is to deviate as little as possible from the original plan. In doing so, the majority of the models proposed in the literature focus on the minimization of trains delays. These can be averaged, summed, or only the maximum can be considered; in addition, one can also account for knock-on delays. Other objectives may relate to the maximization of passengers satisfaction (see [147]) or attempt to weight the perception of customer discomfort respect to the deviation from the planned schedule. Similarly, speed regulation and energy consumption aspects, as well as the possibility of track re-assignments, can be taken into account when modelling the train re-scheduling at a microscopic level. For more comprehensive reviews on the topic we address the reader to [27] and [64].

Summarizing, the RTSP can be defined as follows.

Definition 3.1. *Given a network and its current state at time $\underline{t}$, a set of trains, their ordered list of movements along with the planned arrival and departure times in and from each track segment, find a new feasible schedule by re-timing, re-sequencing and/or re-routing trains according to their position at time $\underline{t}$ while minimizing a measure of the deviation from the original timetable.*

3.4 Problem formulation

As first showed by Szpigel in 1973 ([145]), the RTSP can be formulated as a JSP. A JSP is a well-known *$\mathcal{NP}$-hard* combinatorial problem (see [3, 102, 128] and Section 2.2.1 for insights) that can be defined as follows:

Definition 3.2. *Given a set of machines, a set of jobs defined as a sequence of operation and their processing times, assuming that the machines are unitary resources (i.e. each machine can process a unique job at a time) find a schedule for all jobs optimizing a given objective function.*

A job is an ordered list of operations, referred also as *tasks* or *activities*. When an operation is performed all at once on a specific resource, we say that no preemption is allowed. Common objectives used in the JSP are the minimization of the maximum completion times or of the maximum lateness/tardiness or also the minimization of the number of tardy jobs. Additional conditions relate to the tasks processing times, possible machine and/or sequence-dependent setup times, no-overlapping requirements, and so on.

Szpigel pointed out that similarly in a RTSP each train/job requires the access to a set of single-track lines/machine and we can define different constraints depending on the network structure. However, the idea of modelling each track segment (rather than the entire line) as a single resource became the mostly used approach in literature in the following years (we refer to [46] and [135] for further and more detailed references).
In 2002, Mascis and Pacciarelli ([112] [113]) introduced a *disjunctive graph representation* for the JSP with blocking and no-wait constraints, that is a generalization of the disjunctive graph first introduced by [136]. According to a *blocking constraints* a job cannot move to the next machine until it is set free by the precedent job. On the other hand, a *no-wait constraint* ensures that, for each job, a task is performed immediately after the precedent one and without interruptions. Since then, the disjunctive graph formulation has been applied to a large number of applications of RTSP: [18, 40–44, 50–52, 59, 68, 69, 98, 108, 110] only to mention a few examples. The reason is that the disjunctive programming allows to suitably model the conflicts arising in railways systems and the blocking time theory defined by the UIC ([150]). In fact, while a blocking constraints guarantee the absence of conflicts and deadlocks in the timetable schedule (*no swap blocking constraints* - see [113]), no-wait constraints can help ensure that the minimum departure times are met and force trains to leave a blocking section as soon as possible. A disjunctive formulation can also be easily expressed into the form of a MILP and can be solved through the use of well-known mathematical programming techniques and commercial solvers. We will discuss this aspect

further later on this chapter (see Section 3.4.2), while in the next section we present the disjunctive graph theory more in detail.

3.4.1 Disjunctive Graph representation

Assuming that all trains routes are given, we can associate a *disjunctive graph* to each train scheduling instance. The disjunctive graphs were introduced in the context of production scheduling (see [6, 8, 136]), but in general they can be exploited to represent any problem in which we need to schedule a set of events ([26]).
A disjunctive graph is a triple $G = (N, E, D)$. The nodes in N correspond to the events that need to be scheduled (e.g. start of operations, landing times of airplanes, train occupation of track segments, project milestones, etc.). N also contains two special nodes o and n, called *start/origin* and *end/destination*, which represent respectively the event *start* and the *end* of a schedule. A schedule is a vector $\tau \in \mathbb{R}_+^N$ which associates a time to each event in N. All events are scheduled after o and before n and we may assume w.l.o.g. that $\tau_o = 0$. E is the set of directed fixed edges (or arcs) that impose the time precedence relations between events. In particular, a fixed edge $(u, v) \in E$ of length $l_{uv} \in \mathbb{R}_+$ implies that a constraint of the form $\tau_v - \tau_u \geq l_{uv}$ has to be satisfied. Since no events can start before the start o and after the end n, we have that:

i) $\tau_v - \tau_o \geq 0 \quad \text{for } v \in N$

ii) $\tau_n - \tau_v \geq 0 \quad \text{for } v \in N \quad .$

Moreover, we can set $l_{ov} = 0$ for all edges $(o, v) \in E$ with $v \in N \setminus \{o\}$ and $l_{vn} = 0$ for all edges $(v, n) \in E$ with $v \in N \setminus \{n\}$. Finally, each element $d \in D$ is called a *disjunctive arc*. It is an unordered pair of directed edges $d = \{(u, v), (q, m)\}$ and represents the alternative sequencing of four (not necessarily distinct) events u, v, q and m. An example is shown in Figure 3.5. Given a disjunctive arc d, any feasible schedule must satisfy either $\tau_v - \tau_u \geq l_{uv}$ or $\tau_m - \tau_q \geq l_{qm}$, with $l_{uv}, l_{qm} \in \mathbb{R}_+$. Now, said $G = (N, E, D)$ a disjunctive graph and l a distance vector, in order to find a feasible schedule for the corresponding scheduling problem, we have to decide for each disjunctive arc which of the two directed edges has to be "satisfied". A set S of directed edges is said to be a *complete selection* if it intersects exactly once all disjunctive couples. Therefore, for each pair of edges in D one and only one arc can be taken in a complete selection.
We say that a schedule is feasible if, (i) for each edge $(u, v) \in E$, we have that $\tau_v \geq \tau_u + l_{uv}$ and (ii) for each disjunctive pair $d = \{(u, v), (q, m)\} \in D$, we have either $\tau_v - \tau_u \geq l_{uv}$, or $\tau_m - \tau_q \geq l_{mq}$, with $l_{uv}, l_{mq} \in \mathbb{R}_+$. The following is a key result for the job shop scheduling (and for the graph theory in general - see Section 2.3.2):

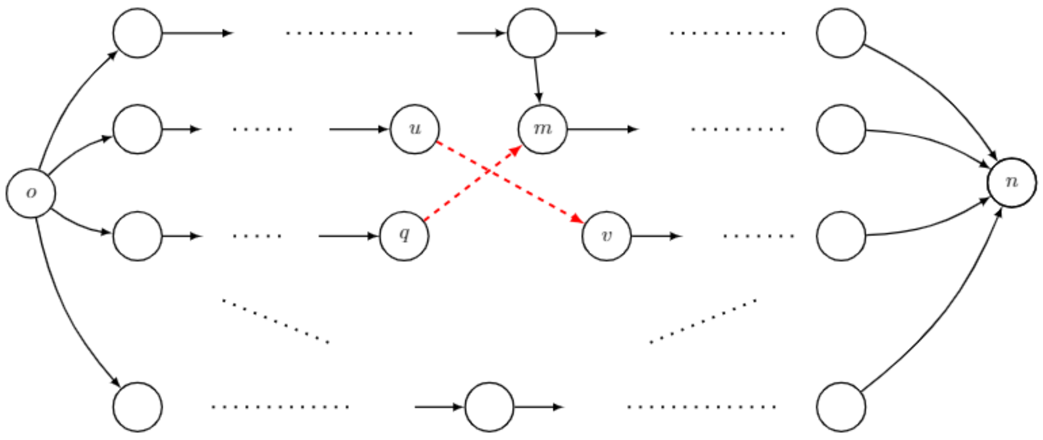


Figure 3.5: A disjunctive graph. The dashed red edges form a disjunctive pair of arcs.

Theorem 3.1. *An instance of a scheduling problem defined by a disjunctive graph $G = (N, E, D)$ admits a feasible schedule if and only if there exists a complete selection S such that the graph $G(S) = (N, E \cup S)$ does not contain a strictly positive directed cycle.*

Such a selection is defined as *consistent*. In particular, given a feasible schedule $\tau \in \mathbb{R}_+^N$, we can immediately derive a selection $S(\tau)$ such that $G(S)$ does not contain directed cycles, by letting, for each $\{(u, v), (q, m)\} \in D$, $(u, v) \in S$ if $\tau_u \geq \tau_v + l_{uv}$, and $(q, m) \in S$ otherwise. On the other hand, given a consistent selection S, one can easily derive a feasible schedule $\tau(S)$ by applying a longest path tree algorithm to the digraph $G(S)$ (see [127]).

Now, let u and v be two event belonging respectively to train i and j and requiring the access to the same track segment r, and $\sigma(u)$ be the subsequent operation of a generic event u. We say that there is a blocking constraint between u and v if u cannot be processed until resource r is set free by v and vice versa. We can represent this scenario with a pair of disjunctive arcs $(\sigma(v), u), (\sigma(u), v)$ with length equal to 0. The example is drawn in Figure 3.6, where the disjunctive pair of edges is represented in red and dashed, while solid black arrows represent fixed edges. Note that we set $l_{\sigma(u)v} = l_{\sigma(v)u} = 0$, however, when more than two trains require access to the same block section r, a small positive length ϵ has to be considered for the disjunctive arcs. This makes it impossible to select edges that create a positive cycle. Since we are considering edge lengths in $_+$, it is not hard to see that if $G = (N, E, D)$ is the disjunctive graph associated with an instance of train scheduling, given a selection S, any cycle in $G(S)$ will be strictly positive. So, the condition of Theorem 3.1 can be simply stated as $G(S) = (N, E \cup S)$ is acyclic, i.e. has no directed cycles. By extension, we say that a feasible S is an *acyclic selection* of G.

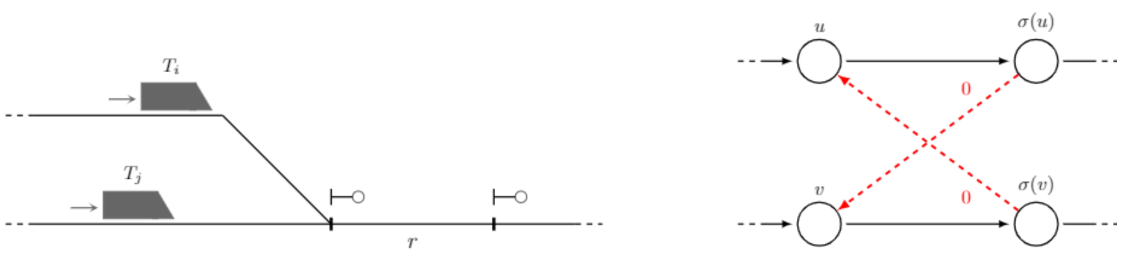


Figure 3.6: Example of a blocking constraint represented via disjunctive graph.

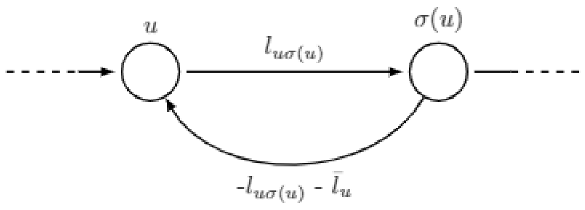
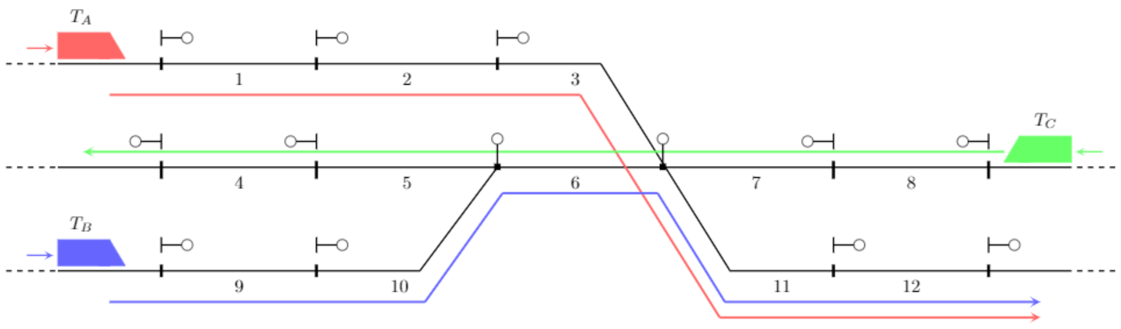


Figure 3.7: Example of a no-wait constraint.

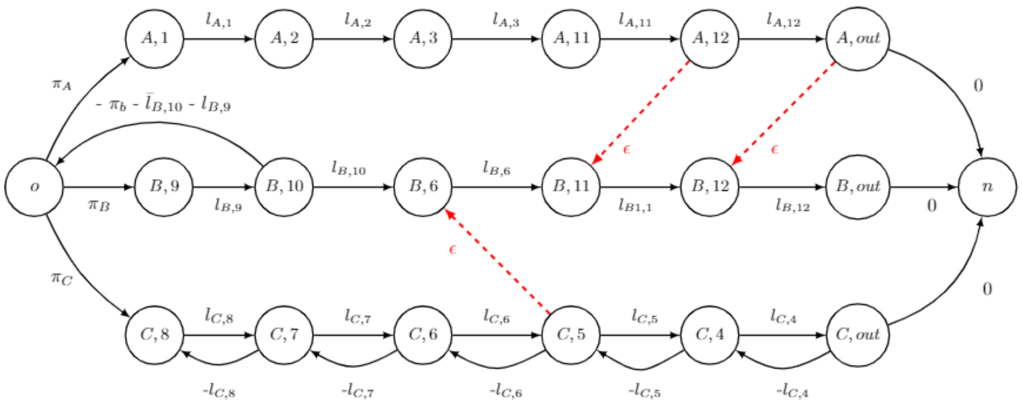
On the other hand, let $l_{u\sigma(u)}$ be the traversing time of a generic event u, and $\bar{l}_u$ be the maximum time that train i can wait before traversing the block segment $\sigma(u)$. A no-wait constraint (or *perishability* constraint) can be formulated through two fixed arcs $(u, \sigma(u))$ and $(\sigma(u), u)$ having respectively a length equal to $l_{u\sigma(u)}$ and $-l_{u\sigma(u)} - \bar{l}_u$ (see Figure 3.7). Naturally, when $\bar{l}_u$ equals 0 the train must proceed without interruption.

To determine the feasibility of a generic instance of a JSP is a $\mathcal{NP}$*-complete* problem (see Section 2.2.1), and this holds also for JSPs with blocking and no-wait constraints. Moreover, said $c : \mathbb{R}^N_+ \to \mathbb{R}$ a cost function, finding a feasible schedule τ that minimize $c(\tau)$ is also a difficult problem.

Now, let τ^* be the minimum cost schedule and $S^* = S(\tau^*)$ be the selection associated with τ^*. We recall that, for all events $u \in N$, τ^*_u equals the length of a longest path from the origin node o to u in $G(S^*)$. If c is monotone and non-decreasing, given a selection S, finding the associated feasible schedule $\tau^*(S)$ which minimizes c is computationally easy, as it corresponds to computing a longest path tree in $G(S)$. Then, given an instance of a train scheduling problem at a normal traffic regime (i.e. with no ongoing disruption), its



(a) A small railway network.



(b) A complete selection for the disjunctive graph associated with the rail network in Figure 3.8a.

Figure 3.8: Example of a small railway network (3.8a) and a complete selection for its disjunctive graph formulation (3.8b).

disjunctive graph formulation is presented below:

$$\begin{aligned} \min \quad & c(\tau) & & \\ s.t. \quad & \tau_u \geq \tau_v + l_{uv} & & (u,v) \in E \\ & (\tau_v - \tau_u \geq l_{uv}) \;\; \wedge \;\; (\tau_m - \tau_q \geq l_{qm}) & & \{(u,v),(q,m)\} \in D \;\; . \end{aligned} \tag{3.1}$$

In Figure 3.8 we show a portion of a railway network (3.8a) and a complete acyclic selection for its disjunctive graph formulation (3.8b). The network is divided in twelve block sections and traversed by three trains: A, B and C. All track segments have a unary capacity and are indicated in Figure 3.8 with increasing numbers from 1 to 12. Trains have fixed preassigned routes: $R_A = \{1,2,3,11,12\}$, $R_B = \{9,10,6,11,12\}$ and $R_C = \{8,7,6,5,4\}$.

Besides, train B has a deadline time to respect at station 10 (indicated by the parameter $\bar{l}_{B,10}$), while train C is a passenger train that must respect the planned running times of each track segment traversed, i.e. it cannot interrupt its run. In Figure 3.8b dashed red lines represent the disjunctive edges of the complete selection, while solid black lines denote fixed arcs. We refer to each event of the type "train i enters track segment r" with a node defined by the couple (i, r). Note that a scheduled start time π_i is given for each train i. Three dummy nodes (i, out) have been added in the disjunctive graph to represent the end of the trains last scheduled event and the edges connecting them to the end node n are said to have a zero length. Lastly, once defined a cost function, the quality of the feasible solution presented can be easily assessed through the calculation of the longest path between node o and n.

3.4.2 MILP formulations

Balas was the first to treat the JSP as a disjunctive program ([6]). He translated the disjunctive constraints of a single-machine scheduling problem into a linear program with logical conditions ([8]). In fact, the disjunctive problem (3.1) presented in the previous section can be converted with little effort into a (mixed linear) mathematical program. In general, one can classify the scheduling models and formulations according to the decision variables choices. In [94] four different Mixed Integer Program (MIP) formulations (using completion times variables [7], time-indexed variables [143], linear ordering variables [57] and assignment positional variables [131]) are presented and their computational performance compared. The work concerns the single-machine scheduling problems but all the formulations can be easily extended to the JSPs. Among all formulation, mainly two classes have been adopted in the literature to solve train re-scheduling problems (see [79], [100]): the big-M formulations, which use continuous-time variables, and Time-Indexed (TI) formulations, which discretize the time horizon.

Big-M formulation. The big-M formulation is a natural MILP formulation able to translate the disjunctive pair constraints. It associates to each event $u \in N$ a continuous starting time variables $t_u \in \mathbb{R}_+$ and uses a binary variables $y_{uvqm} \in \{0, 1\}$ to represent the alternative selections of arcs. Each variable y_{uvqm} models a pair of disjunctive edges $\{(u, v), (q, m)\} \in D$ and is set equal to one when the event u precedes v (arc (u, v) is selected), while is null when q precedes m (arc (q, m) is selected). This is ensured by introducing two constraints containing the notorious big-M term, that is a very large coefficient M, and is formulated as follows:

$$\begin{aligned} &(a)\;\; t_v - t_u + M(1 - y_{uvqm}) \geq l_{uv} \\ &(b)\;\; t_m - t_q + M y_{uvqm} \geq l_{qm} \quad . \end{aligned}$$

Note that $y_{uvqm} = 1$ implies that $t_v \geq t_u + l_{uv}$ and makes constraint *(b)* redundant. A mirrored scenario occurs for $y_{uvqm} = 0$, when the condition $t_m \geq t_q + l_{qm}$ holds and constraint *(a)* is always satisfied. Summarising, given a proper cost function c, we may write Problem (3.1) as:

$$\begin{array}{lll} \min & c(t) & \\ & & \\ s.t. & & \\ (i) & t_u \geq t_v + l_{uv} & (u,v) \in E \\ (ii) & t_v - t_u + M(1 - y_{uvqm}) \geq l_{uv} & \{(u,v),(q,m)\} \in D \\ (iii) & t_m - t_q + M y_{uvqm} \geq l_{qm} & \{(u,v),(q,m)\} \in D \\ & t \in \mathbb{R}_+^N \quad y \in \{0,1\}^{|D|} & \end{array} \tag{3.2}$$

The formulation is extremely versatile and can be easily modified to incorporate constraints of various types and different modelling requirements. As an example, it can be expanded to include re-outing options, adding a binary variable for each alternative route of a train. On the other hand, it is easy to see that such a formulation strongly depends on the value assumed by the coefficient M that can affect both the algorithm resolution speed and the solution quality. In [67] one can observe how the use of a simple heuristic preprocessing phase (to be coupled with a commercial MILP solver) can largely improve the resolution performances. Heuristics of various type, Branch&Bound methods, decomposition methods and hybrid approaches have been proposed over the years to bypass the disadvantages of such a direct mathematical rendering.

Time-indexed formulation. The idea of a TI formulation (from [129]) is to discretize the planning horizon into a number $\bar{p}$ of time-periods. Then, for each event u, a binary variable x_u^p refers to the period p and assumes value equal to 1 if operation u starts at time period p and 0 otherwise. Moreover, the disjunctive pair constraints are translated into *packing constraints* (see Definition 2.20) between *incompatible events.* Given two distinct time events u and v and two period p and p', we say that x_u^p and $x_v^{p'}$ are incompatible between each other if, depending on the length of the two events, u cannot start at p if v started at p'. In turn, this yield at a constraint of the type:

$$x_u^p + x_v^{p'} \leq 1. \tag{3.3}$$

thus, given a function $c : \bar{p} \times |N| \to \mathbb{R}$ that associates a cost to each interval-variable x, a TI formulation can be therefore expressed as follows:

$$\begin{array}{lll} \min & c(x) & \\ s.t. & & \\ (i) & \sum_{p=1}^{\bar{p}} x_u^p = 1 & u \in N \\ (ii) & x_u^p + x_v^{p'} \leq 1 & (u,v) \in I \\ & x \in \{0,1\}^{\bar{p} \times |N|} & \end{array} \tag{3.4}$$

where with I we indicate the set of incompatible events defined by both fixed and disjunctive edges in E and D. Note that constraints (i) ensure that each event starts exactly once and at a particular time p (non-preemption), while the packing constraints (ii) imply that only one of two incompatible time events can occur.
The event-incompatibility constraints can also be expressed in different ways, such as for the fixed precedence constraints stated by the arcs in E. We can indeed write:

$$\sum_{p=1}^{\bar{p}} p \cdot x_u^p \geq \sum_{s=1}^{\bar{p}} s \cdot x_v^s + l_{uv} \qquad \forall (u,v) \in E$$

Obviously, all the formulation choices influence the goodness of the solution reached. In addition, the spanned horizons and the interval widths heavily affect the formulation in terms of number of variables and constraints. Even small instances have a tendency to introduce a large number of variables and constraints that grows proportionally with the parameter $\bar{p}$, the number of potential conflicts and the events duration. For these reasons, the TI models are often coupled with heuristic approaches that compute the earliest and latest times of the events, allowing to reduce the time horizon to discretize, and accordingly the formulation size.

3.5 Looking beyond the classical formulations

A number of alternative formulations have been proposed in the literature to overcome the drawbacks of the classical models presented above. In addition, several heuristic approaches and hybrid methods have been studied. As for example, in [146] four different event-based models are compared to the base TI model presented for the resource-constrained project scheduling problem. The author establishes an order of domination among the formulations and summarizes which model best fits which type of instance. Another popular technique consists of decomposing the original MILP problem into smaller sub-problems and then obtain a solution by merging the partials one. The approach is known as *decomposition method* and can be applied to both the time and space dimensions. In the first case the method is also called *rolling horizon* and consists in solving chronologically the sub-problem derived from the partition of the time horizon into a smaller time intervals. Interesting works on the topic can be found in [65], [122], [116], [139] and [161]. On the other hand, a "space decomposition" generates smaller problems considering only a sub-area of the network or a subset of trains at the time (see [45] and [14] respectively). An example of application of the space decomposition approach is the definition of a different MILP model for each line segment of the network. This mechanism assumes that the obtained solutions can be embedded in a single global feasible solution for the complete problem. Recently, [147] proposed a coordination framework to merge the adjacent area rescheduling solutions obtained by a resource conflict graph model. A second space-based decomposition approach resorts to the principle of macro/microscopic decomposition. It is a two-step procedure that proposes the resolution of a macroscopic model in the first step and a check on the validity of the solution in the second step, when all microscopic elements of the model are included as well. Essentially, the second phase directs a search direction for the overall solution and reduces the search space of the first phase through a control loop. In [98] and [101] is instead presented an exact method exploiting the Benders' decomposition approach ([13]). besides, the recent [156] presents a traffic optimization model with fixed train routes that ensures the feasibility of the entire rail system when predicting the network state for a finite horizon. The paper tries to decompose the original model according to time, considering shorter horizons, and space, considering a subset of trains. Note that both (space and time) decomposition approaches track the actions taken by the dispatchers, who control a limited physical portion of the network and chronologically address conflicts as they occur.

Starting from a classical Benders' reformulation, in [109] the authors present a non-compact formulation, named *Path&Cycle* and derived from [99]. The model concerns the air traffic management but it can be easily extend to RTSPs and in general to JSPs. In particular, it handles the search for the optimal solution by strengthening and lifting of the original conflict constraints and looking for paths and cycles in the associated disjunctive graph (exploiting the theory of Theorem 3.1 and the longest path tree definition). The model, which allows us to get rid of the big-M term, proved to be very promising when compared to the classical formulation, especially in the case of non-convex cost functions. This is due to the fact that when used in a Branch&Bound approach, big-M formulations typically return poor bounds, which in turn leads to large branching trees. As mentioned earlier, TI formulations return in contrast better bounds and smaller trees. However, TI formulations have two main drawbacks depending on the choice of the granularity of the periods in the discretized horizon ([5, 131]):

1. *Oversize.* When periods are small and many, TI formulations typically contains a very large number variables and constraints. This slows down the solution time in each branching node by a factor which is larger than the reduction in the tree size. In the few comparative experiments available in the literature, the comparison is by far in favor of the big-M alternative (see [108]).

2. *Bad approximation.* Larger and fewer periods result in fewer variables and constraints, but poorer approximation. As consequence, solutions which appear feasible to the model, may not be feasible in practice ([22, 78]).

For the above reasons, there are indeed not too many examples in the literature of TI-formulations devoted to train scheduling. The majority of the application concerns the off-line aspect of the train scheduling problem, i.e. the timetabling creation process (see [29] for a survey) and only very few to dispatching, in most cases coupled with a time-expanded graphs representation.
Harrod was one of the first to apply the concept of hypergraphs[3] to conflicting resource and railway scheduling: in [78] he exploits the flexibility of this representation to model trains block occupancy and resource transitions. In [32] a discrete-time model considering the blocking time theory is proposed and the induced mathematical formulation (a constrained combinatorial assignment problem) analyzed. [138] develops a new integer formulation (based on space-time network) considering practical constraints for the train dispatching problem and designs an heuristic algorithms capable to find near-optimal solutions. Also Bettinelli et al. present in [14] a greedy heuristic algorithm applied to a time-space network able to solve in real-time (two seconds ca.) large instances (up to 150 trains and 2 hours planning time horizon). [117] addresses both train re-routing and re-scheduling by decomposing the original problem into a sequence of optimization sub-problems, one per train. For each problem, thy compute the shortest path in a time dependent graph, then a feasible solution to the original problem is obtained by gathering together all partial outputs. As [117], [133] focuses on re-routing and re-timing in real-time, however here a branch-and-price algorithm is used to find the optimal solutions in less then 30 seconds for the majority if the instances considered.
In any case, because the complete formulation would be too large to handle, most of the papers resort to delayed column generation and cut generation techniques (see [1, 54] and the

[3] An hypergraph is a graphical representation that allows to describe non-binary constraint networks, i.e. relations involving more than two objects.

insights in Section 2.2.4). The idea is to start with only a subset of the variables (and constraints), solve this restricted problem and then identify and add the missing variables and constraints before solving again. This procedure is iterated until some conditions are satisfied and, typically, lead to a "final" model that is much smaller than the full TI formulation.

Despite of the complexity and ingenuity of the most recent solution algorithms, TI formulations do not seem to perform better than a classical big-M formulation, solved with a standard commercial solver (see, for instance, the experiments over a large junction performed in [125] with an almost 10 year-old technology). On the other hand, it would be very relevant to be able to solve large instances of train scheduling with TI rather than big-M formulations. Indeed, TI formulations are more flexible than the big-M counterparts in expressing and manipulating complicated constraints and non-linear objectives.
In the present dissertation we will exploit and extent the Dynamic Discretization Discovery (DDD) technique recently introduced by Boland et al. ([20]) to keep at bay the growth of TI formulations. The paradigm can be applied to the RTSP and, more in general, to JSP (cfr. Chapter 4). In our approach, the intervals (and their associated variables) are generated dynamically by iteratively subdividing and redefining the previous partitions of the planning horizon. We show how the time-discretization can be chosen in such a way that the approximation is not worse than the one introduced by a time-continuous (big-M) formulation. We thus provide an exact algorithm to solve the problem under analysis.

Chapter 4

Dynamic Discretization Discovery for the Train Traffic Re-scheduling Problem[†]

4.1 Introduction

Train scheduling is a critical activity in rail traffic management, both off-line (timetabling) and on-line (dispatching). In the on-line/real-time field, rapid and responsive reactions are required to ensure an adequate and consistent level of service in the rail system. Deviations from planned and scheduled activities must be addressed in a very short time by dispatchers and traffic controllers. The Train Traffic Re-scheduling Problem (TTRP) is the facet of the Real-time Train re-Scheduling Problem (RTSP) that has to do with the on-line re-timing and re-ordering of the trains. The goal is to establish new scheduled times for the train plans affected by an unexpected event, whether a disturbance or a disruption - (see chapter 3 for more insight), making the off-line timetable out of date and unfeasible.

There are of course different approaches to tackle the traffic re-scheduling, here we are interested in those based on Mixed Integer Linear Program (MILP), which are the most adopted in the literature. The main issue that such approaches must face is how to represent the fact

[†] This chapter is an extension of a work presented at RailBeijing 2021, the 9th International Conference on Railway Operations Modelling and Analysis (ICROMA). The original work is co-authored with Prof. Carlo Mannino, from SINTEF (Oslo, Norway) and PhD Paolo Ventura from IASI at CNR (Rome, Italy). [A. L. Croella, C. Mannino and P. Ventura. Dynamic Discretization Discovery for the train scheduling problem. RailBeijing 2021. (in conference proceeding)]

that two trains cannot occupy simultaneously the same track (or other pairs of incompatible railway resources). Then, in any feasible schedule, one of the conflicting trains must use the contended resource before the other train occupies it, and this gives rise to a disjunctive constraint on the scheduling variables. Big-M formulations annd *Time-Indexed* formulations, with both their strengths and disadvantages, are the MILP classes mainly adopted in the literature. Time-Indexed (TI) formulations have proven to be stronger than the classical and oldest big-M formulation. Moreover, they can be easily adapted to cope with complex requirements and non-linear objective functions. Unfortunately, their size grows usually very large with the size of the scheduling instance. As a consequence, formulations are typically not fit to attack real-life instances of job-shop scheduling problems, especially when fast, online responses are required. In addiction, the approximation introduced by time discretization often leads to solutions which cannot be realized in practice ([21]). For a more in depth discussion we refer the reader to Section 3.4.2 and Section 3.5.

Either way, as pointed out by Boland in [20], network design problems often rely on a discrete horizon representation. It is indeed extremely tempting to exploit the flexibility and tight bound produced by TI models in solving the TTRP. In the remain of the chapter we present the application of the Dynamic Discretization Discovery (DDD) paradigm to the railway scheduling framework. The DDD technique, recently introduced by Boland et al. for the continuous-time Service Network Design Problem (SNDP)[1] and the Travel Salesman Problem with time windows (TSPTW)[2] (see [22] for an extensive survey), aims to reduce the TI over-sizing and response times while, at the same time, ensuring the necessary modelling precision. The main idea is to consider, for each train i, a partition of the time horizon into periods $\lambda_1, \dots, \lambda_{n_i}$ of different sizes. Then a train i can enter resource k at some time $t^{ik} \in \lambda_p = [h, h^+)$ of the p-th train time period (not necessarily at the beginning of the period, as in the standard TI formulation). As a consequence, a packing constraint (see Definition 2.20) is introduced between two time-events if and only if, for any choice of $t^{ir} \in \lambda_p$ and $t^{jr} \in \lambda_{p'}$, two trains i and j are in conflict on a shared resource r. For each event the set of time periods is generated dynamically, as suggested by the name of the method, and iteratively starting from the previous partition of the planning horizon. We can also associate a graph for each iteration of the method: the nodes represent the partitions intervals, while the edges translate the packing constraints between conflicting resource time-assignments. Then the problem solved can be also seen as a *maximum weight stable set problem.*

[1] The Service Network Design Problem aims at finding an optimal allocation and utilization of resources for the tactical planning process of freight transportation.

[2] The Travel Salesman Problem is the problem of finding the most efficient route that visits all the distinct locations in a list exactly once and ends at the original departure point.

The DDDs specialize in the way intervals are recursively generated. We show how our formulation can return a lower bound on the weight of the optimal solution. At termination, the solution returned by last iteration is indeed feasible for the original problem, and thus optimal. Although the idea of a dynamic discovery is fairly natural, there are many possible ways to implement it and the engineering the algorithm is one of the most delicate phases of its deployment. In the present work, we start to design its effective and efficient implementation and lay the foundation for its application in the train scheduling context.

The chapter is organized as follows: the Section 4.2 gives a review of the related works on the newly proposed DDD technique and on the methods dealing with the poor approximation and oversize drawbacks of the TI models. Section 4.3 then formally describes the TTRP and the notation used. Section 4.4 and Section 4.5 respectively define its formulation through the use of the DDD paradigm and show an efficient implementation to solve it. Section 4.6 hence presents the application of the designed algorithm with a simple illustrative example. Computational results on a set of realistic instances are discussed in Section 4.7. Here extensive comparisons with the big-M and TI formulations performances are presented. We must anticipate that the average verdict is still in favor of the big-M formulation. However, in many cases the computation times are comparable, which is a remarkable achievement given previous experience. Concluding remarks in Section 4.8 finally indicate future research directions to fill the residual gap.

4.2 Literature overview

The DDD is a novel technique to solve TI-formulations recently introduced in [20] and then further extended in [81–83, 97, 111, 140, 155]. It was developed to cope with the (previously discussed) size and approximation issues of the time-dependent models and it proposes to create a TI model with both a fine (hence near-optimal) discretization and a limited number of variables and constraints. This is done thanks to:

- the construction of (small) partially refined models that are easy to solve and that provide either a lower bound, an upper bound or an exact solution,
- a dynamic discover mechanism able to quickly and efficiently refine the partial discretizations in order to produce a "better" model.

A generalized scheme of the paradigm DDD can be seen in Figure 4.1. Observe that when the partial model provides a lower bound for the original problem, the refinement of the time

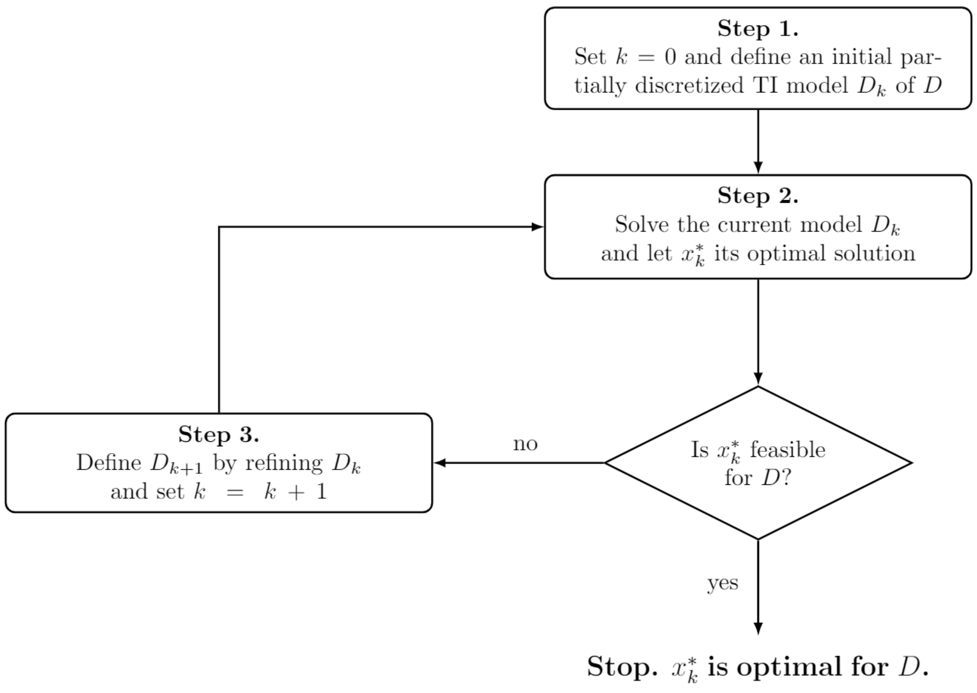

Step 1.
Set $k = 0$ and define an initial partially discretized TI model D_k of D
Step 2.
Solve the current model D_k and let x_k^* its optimal solution
Is x_k^* feasible for D?
no
Step 3.
Define D_{k+1} by refining D_k and set $k = k + 1$
yes
Stop. x_k^* **is optimal for** D.

Figure 4.1: Generalized flowchart of the DDD paradigm.

discretization performed at step 3 makes the current solution infeasible for the newly generated partial model. Additional mechanisms can be added to the method, such as those repairing a currently infeasible solution. Also, one can make use of models that provide both an upper and a lower bound for the optimal solution. We remark that a number of parameters, ranging from the way the new refinements are generated to the efficiency of the algorithm used to solve the partial models, affects the implementation of the DDD paradigm.

Boland first uses the DDD technique to tackle the continuous-time SNDP ([20]). The method proposed is based on the partitioning the time-defined windows into sub-windows, as introduced by Wang and Regan in [157], and exploits the fact that the more refined the intervals are, the closer the solution is to the optimal one ([158]). Of course, in the extreme case, a complete discretization (a full TI model) leads to the optimal continuous-time solution. Though, in the paper the explicit modelling of all the time points is avoided by selecting only a few specific time points to be added by the algorithm at each iteration. In this way, the size of the model is kept at bay making the TI version of the problem tractable in practice.
An enhancement of the initial intuition on the use of the DDD paradigm for the service network design problem is shown in [111]. Here the concept of interval-based variables is introduced: a variable is associated with an event occurring at some time along an interval $[h, h^+)$ (with $h^+ > h$) of the discretized horizon rather than being associated with a specific time point belonging to it. The authors show that this *interval-view* approach proves effective for the problem under analysis, leading to the construction of final models with dimensions smaller than 1% of the corresponding full TI formulation.
Likewise, in [81] He et al. leverage on the interval concept to solve the Shortest Path Problem (SPP) with piece-wise linear arc travel times. Both lower bound and upper bound solutions (the seconds obtained by repairing the lower bound ones) are used to iteratively generate the partial models. In addition, the algorithm gives the possibility to drop some of the previously generated intervals to improve the bounds. Computational experiments indicate that the benefit of the considered approach increases as the size of the instances grows.
Vu et al. instead apply in [155] the DDD framework to the TSPTW, starting from the time-expanded method proposed in [53]. The technique is applied to a scenario with time-dependent travel times and shows promising results also with regard to the robustness feature of the algorithm. We highlight that the approach is superior to the best known algorithms and methods presented in the literature in the related field.
Hewitt in [83] proposes instead an improved version of the paradigm for the continuous time

load plan design problem. In the paper is presented a two-stage DDD that firstly solves the linear relaxation of the problem and then uses the time-expanded representation obtained to initialize the discretization discovery. Four additional enhancements attempt to reduce the number of partial integer problems to be solved (i.e. the number of iterations) and/or the time required to solve them.
The service network problem is also addressed in [140] where two revised version of the DDD algorithm implemented in [83] are proposed. Here, the resolution of the intermediate models is improved through the use of ad hoc heuristics and a set of valid inequalities that are iteratively added in the formulations.
Finally, [82] shows an application of the technique for a service network design and routing problem in a logistics hub network, while [97] applies the DDD concepts to the continuous time inventory routing problem proving the high quality of the solutions achieved through the algorithm.

A second trend of research dedicated to overcoming the pitfalls of TI is embodied in the creation of exact integer problems based on partially discretized times. In this second research stream, the time periods of the planning horizon are referred as *buckets* and, unlikely the classical TI model, can have different lengths. Obviously, the lengths of the buckets influence the size of the formulation and consequently its resolution times. It is easy to see that a full TI model is a particular case of bucket discretization with a fixed unary interval length. Firstly applied in the lot sizing context, the concept of bucket has also been applied to non-preemptive single machine scheduling problems (see [11] and [19]). The resulting models have been shown to outperform their naive TI counterparts, presenting good performance even when considering large instances.
Besides, the idea of solving the Linear Program (LP) relaxation of the full TI formulations represents an alternative search path. This can be done through the exploitation of the Lagrangian relaxation model and/or its dual problem combined with branch-and-price techniques. In particular, [75] introduces a new idea to recursively refine a time-expanded graph representation. In this work the terminology *iterative refinement* (adopted from [134]) is used in contrast to the term *dynamic discretization discovery* to highlight the fact that the procedure goes beyond the discretization of the intervals and extends also to the branch&bound exploration when solving a relaxation of the TI formulation, or in general a MILP. The resulting algorithm, named *branch-and-refine*, is tested for both the SPP and the TSPTW.
A different branch-and price method is instead proposed by Lusby et al. ([104, 105]) for the real-time version of the problem of routing the trains through railway junctions. They

propose a flexible set packing formulation (that can be also generated dynamically using variable generation) that aims to minimize the delay propagation, while nevertheless placing more value on feasibility than optimality. Trains are given the possibility of varying their speed and the output is a set of conflict-free paths traversing the junctions and respecting the signalling system and other additional constraints.

To the best of our knowledge, the dynamic generation of time-discretized networks in the context of train scheduling has been studied also by Fisher et al.. In [66] the authors design a dynamic graph generation procedure able to efficiently and rapidly compute the shortest path on a truncated (hence restricted in size) graph. The modelling approach is here applied to the off-line version of the train scheduling problem, i.e. the train timetabling, and tested successfully on a large scale instances.
On the other hand, Ostrowski et al. recently showed in [12] the assumptions under which the linear relaxation of the non-periodic train scheduling problem is able to provide an integer solution. They propose a new formulation with discretized time intervals based on the notions of multi-commodity flow and set-packing (similar to the one exposed later in the present chapter). The decomposition of the basic problem into smaller problems then allows for low resolution times and easy scalability.

4.3 Problem statement and notation

We applied the DDD paradigm to the TTRP. The problem can be seen as a particular case of the RTSP (see Definition 3.1) where we do not consider the possibility of rerouting trains. Its formal definition is stated below.

Definition 4.1. *Given a network and its current state at time $\underline{t}$, a set of trains, their ordered list of movements along with the planned arrival and departure times in and from each track segment, find a new feasible schedule by re-timing and re-ordering trains according to their position at time $\underline{t}$, while minimizing a measure of the deviation from the original timetable.*

We remark that, when given a reference timetable (the public one) and the position of the trains at the current time, trains may be delayed with respect to the original plan. Depending on the next scheduling decisions, some trains may reduce their delays at destination, while others may increase it. The target is to schedule trains for the next few hours so to minimize a function of the delays. We next introduce the main modelling choices adopted and the necessary formalism for describing our model. Note that, although a more accurate representation of the network is necessary for the practical implementation of such

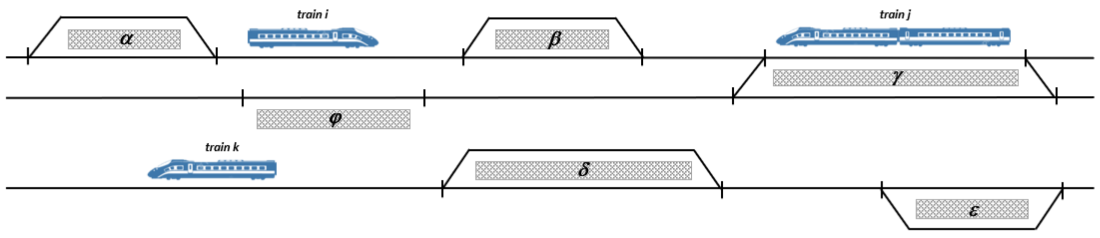


Figure 4.2: Schematic representation of a railway infrastructure. Stations are depicted as filled rectangles.

a method, a higher level of detail is not crucial in understanding the theory and can be disregarded. For a more rigorous definition of the elements of the railway system we refer the reader to Section 3.3.

Networks, trains and routes. In Figure 4.2 we show a schematic section of a railway infrastructure composed of a number of sub-networks, called lines. Each line consists of stations and tracks, connecting the station pair-wise. Tracks can also be divided in sub-tracks, referred as *tracks segments* or *block sections*, that can accommodate a train at the time. Particular type of track segments, called *sidings*, allow the trains to meet or bypass each other. We can define a set R of all track segments that make up the network and that can be traversed by trains either in one direction (*unidirectional track*) or in both directions (*bidirectional track*). In the example, train i travels from east to west reaching station β, while train j is departing from station from station γ running to east. Note that the latter, starting from its position can reach both station β and station φ. Suppose next station visited along its run is β, train i and j are said to be *crossing* (or *opposite*) *trains*, since they visit station α and β in opposite order. On the contrary, trains running on the same direction are instead called *followers trains.*
In the TTRP we are given a set of trains I and we assume they have fixed routes. Each train route is a rail path, a list of atomic movements that can be defined as follows:

Definition 4.2. *A* rail path *is a sequence of contiguous track segments of the line, ordered either west-bound or east-bound.*

For each train $i \in I$, we let R_i be the ordered set of track segments that i crosses from origin to destination. We assume, without loss of generality, that each segment is traversed at most once. We therefore have that $R = \cup_{i \in I} R_i$. Moreover, for $r, q \in R_i$, we use the notation $r \prec q$ if r precedes q in the route of i.

For $r \in R_i$, we also let $l^{ir} \in \mathbb{Z}_+$ be the amount of time i needs to cross track segment r. Trains may also stop when at a station in order to load and/or unload passengers and/or freights. In general, we include the value of this *wait* or *dwell* time in the amount l^{ir} of the track segment r ending at that station. Furthermore, for each train i and for each segment r of its route, either because the reference timetable establishes the wanted departure time for every station, or because i is delayed, we can calculate a value $\underline{t}^{ir}$, that is the earliest time when the train can enter the block section (i.e. *expected arrival time*).

Conflicts and schedules. We say that there is a *conflict* between two track sections $r, q \in R$ if they cannot be occupied at the same time by two different trains. Although we do not model trains length explicitly, trains may for example occupy more than one track at a time. Also, two track segments r and q can share some piece of physical track and be in conflict between each other. We let $\mathcal{D}$ denote the set of all couples of conflicting/non-shareable track sections and, for each r in R, we let $\{r, r\} \in \mathcal{D}$. Besides, for each $\{r, q\} \in \mathcal{D}$ with $r \in R_i$ and $q \in R_j$, $i \neq j$, we denote by $l^{ij}_{rq} \in \mathbb{Z}_+$ the amount of time j has to wait before using q, once i entered r (such amount is usually equal to l^{ir} plus some safety extra time that depends on j and q).
Now, said t^{ir} the time in which train i enters a track $r \in R_i$, we define a *schedule*, i.e. a vector specifying the timing of all planned trains movements, as $\tau = \{t^{ir} \mid i \in I, r \in R_i\}$. Note that trains cannot leave a station before the official departure time. Then, we say that a schedule is *feasible* if:

i) each train i occupies the track segments of R_i according to the prescribed order;

ii) for each train i and track $r \in R_i$, the earliest time of arrival is respected;

iii) for each conflicting couple $r, q \in R$ with $r \in R_i$ and $q \in R_j$, either i enters r after j has freed q, or j enters q after i has freed r.

More formally:

Definition 4.3. *A schedule τ is said to be feasible if the following requirements hold:*

i) $t^{ir} \geq \underline{t}^{ir}$*, for each* $i \in I$*, and* $r \in R_i$*;*

ii) $t^{iq} \geq t^{ir} + l^{ir}$*, for each* $i \in I$*, and* $r, q \in R_i$ *with* $r \prec q$*;*

iii) either $t^{jq} \geq t^{ir} + l^{ij}_{rq}$ *or* $t^{ir} \geq t^{jq} + l^{ji}_{qr}$*, for each distinct* $i, j \in I$*,* $i \neq j$*, and* $r \in R_i$*,* $q \in R_j$ *with* $\{r, q\} \in \mathcal{D}$*.*

We will refer to the requirements of type *i*) as *departure constraints*, to those of type *ii*) as *route constraints*, and to constraints *iii*) as *disjunctive constraints.*

Objective function. The *original timetable* is the schedule planned in advance at the strategic stage of the train scheduling process, while the *real-time timetable* is the one that occurs in practice during the implementation of the first. The deviation between them can be weighted according to a properly defined function. Thus, for each event we define a weight that translates the train priority and/or the occupied resource relevance. In the following, we assume the weight function $w(\tau)$ to be non-decreasing and separable, namely $w(\tau) = \sum_{i \in I} \sum_{r \in R_i} w^{ir}(t^{ir})$. However, different objective functions can also be considered. In the TTRP we want to find a feasible schedule τ that minimizes the objective function. Observe, that, we can always assume that, for all components t^{ir} of τ, we have that $0 \leq t^{ir} \leq M$, for some finite coefficient $M \in \mathbb{Z}$. Indeed this is the basic assumption for all big-M formulations (see Section 3.4.2).

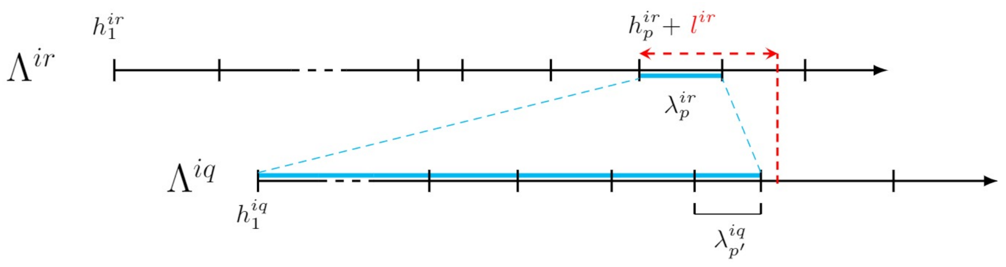
4.4 The Dynamic Discretization Discovery problem

In this section, we present a new combinatorial problem whose optimal solution defines a lower bound for the optimal solution of the TTRP.

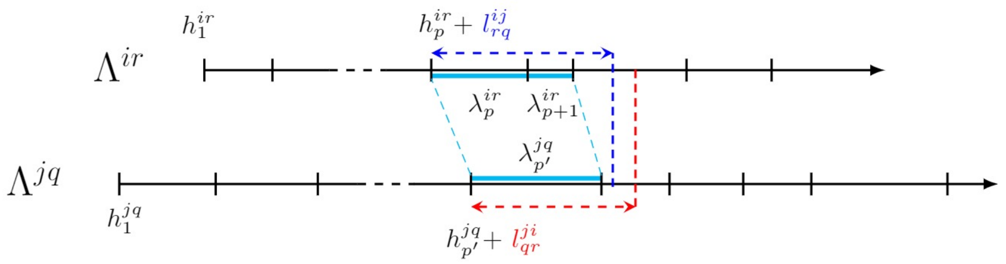
We start by associating, with each train $i \in I$ and each track section $r \in R_i$, a partition $\Lambda^{ir} = \{\lambda_1^{ir}, \lambda_2^{ir}, \ldots, \lambda_{n_{ir}}^{ir}\}$ of the time interval $[\underline{t}^{ir}, M)$, where we indicate with n_{ir} the index of the last block λ. In particular, let $\lambda_p^{ir} = [h_p^{ir}, h_{p+1}^{ir})$, with $h_p^{ir} < h_{p+1}^{ir}$, for $1 \leq p < n_{ir}$, where $h_1^{ir} = \underline{t}^{ir}$, and $\lambda_{n_{ir}}^{ir} = [h_{n_{ir}}^{ir}, M)$. Observe that the *departure constraints* imposed by condition $i)$ of Definition 4.3 are easily satisfied thanks to the adoption of $h_1^{ir} = \underline{t}^{ir}$ for each $i \in I$ and $r \in R_i$. We use Λ to denote the set of all partitions, for all $i \in I$ and $r \in R_i$. Also, we let $\bar{w}(\lambda_p^{ir}) = w(h_p^{ir})$ be the weight associated with the interval λ_p^{ir}.
Now, let $i \in I$ and $r, q \in R_i$ with $r \prec q$. We say that the intervals λ_p^{ir} and $\lambda_{p'}^{iq}$ are *route incompatible* if condition $ii)$ of Definition 4.3 is violated for any $t^{ir} \in \lambda_p^{ir}$ and any $t^{iq} \in \lambda_{p'}^{iq}$. Therefore, we have that:

Definition 4.4. *Given two intervals $\lambda_p^{ir} = [h_p^{ir}, h_{p+1}^{ir})$ and $\lambda_{p'}^{iq} = [h_{p'}^{iq}, h_{p'+1}^{iq})$, with $r, q \in R_i$ and $r \prec q$, they are said* route incompatible *if and only if $h_p^{ir} + l^{ir} > h_{p'+1}^{iq}$.*

In other words, if there is no way for train i to enter r during interval λ_p^{ir} and, later on, to enter q during time interval $\lambda_{p'}^{iq}$. Furthermore, let $i, j \in I$ with $i \neq j$, $r \in R_i$, and $q \in R_j$. We say that the intervals λ_p^{ir} and $\lambda_{p'}^{jq}$ are *conflict incompatible* if condition $iii)$ of Definition 4.3 is violated for any $t^{ir} \in \lambda_p^{ir}$ and any $t^{jq} \in \lambda_{p'}^{jq}$. Namely, train i cannot enter r during interval λ_p^{ir} together with j entering q during time interval $\lambda_{p'}^{jq}$. We have that:



(a) Example of *route incompatibility* between two track segments traversed by train i and such that $r \prec q$.



(b) Example of *conflict incompatibility* between two distinct trains i and j traversing respectively track segments r and q.

Figure 4.3: Example of intervals that are *route incompatible* (4.3a) and *conflict incompatible* (4.3b).

Definition 4.5. *Given two intervals $\lambda_p^{ir} = [h_p^{ir}, h_{p+1}^{ir})$ and $\lambda_{p'}^{jq} = [h_{p'}^{jq}, h_{p'+1}^{jq})$, with $i \neq j$, $r \in R_i$ and $q \in R_j$, they are said* conflict incompatible *if and only if $h_{p+1}^{ir} < h_{p'}^{jq} + l_{qr}^{ji}$ and $h_{p'+1}^{jq} < h_p^{ir} + l_{rq}^{ij}$.*

Figure 4.3 shows an example for the route incompatibility and one for the conflict incompatibility. In Figure 4.3a we can see how all blocks going from λ_1^{iq} to $\lambda_{p'}^{iq}$ are route incompatible with λ_p^{ir}, while in Figure 4.3b the interval $\lambda_{p'}^{jq}$ has a conflict incompatibility with the intervals λ_p^{ir} and λ_{p+1}^{ir}. Moreover, we observe that, for both route incompatibility and conflict incompatibility, the following holds.

Remark 4.1. *Said λ, λ' two incompatible intervals, let $\bar{\lambda}, \bar{\lambda}'$ be two intervals such that $\bar{\lambda} \subseteq \lambda$ and $\bar{\lambda}' \subseteq \lambda'$, then $\bar{\lambda}$ and $\bar{\lambda}'$ are incompatible interval.*

Now we can formally define the set of intervals that we are interested in.

Definition 4.6. *Given a set of partitions Λ, a set $S \subseteq \Lambda$ of (distinct) intervals is said to be* Λ-feasible *if*

i) S contains exactly one interval for each Λ^{ir}, with $i \in I$ and $r \in R_i$;

ii) all intervals of S are pairwise not incompatible.

Then, we introduce the function Ψ that assigns to each value t^{ir} the unique interval $\Psi(t^{ir}) = \lambda_p^{ir} \in \Lambda^{ir}$ such that $t^{ir} \in \lambda_p^{ir}$. We extend the notation by denoting with $\Psi(\tau)$ the set of (distinct) intervals $\{\Psi(t^{ir}) \mid i \in I, r \in R_i\}$. Now, given a set of partitions Λ, we can define the DDD problem as the problem of finding the Λ-feasible set of intervals of minimum weight. We now prove the following:

Lemma 4.1. *The value of the optimal solution to the DDD problem defines a lower bound for the value of the optimal solution to the corresponding TTRP.*

Proof. Let τ be a feasible schedule. Then $\Psi(\tau)$ is a feasible set of intervals. Moreover, since the objective function w is assumed to be non-decreasing, we have that $\bar{w}(\Psi(\tau)) := \sum_{i\in I}\sum_{r\in R_i} \bar{w}(\Psi(t^{ir})) \leq \sum_{i\in I}\sum_{r\in R_i} w(t^{ir}) = w(\tau)$. □

Let Φ be the function that assigns to each time interval $[h, h^+)$ the time instant $t = h$. Moreover, we extend such definition to a set of interval $S \subseteq \Lambda$, so to let $\tau = \Phi(S)$ be the schedule $\{t^{ir} = h_p^{ir} \mid \lambda_p^{ir} = [h_p^{ir}, h_{p+1}^{ir}) \in S\}$. Therefore, because of Lemma 4.1, if S^* is a Λ-feasible set of intervals of minimum weight, with respect to any given partition Λ, and $\tau^* = \Phi(S^*)$ is feasible, then τ^* is also optimal for the TTRP.

4.4.1 An ILP formulation for the DDD problem

We now present an Integer Linear Program (ILP) for the DDD problem.
For each $i \in I$ and $r \in R_i$, we are given a partition Λ^{ir} of the interval $[\underline{t}^{ir}, M)$. Then, we introduce the following set of binary variables:

$$x_p^{ir} = \begin{cases} 1 & \text{if the interval } \lambda_p^{ir} \text{ is taken} \\ 0 & \text{otherwise} \end{cases} \qquad i \in I,\ r \in R_i,\ \lambda_p^{ir} \in \Lambda^{ir}\ .$$

Hence, the DDD problem can be formulated as follows:

$$\begin{array}{lll} \min & \sum_{i\in I}\sum_{r\in R_i}\sum_{\lambda_p^{ir}\in\Lambda^{ir}} \bar{w}(\lambda_p^{ir}) \cdot x_p^{ir} & \\ \textit{s.t.} & & \\ \textit{(1)} & \sum_{\lambda_p^{ir}\in\Lambda^{ir}} x_p^{ir} = 1, & i \in I,\ r \in R_i \\ \textit{(2)} & x_p^{ir} + x_{p'}^{jq} \leq 1, & \lambda_p^{ir}, \lambda_{p'}^{jq} \in \Lambda \text{ with } \lambda_p^{ir} \text{ and } \lambda_{p'}^{jq} \text{ incompatible} \\ & x_p^{ir} \in \{0,1\} & i \in I,\ r \in R_i,\ \lambda_p^{ir} \in \Lambda^{ir}\ . \end{array} \tag{4.1}$$

One can easily see that constraints *(1)* and *(2)* implement conditions *i*) and *ii*) of Definition 4.6, respectively. Moreover, observe that the proposed formulation reduces the DDD problem to a particular packing problem where one has to minimize the sum of the weights associated with the selected elements subject to some *tight* packing constraints (constraints *(1)*). Thus it can be seen as a Maximum Weight Stable Set Problem (MWSSP) (simply inverting the sign of the weight function). We will focus on this aspect further in the next Section. Also observe that if all the intervals of Λ have length one, then the formulation (4.1) does coincide with a standard time indexed formulation for the TTRP (see formulation (3.4) in Section 3.4.2). Therefore, the following holds:

Observation 4.1. *If a set of partition Λ is such that $h^{ir}_{p+1} = h^{ir}_p + 1$, for all $i \in I$, $r \in R_i$, and $p = \{1, \dots, n_{ir}-1\}$, then the value of the optimal solution of the DDD problem coincides with the value of the optimal solution of the corresponding TTRP.*

4.5 The algorithmic framework

In the following, we describe our implementation of the Step 1, Step 2, and Step 3 of the DDD paradigm, exploiting the definition of the DDD problem that we gave in the previous section. We propose an algorithm, named DDD-ALG, that iteratively solves a TTRP instance D_k defined on a proper partition set $\Lambda_k = \{\Lambda^{ir}_k | \; i \in I, r \in R_i\}$. Then, we prove that the DDD-ALG terminates after a finite number of steps, returning the optimal solution of the original TTRP.
Note that, for each iteration k, the set Λ_k is always *more refined* with respect to the partition set Λ_{k-1} defined at the previous iteration. Moreover, as we pointed out in Section 4.4.1 the DDD problem can be seen as a particular MWSSP. It follows that we can immediately associate with each set of partitions Λ_k a *DDD graph* $G_k(V_k, E_k)$. In particular, we have that $V_k := \Lambda_k$, with $\Lambda^{ir}_k = \{\lambda^{ir}_1, \lambda^{ir}_2, \dots, \lambda^{ir}_{n_{ir}}\}$ for each train $i \in I$ and each track segment $r \in R_i$, while the set E_k contains an edge of the type $\{\lambda^{ir}_p, \lambda^{jq}_s\}$ for each incompatible couple of intervals in Λ_k (with i, j and r,q not necessarily distinct).

4.5.1 Initialize the DDD-ALG (Step 1)

We define the initial model D_0 of the DDD problem by just considering, for each resource used by each train, a single interval of the type $\lambda^{ir}_1 = [\underline{t}^{ir}, M)$. Thus, we set $\Lambda_0 = \{\Lambda^{ir}_k = \{\lambda^{ir}_1\} \mid i \in I, r \in R_i\}$.

4.5.2 Solve the DDD problem (Step 2)

Observe that, for all $i \in I$ and $r \in R_i$, the set of nodes of $\Lambda^{ir}_k \in \Lambda_k$ defines a clique in the graph G_k. We denote such cliques as *resource assignment cliques*. One can also identify a second type of clique: given two consecutive track segments r and q traversed by a train i such that $r \prec q$, for each λ_p^{ir} in the considered partition Λ_k, we can write a single *fixed precedence clique* constraint defined by the route incompatibilities. As consequence, formulation (4.1) can be strengthened, by substituting, for each clique C of G_k, the inequalities of type $x_p^{ir} + x_{p'}^{jq} \leq 1$ of $\lambda_p^{ir},\ \lambda_{p'}^{jq} \in C$, with an equivalent clique inequality $\sum_{\lambda_p^{ir} \in C} x_p^{ir} \leq 1$ (see [141]). We have that:

$$x_p^{ir} \ + \ \textstyle\sum_{\lambda_{p'}^{iq} \mid h_{p'}^{iq} < h_p^{ir} + l^{ir}} x_{p'}^{iq} \leq 1 \qquad i \in I,\ r, q \text{ consecutive} \in R_i \mid r \prec q,\ \lambda_p^{ir} \in \Lambda_k^{ip}\ . \tag{4.2}$$

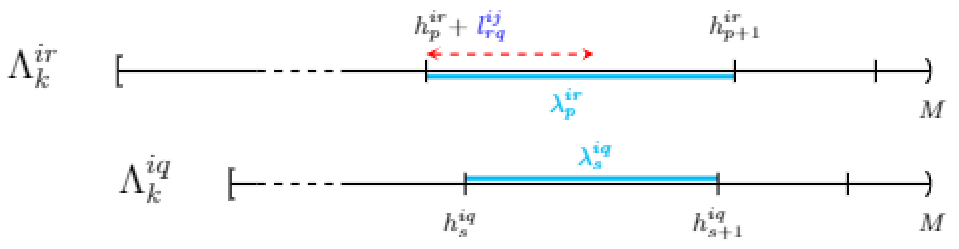
One can visualize an example of fixed precedence clique in Figure 4.3a. Here all intervals of Λ^{iq} highlighted in cyan belong to the clique defined by λ_p^{ir}.
Resuming, the DDD problem associated with Λ_k (and G_k), and solved at Step 2 of the DDD-ALG, reduces to find a minimum weighted stable set S_k^* of G_k that intersects:

- each resource assignment clique of G_k exactly once,
- each fixed precedence clique at least once,
- and each incompatible couple of intervals at least once.

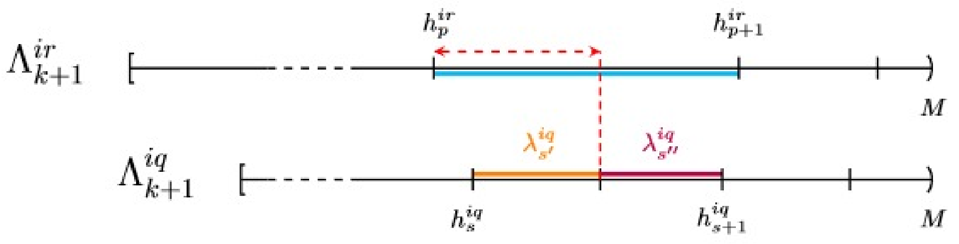
4.5.3 Refine the DDD problem (Step 3)

Let Λ_k be the set of partitions defined at iteration k of DDD-ALG, G_k be the corresponding graph, and let S_k^* be the optimal stable set of G_k (i.e. the optimal solution of the DDD problem D_k associated with Λ_k). Now, as already noticed, if $\tau_k^* = \Phi(S_k^*)$ is a feasible schedule than it defines an optimal solution to the TTRP and we are done. So, assume τ_k^* is not feasible. This means that S_k^* contains two intervals, say $\lambda_p^{ir} = [h_p^{ir}, h_{p+1}^{ir})$ and $\lambda_s^{jq} = [h_s^{jq}, h_{s+1}^{jq})$ that are compatible (since S_k^* is Λ_k-feasible) but such that $t^{ir} = h_p^{ir}$ and $t^{jq} = h_s^{jq}$ violate either constraint *ii*) or constraint *iii*) of Definition 4.3. We consider separately the two cases.

A route constraint is violated. In this case, $i = j$ and we can assume, w.l.o.g., that $r \prec q$. Since t^{ir} and t^{iq} violate a route constraint, then $t^{ir} + l^{ir} < t^{iq}$. Moreover, as λ_p^{ir} and λ_s^{iq} are compatible in S_k^*, then $t^{ir} + l^{ir} \leq h_{s+1}^{iq}$. Therefore, we construct Λ_{k+1} from Λ_k (and then G_{k+1} from G_k) by replacing the interval λ_s^{iq} with the two smaller intervals $\lambda_{s'}^{iq} = [h_s^{iq}, t^{ir} + l^{ir})$ and $\lambda_{s''}^{iq} = [t^{ir} + l^{ir}, h_{s+1}^{iq})$.
Observe that, in Λ_{k+1}, the intervals λ_p^{ir} and $\lambda_{s'}^{iq}$ are incompatible and, as a consequence, the optimal stable set S_{k+1}^* (the optimal schedule τ_{k+1}^*) calculated at the next iteration of the algorithm cannot contain both λ_p^{ir} and $\lambda_{s'}^{iq}$ ($t^{ir} = h_p^{ir}$ and $t^{iq} = h_s^{iq}$). This is equivalent to



(a) Violated route constraint between two track segments traversed by train i, with $r \prec q$.



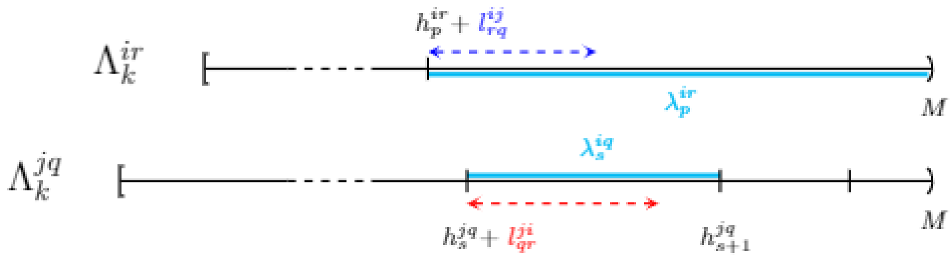
(b) Resolution of the route incompatibility shown in Figure 4.4a.

Figure 4.4: Example of resolution of a route incompatibility at Step 3 of the DDD-ALG.

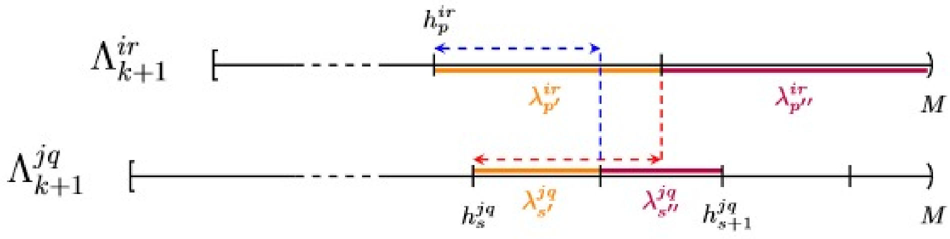
add an edge $\{\lambda_p^{ir}, \lambda_{s'}^{iq}\}$ in the set E_{k+1}, and thus append the term $x_{s'}^{iq}$ to the fixed precedence clique associated to x_p^{ir}. Concurrently, we also have to insert a new fixed precedence clique relative to the interval $\lambda_{s''}^{iq}$. Figure 4.4 shows how, starting from the route constraint violated in Figure 4.4a, the new intervals $\lambda_{s'}^{iq}$ and $\lambda_{s''}^{iq}$ are generated.

A disjunctive constraint is violated. In this case, we have that $i \neq j$, $t^{jq} < t^{ir} + l_{rq}^{ij}$ and, at the same time, $t^{ir} < t^{jq} + l_{qr}^{ji}$. Since λ_p^{ir} and λ_s^{jq} are compatible in S_k^*, we have that $t^{ir} + l_{rq}^{ij} \leq h_{s+1}^{jq}$ and $t^{jq} + l_{qr}^{ji} \leq h_{p+1}^{ir}$. Therefore, we obtain Λ_{k+1} from Λ_k by breaking λ_p^{ir} into $\lambda_{p'}^{ir} = [h_p^{ir}, t^{jq} + l_{qr}^{jq})$ and $\lambda_{p''}^{ir} = [t^{jq} + l_{qr}^{jq}, h_{p+1}^{ir})$, and λ_s^{jq} into $\lambda_{s'}^{jq} = [h_s^{jq}, t^{ir} + l_{rq}^{ij})$ and $\lambda_{s''}^{jq} = [t^{ir} + l_{rq}^{ij}, h_{s+1}^{jq})$.
Again here, we observe that the newly generated intervals $\lambda_{p'}^{ir}$ and $\lambda_{s'}^{jq}$ are now incompatible. This implies that an edge $\{\lambda_{p'}^{ir}, \lambda_{s'}^{jq}\}$ must be added in the set E_{k+1} since $\tau_{k+1}^* = \Phi(S_{k+1}^*)$ cannot contain the couple $(t^{ir} = h_p^{ir}, t^{jq} = h_s^{jq})$. Also, note that other conflicts and route incompatibilities may be generated by the definition of the new intervals. Consequently, these incompatibilities should be properly added to the set E_{k+1}. In Figure 4.5 we report the generation of the new intervals.

In both cases, once a new interval is defined, we can *propagate* the new time points discovered along the train route in order to ensure a time consistency with the intervals belonging



(a) Violated conflict constraint between two different trains i and j traversing the non-shareable track segments r and q.



(b) Resolution of a conflict incompatibility shown in Figure 4.5a.

Figure 4.5: Example of resolution of a conflict incompatibility at Step 3 of the DDD-ALG.

to subsequent track segments. With some abuse of notation in the following, given a track segment r traversed by a train i, we indicate the subsequent track segment on the rail path of i with the index $(r+1)$. Then, said $\lambda_p^{ir} = [h_p^{ir}, h_{p+1}^{ir})$ a newly generated interval and $\lambda_s^{i(r+1)} = [h_s^{i(r+1)}, h_{s+1}^{i(r+1)}) = max\{\Lambda_k^{i(r+1)} \ni \lambda_s^{i(r+1)} \mid h_s^{i(r+1)} < h_p^{ir} + l^{ir}\}$, we define two new intervals of the type $\lambda_{s'}^{i(r+1)} = [h_s^{i(r+1)}, h_p^{ir} + l^{ir})$ and $\lambda_{s''}^{i(r+1)} = [h_p^{ir} + l^{ir}, h_{s+1}^{i(r+1)})$. This procedure allows us to identify a lower bound $h_{s''}^{i(r+1)}$ for $(r+1)$ that avoids the violation of the route constraints. The propagation is thus recursively applied on $\lambda_{s'}^{i(r+1)}$ until the train destination segment is reached. Naturally, the creation of these intervals must be done in conjunction with the addition of any resulting route and/or conflict incompatibility between the intervals in Λ_k.

Summarizing, said $G_k(V_k, E_k)$ the DDD graph at a generic iteration k, the DDD-ALG follows the steps reported below. Note that, since we apply a route time consistency each time we propagate a new interval, the feasibility check on the associated schedule τ_k^* can be limited to the disjunctive constraint.

```
DDD-ALG $k - th$ iteration

  Data: Graph $G_k(V_k := \Lambda_k, E_k)$, functions $\Phi(S) : \Lambda_k \to \mathbb{R}_+$ and $\bar{w}(S) : \Lambda_k \to \mathbb{R}_+$.

  Solve the DDD problem (Step 2)
    Find $S_k^*$ on $G_k(V_k, E_k)$ and compute $\tau_k^* = \Phi(S^*)$

  $\Diamond$Check for solution optimality
    For each $i, j \in I$, $i \neq j$, and $r \in R_i$, $q \in R_j$ with $\{r, q\} \in \mathcal{D}$
      If $t^{ir}, t^{jq} \in \tau_k^*$ violate a disjunctive constraint (Definition 4.3 iii)) Then
          (i) from $\lambda_p^{ir} \in S_k^*$ define $\lambda_{p'}^{ir} = [h_p^{ir}, t^{jq}+l_{qr}^{jq})$ and $\lambda_{p''}^{ir} = [t^{jq}+l_{qr}^{jq}, h_{p+1}^{ir})$,
              set $\Lambda_k^{ir} = \Lambda_k^{ir} \setminus \{\lambda_p^{ir}\} \cup \{\lambda_{p'}^{ir}, \lambda_{p''}^{ir}\}$;
          (ii) from $\lambda_s^{jq} \in S_k^*$ define $\lambda_{s'}^{jq} = [h_s^{jq}, t^{ir}+l_{rq}^{ij})$ and $\lambda_{s''}^{jq} = [t^{ir}+l_{rq}^{ij}, h_{s+1}^{jq})$,
              set $\Lambda_k^{jq} = \Lambda_k^{jq} \setminus \{\lambda_s^{jq}\} \cup \{\lambda_{s'}^{jq}, \lambda_{s''}^{jq}\}$;
          (iii) set $E_k = E_k \cup \{\lambda_{p'}^{ir}, \lambda_{s'}^{jq}\}$.
    If $\nexists$ violated disjunctive constraint Then
      STOP. $\tau^* = \tau_k^*$ with value $w(\tau^*) = \bar{w}(S_k^*)$.

  Refine the DDD problem (Step 3)
    For each newly defined interval $\lambda_p^{ir}$
        (i) update resource assignment clique: for each $\lambda_{p'}^{ir} \in \Lambda_k^{ir} \setminus \{\lambda_p^{ir}\}$ set $E_k = E_k \cup \{\lambda_p^{ir}, \lambda_{p'}^{ir}\}$;
        (ii) update fixed precedence clique (Definition 4.4): for each $\lambda_s^{iq} \in \Lambda_k^{iq}$ with $(q+1) = r$ such that $h_s^{iq} + l^{iq} > h_{p+1}^{ir}$ add a new edge $\{\lambda_p^{ir}, \lambda_s^{iq}\}$ to $E_k$;
        (iii) update conflict incompatibility (Definition 4.5): for each $\lambda_s^{iq} \in \Lambda_k^{iq}$ with $i \neq j$, $\{r, q\} \in \mathcal{D}$, such that $h_{p+1}^{ir} < h_{p'}^{jq} + l_{qr}^{ji}$ and $h_{p'+1}^{jq} < h_p^{ir} + l_{rq}^{ij}$ add a new edge $\{\lambda_p^{ir}, \lambda_s^{jq}\}$ to $E_k$;
        (iv) propagate along the train route: if exists $(r+1) \in R_i$, let $\lambda_s^{i(r+1)} = max\{\Lambda_k^{i(r+1)} \ni \lambda_s^{i(r+1)} \mid h_s^{i(r+1)} < h_p^{ir}+l^{ir}\}$, define $\lambda_{s'}^{i(r+1)} = [h_s^{i(r+1)}, h_p^{ir}+l^{ir})$ and $\lambda_{s''}^{i(r+1)} = [h_p^{ir}+l^{ir}, h_{s+1}^{i(r+1)})$, set $\Lambda_k^{i(r+1)} = \Lambda_k^{i(r+1)} \setminus \{\lambda_s^{i(r+1)}\} \cup \{\lambda_{s'}^{i(r+1)}, \lambda_{s''}^{i(r+1)}\}$.
    Set $V_{k+1} = V_k$ and $E_{k+1} = E_k$
    $k = k + 1$
```

4.5.4 Convergence of the algorithm

We now show that the DDD-ALG converges to the optimal solution in a finite number of steps. We remark that our approach to solve the TTRP can be also seen as a primal-dual procedure with both row and column generations (see Section 2.2.4). At each iteration k a restricted relaxation of the basic problem is solved, if the solution cannot be converted into a feasible solution for the original formulation (at least) a row is added to the constraints groups and (at least) a new variable is generated and added to the problem D_{k+1}. Otherwise, Lemma 4.1 ensure that a solution of the same value can be obtained for the TTRP. In particular, we prove the following:

Theorem 4.1. *The DDD-ALG does always terminate providing the optimal solution of the TTRP.*

Proof. Lemma 4.1 shows that at each iteration k of the DDD-ALG, the value of the optimal solution S_k^* defines a lower bound on the value of the optimal solution of the TTRP. At the first iteration of the algorithm, each partition Λ^{ir} is defined by one interval (see Section 4.5.1). As shown in Section 4.5.3, at each iteration k, we add at least one interval to the current set Λ_k. Then, Observation 4.1 implies that, after at most $\sum_{i \in I} |R_i| M$ iterations, the DDD-ALG terminates providing the optimal solution for the TTRP. $\square$

4.6 An illustrative example

In this section we present an example application showing how the DDD graph is iteratively built by the DDD-ALG.

We are given four trains: $I = \{1, 2, 3, 4\}$ running on a railway network portion composed by seven distinct track segments: $R = \{a, b, c, d, e, f, g\}$. Trains routes, i.e. the ordered sequences of track segments occupied by each train, are defined as follows: $R_1 : \{a, b, g\}$, $R_2 : \{c, b\}$, $R_3 : \{d, b, f\}$ and $R_4 : \{e, f\}$. All trains have the same characteristics and priorities, therefore the time needed to traverse the track segments does not depend on the rolling stock and the train service. We have that $l = \{l^a, l^b, l^c, l^d, l^e, l^f, l^g\} = \{6, 3, 4, 9, 10, 5, 8\}$. All trains depart from their origin at time 0. Figure 4.6 schematically depicts the railway network topography, showing for each track segment the traversing time. Then, the graph in Figure 4.7 presents the trains given routes (oriented black lines) and the potentially conflicting pairs of non-shareable resources (red lines) defined by the set: $\mathcal{D} =$

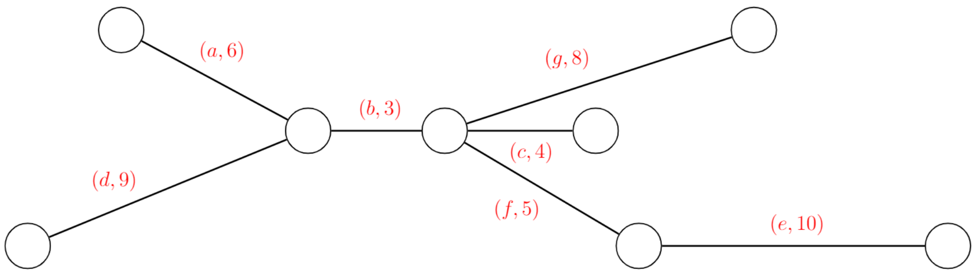


Figure 4.6: Topography of the example rail network. For each track segments we report in red, next to its label, the time needed by the trains to traverse it.

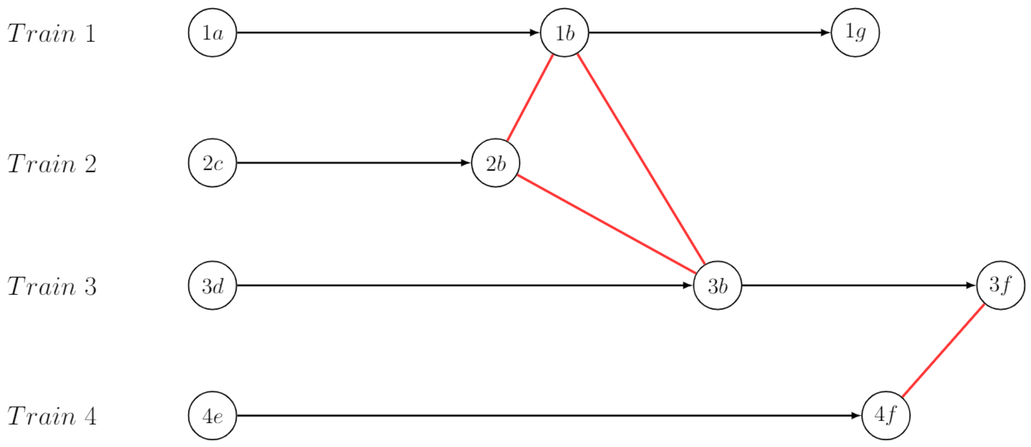


Figure 4.7: Graph of the trains routes. Oriented black lines represents trains fixed precedence, while red edges identify the couples of potentially conflicting events.

$\{\{1b, 2b\}, \{1b, 3b\}, \{2b, 3b\}, \{3f, 4f\}\}$. We simply consider the weight function in terms of scheduled departure times and for each interval λ_p^{ir} we set $\bar{w}(\lambda_p^{ir}) = w^{ir}(h_p^{ir}) = h_p^{ir}$.

Initialize the DDD problem. We build the initial graph $G_0(V_0, E_0)$ considering for each $i \in I$ and $r \in R_i$ an initial partition composed only by the interval $\lambda_1^{ir} = \{[\underline{t}^{ir}, M)\} = \{[0, 30)\}$. We thus associate with each interval a node in V_0. Each node is referred as v_h^{ir}, where h is the lower bound h_p^{ir} of the represented interval λ_p^{ir}, and labelled with h. We initialize $E_0 = \emptyset$. In Figure 4.8 we report the DDD graph G_0, where each train corresponds to a different layer in the graph. Note that at the beginning of the DDD-ALG we have only isolated nodes since we define a partition set with no route or conflict incompatibilities (see Definition 4.4 and Definition 4.5).

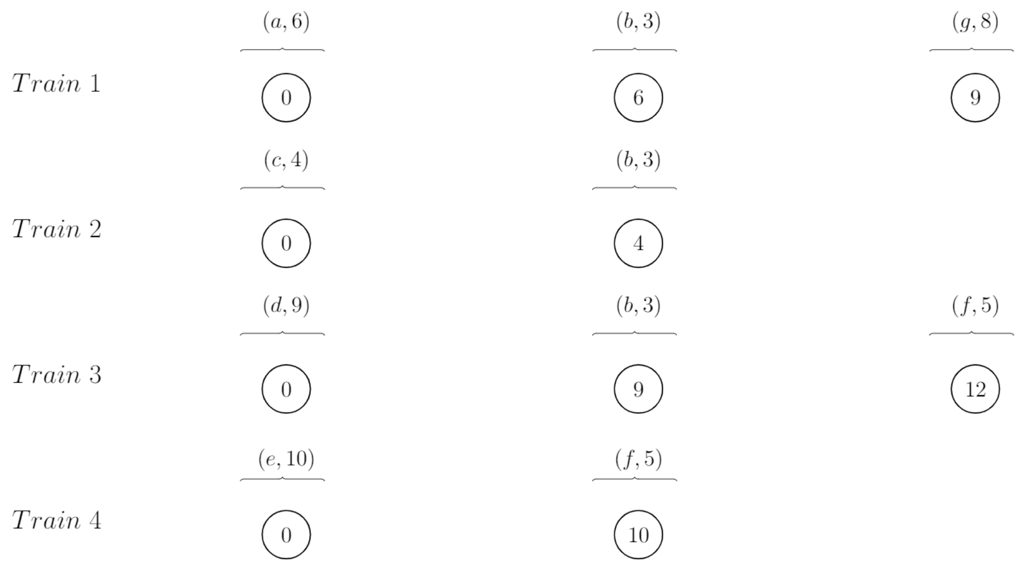
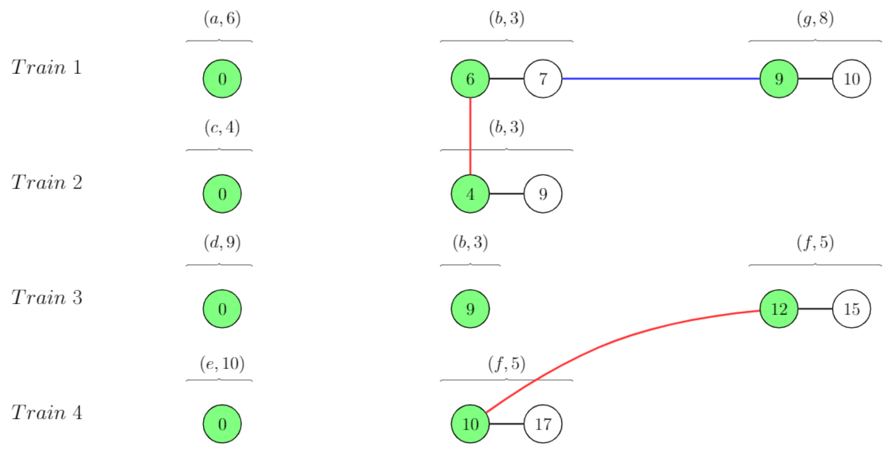
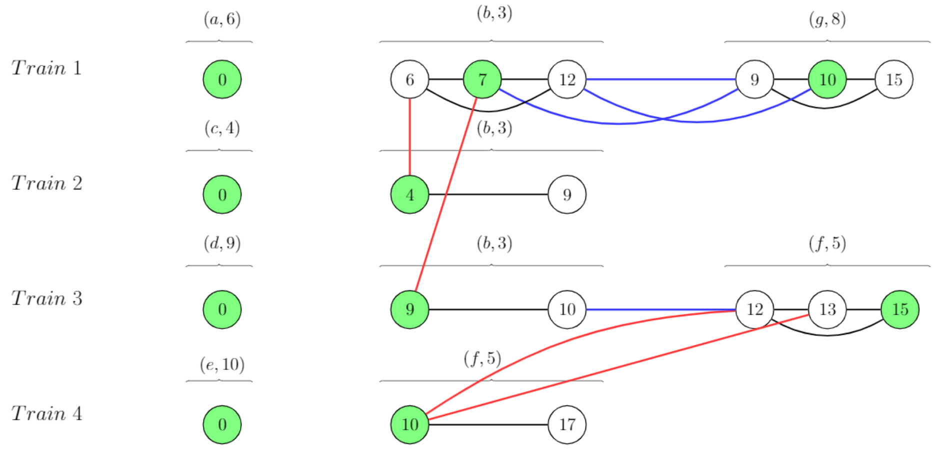


Figure 4.8: Graph $G_0(V_0, E_0)$ associated with the initialization of the DDD-ALG applied to the proposed example.

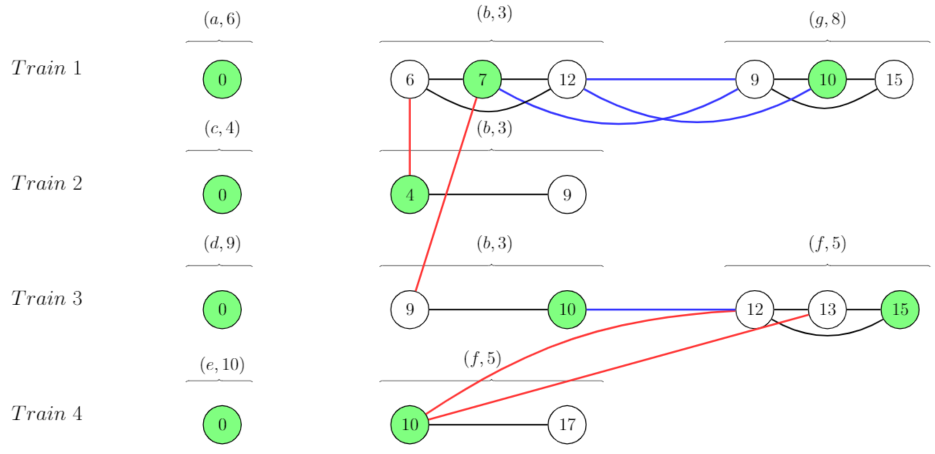
Apply the DDD-ALG. We now set $k = 1$ and apply to the example the proposed algorithm. Figure 4.9a, 4.9b and 4.9c respectively report the DDD graphs at the end of each iteration $k = \{1, 2, 3\}$. In particular, for each train i we visually group the nodes by track segments. We draw the edges belonging to the *resource assignment clique* in black and the one belonging to the *fixed precedence clique* in blue, while the arcs referring to the disjunctive conflicts are colored in red. Finally, the nodes belonging to the optimal stable set computed at *Step 2* are depicted filled in light green.
Note that at termination, the set S_3^* in the DDD graph $G_3(V_3, E_3)$ does not contain any adjacent node. The solution is therefore a stable set for the graph, namely the one having the minimum weight.



(a) $G_1(V_1, E_1)$



(b) $G_2(V_2, E_2)$



(c) $G_3(V_3, E_3)$

Figure 4.9: Graphs associated with the iterations of the DDD-ALG when applied to the proposed example.

DDD-ALG 1 – *st* iteration

Data: $G_1(V_1, \emptyset)$, Φ, $\bar{w}$.

Solve the DDD problem (Step 2)

$S_1^* = V_1 = \{v_0^{1a}, v_6^{1b}, v_9^{1g}, v_0^{2c}, v_4^{2b}, v_0^{3d}, v_9^{3b}, v_{12}^{3f}, v_0^{4f}, v_{10}^{4f}\}$, $\tau_1^* = \{0, 6, 9, 0, 4, 0, 9, 12, 0, 10\}$

$\diamondsuit$ ***Check for solution optimality***

We find $t^{1b} = 6$ and $t^{2b} = 4$ such that $6 < 4 + 3 = 7$ and $4 < 6 + 3 = 9$ hence:

(i) we define $\lambda_1^{1b} = [6, 7)$, $\lambda_2^{1b} = [7, 30)$, and add v_7^{1b} to V_1;

(ii) we define $\lambda_1^{2b} = [4, 9)$, $\lambda_2^{2b} = [9, 30)$, and add v_9^{2b} to V_1;

(iii) we add edge $\{v_6^{1b}, v_4^{2b}\}$ to E_1.

We find $t^{3f} = 12$ and $t^{4f} = 10$ such that $12 < 10 + 5 = 15$ and $10 < 12 + 5 = 17$ hence:

(i) we define $\lambda_1^{3f} = [12, 15)$, $\lambda_2^{3f} = [15, 30)$, and add v_{15}^{3f} to V_1;

(ii) we define $\lambda_1^{4f} = [10, 17)$, $\lambda_2^{4f} = [17, 30)$, and add v_{17}^{4f} to V_1;

(iii) we add edge $\{v_{12}^{3f}, v_{10}^{4f}\}$ to E_1.

Refine the DDD problem (Step 3)

We select the new interval $\lambda_2^{1b} = [7, 30)$:

(i) we add edge $\{v_6^{1b}, v_7^{1b}\}$ to E_1;

(ii) we add edge $\{v_7^{1b}, v_9^{1g}\}$ to E_1;

(iv) we propagate $\lambda_2^{1b} = [7, 30)$ on track segment g: we define $\lambda_1^{1g} = [9, 10)$, $\lambda_2^{1g} = [10, 30)$, and add v_{10}^{1g} to V_1.

We select the new interval $\lambda_2^{2b} = [9, 30)$:

(i) we add edge $\{v_4^{2b}, v_9^{2b}\}$ to E_1.

We select the new interval $\lambda_2^{3f} = [15, 30)$:

(i) we add edge $\{v_9^{3f}, v_{15}^{3f}\}$ to E_1.

We select the new interval $\lambda_2^{4f} = [17, 30)$:

(i) we add edge $\{v_{17}^{3f}, v_{10}^{4f}\}$ to E_1.

We select the new interval $\lambda_2^{1g} = [10, 30)$:

(i) we add edge $\{v_9^{1g}, v_{10}^{1g}\}$ to E_1.

We set $V_2 = V_1$ and $E_2 = E_1$

$k = 2$

DDD-ALG 2 – *nd* iteration

Data: $G_2(V_2, E_2)$, Φ, $\bar{w}$.

Solve the DDD problem (Step 2)

$S_2^* = \{v_0^{1a}, v_7^{1b}, v_{10}^{1g}, v_0^{2c}, v_4^{2b}, v_0^{3d}, v_9^{3b}, v_{15}^{3f}, v_0^{4f}, v_{10}^{4f}\}$, $\tau_2^* = \{0, 7, 10, 0, 4, 0, 9, 15, 0, 10\}$

$\diamondsuit$ ***Check for solution optimality***

We find $t^{1b} = 7$ and $t^{3b} = 9$ such that $7 < 9 + 3 = 12$ and $9 < 7 + 3 = 10$ hence:

(i) we define $\lambda_2^{1b} = [7, 12)$, $\lambda_3^{1b} = [12, 30)$, and add v_{12}^{1b} to V_2;

(ii) we define $\lambda_2^{3b} = [9, 10)$, $\lambda_3^{3b} = [10, 30)$, and add v_{10}^{3b} to V_2;

(iii) we add edge $\{v_7^{2b}, v_9^{2b}\}$ to E_2.

Refine the DDD problem (Step 3)

We select the new interval $\lambda_3^{1b} = [12, 30)$:

(i) we add edges $\{v_6^{1b}, v_{12}^{1b}\}$ and $\{v_7^{1b}, v_{12}^{1b}\}$ to E_2;

(ii) we add edges $\{v_{12}^{1b}, v_9^{1g}\}$ and $\{v_{12}^{1b}, v_{10}^{1g}\}$ to E_2;

(iv) we propagate t$\lambda_3^{1b} = [12, 30)$ on track segment g: we define $\lambda_2^{1g} = [10, 15)$, $\lambda_3^{1g} = [15, 30)$, and add v_{15}^{1g} to V_2.

We select the new interval $\lambda_3^{3b} = [10, 30)$:

(i) we add edges $\{v_9^{3b}, v_{10}^{3b}\}$ to E_2;

(ii) we add edges $\{v_{10}^{3b}, v_{12}^{3f}\}$ to E_2;

(iv) we propagate $\lambda_3^{3b} = [10, 30)$ on track segment f: we define $\lambda_2^{3f} = [10, 13)$, $\lambda_3^{1g} = [13, 15)$, and add v_{13}^{3f} to V_2.

We select the new interval $\lambda_3^{1g} = [15, 30)$:

(i) we add edges $\{v_9^{1g}, v_{15}^{1g}\}$ and $\{v_{10}^{1g}, v_{15}^{1g}\}$ to E_2.

We select the new interval $\lambda_3^{3f} = [13, 15)$:

(i) we add edges $\{v_{13}^{3f}, v_{12}^{3f}\}$ and $\{v_{13}^{3f}, v_{15}^{3f}\}$ to E_2;

(iii) we add edge $\{v_{13}^{3f}, v_{10}^{4f}\}$ to E_2.

We set $V_3 = V_2$ and $E_3 = E_2$
$k = 3$

DDD-ALG 3 – *rd* iteration

Data: $G_3(V_3, E_3)$, Φ, $\bar{w}$.

Solve the DDD problem (Step 2)
$S_3^* = \{v_0^{1a}, v_7^{1b}, v_{10}^{1g}, v_0^{2c}, v_4^{2b}, v_0^{3d}, v_{10}^{3b}, v_{15}^{3f}, v_0^{4f}, v_{10}^{4f}\}$, $\tau_3^* = \{0, 7, 10, 0, 4, 0, 10, 15, 0, 10\}$

$\diamondsuit$ ***Check for solution optimality***

No violated disjunctive constraint
Stop. $\tau^* = \tau_3^*$ with value $w(\tau^*) = \bar{w}(S_3^*) = 56$.

4.7 Computational experiments

In this section we report the results of the computational experiments conducted to asses the performance of the DDD approach in the train re-scheduling context. The test set consists of 20 real-life instances derived from two single-track railroad networks, later named *Line A* and *Line B*, of the Norwegian railway. We run the experiments considering a step-wise linear convex function that weights the trains delays. We compare the results achieved by the DDD-ALG with the ones obtained by the big-M formulation and by the (full) TI formulation.

4.7.1 Instances

The instances considered refer to portions of a physical railway infrastructure composed by stations and tracks. A set of trains with different speed classes traverses the net. For each of them we are given a desired timetable. At a given instant in time, the state of the network is provided by means of the current position of all trains and their deviations from the scheduled departure times. Note that when taking a snapshot of the network at a particular instant, a train may result *on time*, i.e. its arrival and departure times adhere to the timetable schedule, or it may be affected by a delay. Besides, a train can be either positioned at a station (i.e. *in station*) or on an open line track (i.e. *in connection*). In the second case, the delay refers to the time at which the the train entered the last track segment traversed, i.e. the one it is occupying at that moment. We do not consider the possibility of accelerating or decelerating the rolling stock, therefore we consider the train speeds, and consequently the train travel time, as fixed.
Table 4.1 reports some details about the test instances. For each of them, we present the number of trains ($|I|$), the total number of tracks ($|R|$), the average number of track

Instance	$\|I\|$	$\|R\|$	$\mathbb{E}[R_i]$	$\|\mathcal{D}\|$	$\#Dly$	$\mathbb{E}[d]_{(ms)}$
$\mathcal{I}_1^A$	28	33	20	5322	3	242
$\mathcal{I}_2^A$	24	33	18	3145	8	501
$\mathcal{I}_3^A$	16	33	21	1848	3	313
$\mathcal{I}_4^A$	12	33	20	943	4	293
$\mathcal{I}_5^A$	12	33	17	744	6	285
$\mathcal{I}_6^A$	8	33	17	250	4	406
$\mathcal{I}_7^A$	8	33	11	92	7	115
$\mathcal{I}_8^A$	25	33	19	3889	6	526
$\mathcal{I}_9^A$	19	33	19	2129	5	608
$\mathcal{I}_{10}^A$	16	33	17	1383	8	205
$\mathcal{I}_1^B$	18	25	16	1771	2	227
$\mathcal{I}_2^B$	5	25	13	69	2	105
$\mathcal{I}_3^B$	5	25	15	116	2	150
$\mathcal{I}_4^B$	16	25	17	1724	3	98
$\mathcal{I}_5^B$	5	25	16	116	1	157
$\mathcal{I}_6^B$	5	25	14	76	2	72
$\mathcal{I}_7^B$	18	25	16	2002	3	95
$\mathcal{I}_8^B$	4	25	14	54	3	59
$\mathcal{I}_9^B$	6	25	12	81	4	84
$\mathcal{I}_{10}^B$	6	25	11	64	4	68

Table 4.1: Relevant statistics of test instances.

segments per train ($\mathbb{E}[R_i]$), the total number of conflicting tracks pairs ($|\mathcal{D}|$), the number of trains having a positive delay at the "snapshot time" ($\#Dly$) and their average delay in milliseconds ($\mathbb{E}[d]_{(ms)}$). In the following, we will refer to the instances with the notation $\mathcal{I}_n^l$, where by $l \in \{A, B\}$ we denote the line to which they belong and by the subscript $n \in \{1, ..., 10\}$ we indicate the instance number. In particular, Line A is 124 km long line for passenger trains and includes 30 stations and 33 track segments. The A-instances present on average a set of 17 trains with an average of 18 track segments each. Line B is smaller (115 km, 20 stations and 25 tracks) and is crossed by commuters and freight trains. The B-instances present, on average, 9 trains and 14 track segments for each scheduled path. We emphasize that, since the instances of Line A include in most of the cases more trains than those belonging to Line B, the number of potential conflicting resources can be significantly higher for them, thus they will produce more complex models.

4.7.2 Settings

Given the state of the network at a particular instant in time, we define as optimal a schedule that minimizes the train delays at their final destination stations. Thus, we assign a

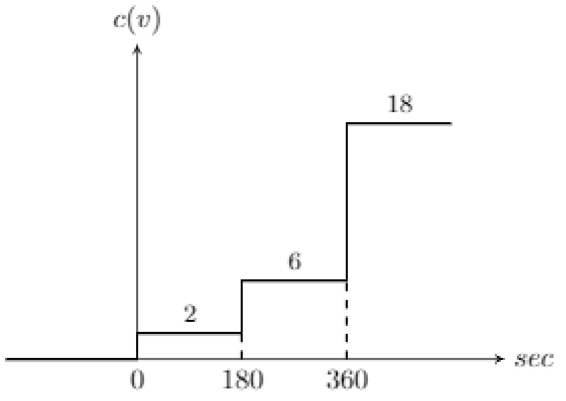


Figure 4.10: Example of a step-wise linear convex function considered to weight the delay (in seconds) of a generic train.

positive weight to the (positive) delays affecting the last scheduled activity of each train. We do not consider early arrivals since trains scheduled departure times are considered as constraints on the time of departure (see Section 4.3). The weight function considered is a step-wise linear convex function. Each train has its own *steps* valid for specific *delay range* values. An example for the function considered for a generic train is shown in Figure 4.10. Here we have respectively a step equal to 2 for the delay values between 0 and 3 minutes, a step equal to 6 between 3 and 6 minutes and a step of 18 for the remaining values of the delay. Note that such a weight function allows us to define different categories for delays and to classify trains into priority classes. Thus, trains with higher steps show a higher priority and we expect that an eventual delay for them is displaced among other trains belonging to a lower class.

We use an x64 MS Windows 10 machine with an Intel (R) Core (TM) i7-10510U CPU and 16 GB of RAM. The algorithm is implemented in Java and uses *IBM ILOG CPLEX v12.10* to solve, at each iteration, the associated DDD problem presented in Section 4.4.1. We use the same Mixed Integer Program (MIP) optimizer to solve for the big-M and TI formulations of the problem. All default *CPLEX* settings are used when testing the big-M and TI models. On the other hand, when solving the DDD problem, we force *CPLEX* to delete all *MIPStarts* solutions (which slow down the Branch&Bound search) and set a *MIPGap* tolerance to 5%. Besides, for the TI formulations, we consider a time window of 30 minutes and a fixed intervals width of 30 seconds (a standard value for the context under analysis).

$Instance$	k	$\#Conf$	$\mathbb{E}[\Lambda^{ir}]$	$Time(ms)$	$\%Opt$	$\%Built$	$OptLast(ms)$
$\mathcal{I}_1^A$	17	214	8	10758	69 %	30 %	446
$\mathcal{I}_2^A$	12	103	5	1780	71 %	28 %	144
$\mathcal{I}_3^A$	8	70	5	701	76 %	22 %	88
$\mathcal{I}_4^A$	4	29	3	155	79 %	19 %	36
$\mathcal{I}_5^A$	10	47	4	473	82 %	17 %	74
$\mathcal{I}_6^A$	4	12	2	57	91 %	8 %	15
$\mathcal{I}_7^A$	4	8	3	50	92 %	8 %	13
$\mathcal{I}_8^A$	15	166	7	4901	65 %	34 %	320
$\mathcal{I}_9^A$	24	130	5	2630	88 %	11 %	119
$\mathcal{I}_{10}^A$	5	27	2	164	88 %	10 %	36
$\mathcal{I}_1^B$	10	62	4	559	84 %	15 %	90
$\mathcal{I}_2^B$	60	68	4	669	96 %	2 %	12
$\mathcal{I}_3^B$	4	10	2	46	95 %	4 %	10
$\mathcal{I}_4^B$	15	84	6	2222	79 %	20 %	149
$\mathcal{I}_5^B$	3	21	5	134	74 %	25 %	35
$\mathcal{I}_6^B$	4	6	2	40	95 %	3 %	9
$\mathcal{I}_7^B$	13	107	7	2264	74 %	25 %	182
$\mathcal{I}_8^B$	5	10	3	40	95 %	5 %	8
$\mathcal{I}_9^B$	2	8	2	16	95 %	5 %	6
$\mathcal{I}_{10}^B$	1	2	1	6	91 %	0 %	2

Table 4.2: DDD-ALG computational results.

4.7.3 Computational results

Table 4.2 shows the computational results obtained by applying the DDD-ALG to the considered instances. The first columns present, for each instance ($Instance$), the number of iteration performed by the algorithm (k), the number of violated disjunctive constraints identified in the optimality check ($\#Conf$), the average number of intervals created for each train-track couple ($\mathbb{E}[\Lambda^{ir}]$). Column $Time(ms)$ reports then the total computation time (in milliseconds), while columns $\%Opt$ and $\%Built$ respectively show the percentage of time spent in solving the DDD problems and the percentage of time used to identify the conflicts and refine the DDD problem at each iteration (Step 2-3 of the DDD-ALG). Finally, last column ($OptLast(ms)$) provides the resolution time of the DDD problem in the last iteration of the algorithm.

The DDD-ALG is able to solve to optimality all the instances of the considered data set. Note that, the number of iterations and the number of solved conflicts can vary significantly from instance to instance. Both these indicators affect the overall solution time. In fact, at each iteration k, a new DDD problem has to be solved and its complexity grows with

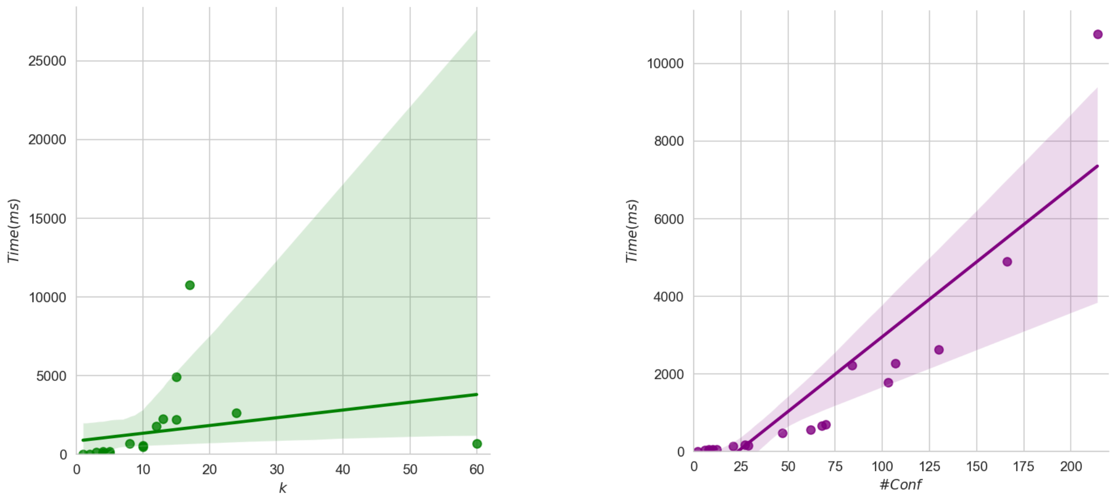


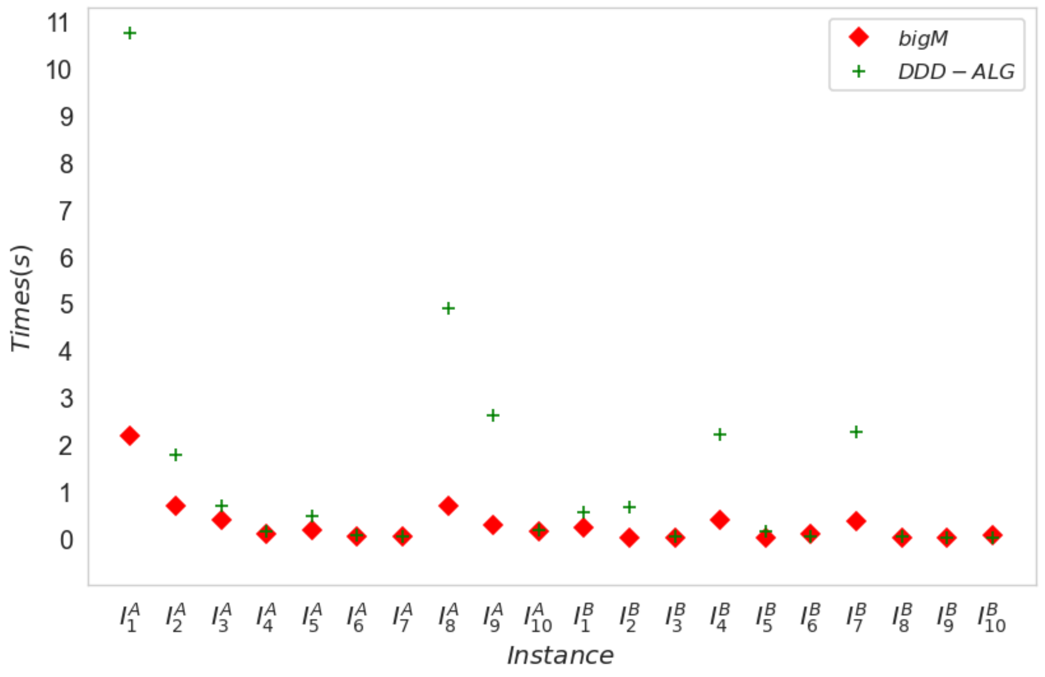
Figure 4.11: Correlation of the algorithm total running time $Time(ms)$ with the number of iterations k (on the left), and with the number of solved conflicts $\#Conf$ (on the right).

the number of nodes and edges defined in G_k. The scatter plots in Figure 4.11 show the correlation of the total running time ($Time(ms)$) with the number of iterations k of the DDD-ALG and with the number of violated disjunctive constraints solved $\#Conf$ (on the right). However, despite their size, all the DDD problems at the last iteration of the DDD-ALG are solved in a matter of milliseconds. Furthermore, we can observe that most of the time is taken by the DDD-ALG to solve the corresponding DDD problem. Though, detecting the conflicts and constructing the new model is not a negligible operation and, for 3 of the 20 instances, it takes about 30% of the total computation time. We highlight that, in these cases we also have an high number of conflicts pairs and intervals.

Table 4.3 compares the performances of the big-M and TI formulations with the one obtained by our algorithm. For each instance ($Inst$), we report the total number of trains ($|I|$) and the average number of track segments traversed by the trains ($\mathbb{E}[R_i]$). Then, for each resolution approach we report the following metrics. In columns Obj and $Time(ms)$ we present, respectively, the value of the optimal solution and the total computational time (in milliseconds). In particular, for the TI formulation section we report in parenthesis in the Opt column the percentage gap with respect to the *real* optimal solution value, due to the time step rounding. Besides, for the big-M and the TI approaches, columns $\#Var$ and $\#Con$ report the number of variables and constraints of the related formulations. Columns $\#Nod$ and $\#Edg$ in the DDD-ALG section refer instead to the number of nodes and edges of the DDD graph G_k at the last iteration of the algorithm.

			Big-M formulation				**TI formulation**				**DDD-ALG**			
$Inst$	$\|I\|$	$\mathbb{E}[\Lambda^{ir}]$	Obj	$Time$ (ms)	$\#Var$	$\#Con$	Obj $(gap\%)$	$Time$ (ms)	$\#Var$	$\#Con$	Obj	$Time$ (ms)	$\#Nod$	$\#Edg$
$\mathcal{I}_1^A$	28	20	158	2173	5977	11842	158	94245	34915	246999	158	10758	4707	186745
$\mathcal{I}_2^A$	24	18	1081	700	3650	7204	1081	30848	26485	160810	1081	1780	2561	58160
$\mathcal{I}_3^A$	16	21	106	382	2234	4404	106	18740	20666	110200	106	701	1739	22243
$\mathcal{I}_4^A$	12	20	137	91	1221	2394	137	11309	14798	65058	137	155	895	5195
$\mathcal{I}_5^A$	12	17	784	163	992	1936	784	8448	12968	51326	784	473	1011	21150
$\mathcal{I}_6^A$	8	17	61	37	410	788	61	4393	8320	19047	61	57	358	1215
$\mathcal{I}_7^A$	8	11	43	34	205	378	43	2286	5453	21744	43	50	271	987
$\mathcal{I}_8^A$	25	19	389	689	4440	8780	404 (4%)	47753	29111	205806	389	4901	3495	183946
$\mathcal{I}_9^A$	19	19	171	292	2547	5018	174 (2%)	22268	22078	128728	171	2630	2123	120522
$\mathcal{I}_{10}^A$	16	17	318	150	1716	3368	321 (1%)	14646	17433	79610	318	164	798	4783
$\mathcal{I}_1^B$	18	16	236	217	2113	4154	236	16416	17622	82935	236	559	1276	10479
$\mathcal{I}_2^B$	5	13	159	16	151	282	(100%)	2241	4102	22806	159	669	317	72571
$\mathcal{I}_3^B$	5	15	117	21	209	398	117	2788	4773	23478	117	46	228	753
$\mathcal{I}_4^B$	16	17	215	391	2047	4030	215	18271	16823	80727	215	2222	1813	101542
$\mathcal{I}_5^B$	5	16	76	21	214	408	110 (45%)	3067	5078	28399	76	134	445	16400
$\mathcal{I}_6^B$	5	14	46	89	161	302	46	2040	4285	16307	46	40	182	331
$\mathcal{I}_7^B$	18	16	248	364	2356	4640	249	21949	18354	93001	248	2264	2148	100303
$\mathcal{I}_8^B$	4	14	55	16	125	234	89 (62%)	1757	3611	15439	55	40	195	721
$\mathcal{I}_9^B$	6	12	484	17	171	318	484	2319	4410	33141	484	16	160	213
$\mathcal{I}_{10}^B$	6	11	104	53	148	272	104	1687	4044	16183	104	6	93	23

Table 4.3: Computational results comparison between big-M and TI formulations and DDD-ALG.



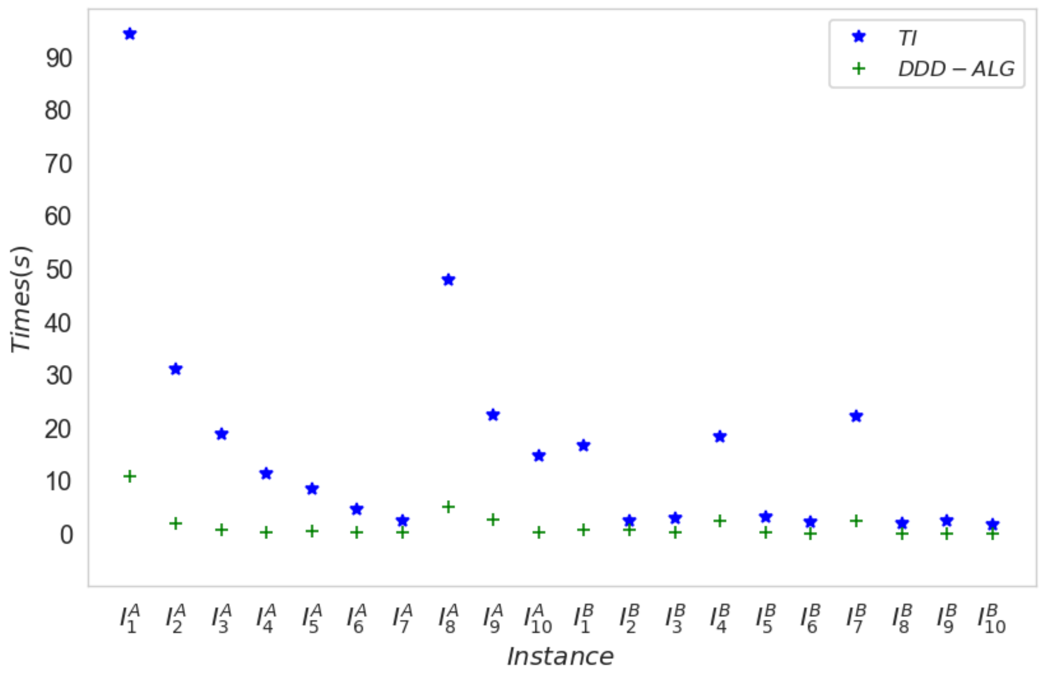


Figure 4.12: Computation times performance comparison between our algorithm and the big-M model (above) and between our algorithm and the TI model (below).

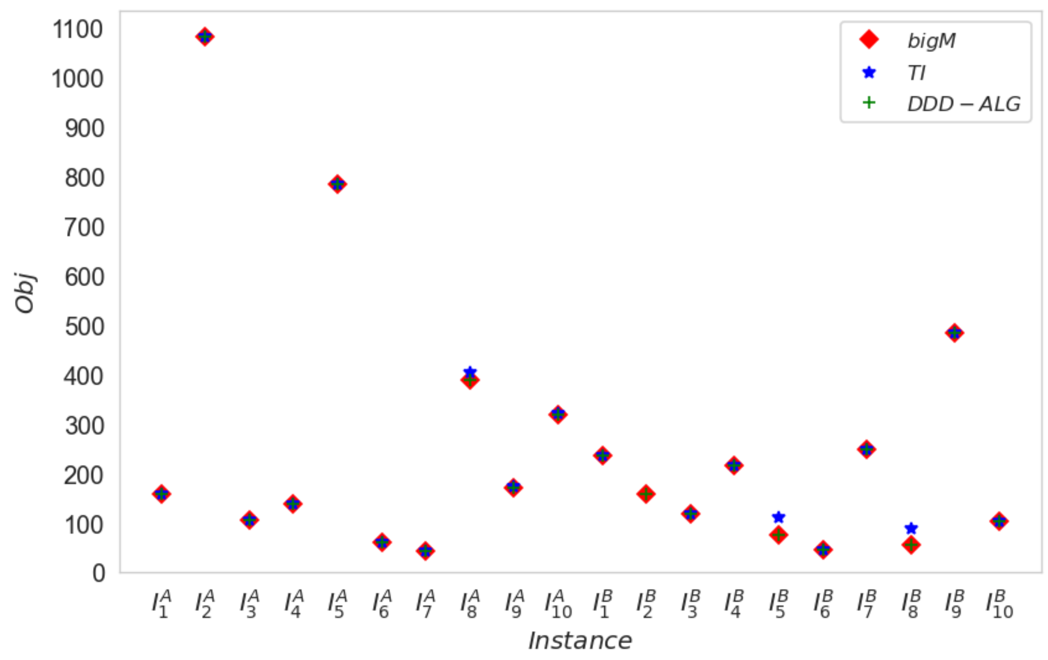


Figure 4.13: Comparison on the solution quality obtained by the big-M model, the TI model and the DDD-ALG.

Figure 4.12 and Figure 4.13 allow to visualize, respectively, the computational time performances and the solutions quality provided by the three models tested. We can notice that the DDD-ALG outperforms the TI formulation approach on all the considered instances. Observe that the TI approach is not able to solve one of the instances; this because the generalized assumptions made on the time window and interval width do not fit the special case of $\mathcal{I}_2^B$. Moreover, the TI approach reaches the *real* optimal solution only for 14 instances, while the huge number of variables and constraints introduces by the formulation drastically affects its resolution time. Thus, beside the fact that the DDD-ALG always provides the exact optimal solution, it also requires much less computation time.
On the other hand, by examining the performance of big-M models, we show that our algorithm exhibits similar performance in terms of computation time on medium-easy instances, i.e., those with few train conflicts. Nevertheless, the big-M approach is definitively the most efficient on the most difficult test instances.

We remark that the DDD problem generated by the last iteration of the DDD-ALG (column $OptLast(ms)$ of Table 4.2) is solved very efficiently and anyway faster than the big-M model (see graph in Figure 4.14). This fact is particularly relevant for two aspects. First,

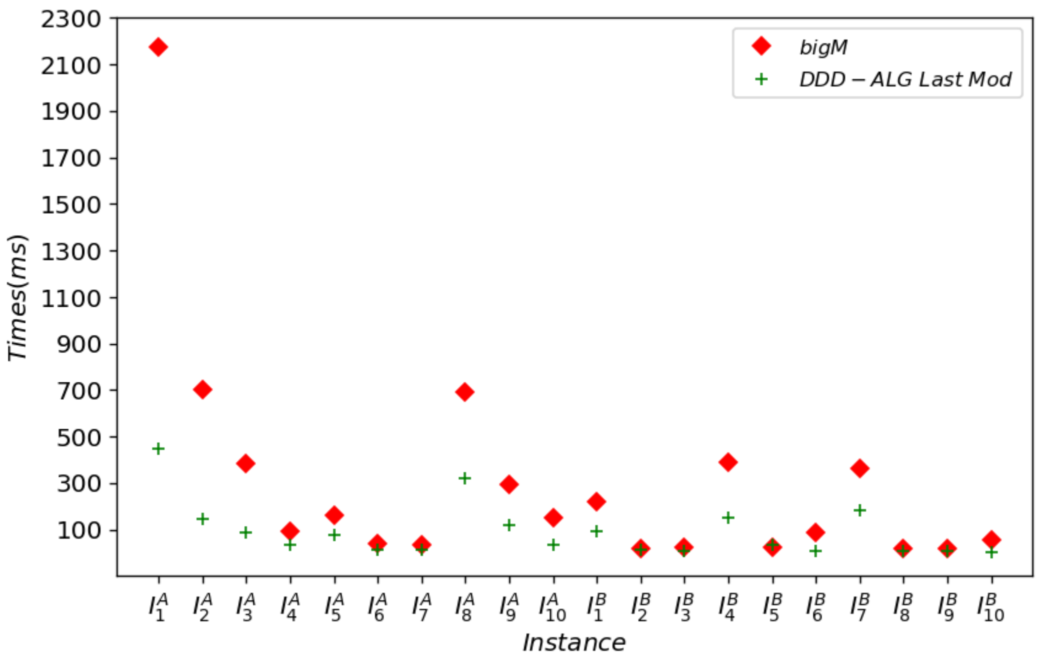


Figure 4.14: Computation times performance comparison between last iteration model solved by our algorithm and the big-M formulation model.

it encourages us to investigate a more effective way to detect and remove the conflicts at each iteration of the procedure, so to reduce the total number of required iterations, and therefore to speed up the total computational time. Secondly, the model resolution speed is crucial when applied to the dynamic solution of train dispatching problems. In this case, a new instance is generated from the field every ten seconds or so ([100]). Typically instances only slightly change over time; therefore, one can massage the last instance generated at the previous resolution process and hope to produce only a few new intervals[3].

4.8 Conclusions and future works

The main purpose of this chapter is to show how the DDD paradigm can be exploited in order to make the TI formulations competitive with the big-M formulations in the train dispatching field. To this end, we demonstrate that our our approach outperforms the standard TI formulation on all instances, whereas it is still inferior to the classical big-M approach on the difficult instances. On the other hand, the fact that the DDD behaves well

[3] We observe that this approach proved to be successful when extending the *Path&Cycle* formulation to cope with dynamic instances of the TTRP ([110]).

on instances with few conflicts, makes it suitable for real-time applications.

The fact that the DDD method can benefit more than other approaches from the computation performed in the previous iterations, makes it an excellent candidate for handling the train rescheduling problem. Indeed, in this framework we aim to solve a dynamic problem in which the next instance is very similar to the previous one.
It follows a very natural road map for future studies and developments. First, adapt the approach to handle dynamic problems and re-optimization. Actually, the refinement of the interval between an iteration and the next can be seen as a decomposition, so one can ask how to fit previously generated resolution cuts to new partial models. Also, fixed variable values retrieved in the preprocessing (solving) phase (or previously calculated bounds) can be exploited for those elements that are not directly "involved" in the conflicts solved at a generic iteration k. Second, explore ways to reduce the number of internal iterations of the DDD-ALG, for example by generating more intervals at each iteration. Third, develop techniques to limit the generation of sub-intervals at each iteration. In fact, it can be shown that normally a small subset of the intervals of the last model is actually sufficient to find an overall feasible (and thus optimal) solution. Fourth, the described methodology applies to any job-shop scheduling problem, so it is logical to extend it to cope with other contexts, such as industrial production or project scheduling.
Alternatively, a different line of research might be to reduce the DDD problem to a (Maximum) Satisfiability problem, by using the same idea of abstraction refinement with a unary-number-like representation of the time-domain variables. In particular, the SATisfiability problem (SAT) is the problem of deciding, given a boolean expression in variables, whether some assignment of the variables makes the expression true. It is a canonical $\mathcal{NP}$*-complete* problem, however many commercial (exact) SAT solvers have been proposed, developed and improved in the last decades. The good computational performances of such tools may turn out to be very promising also when used to solve the DDD problem.

Chapter 5

Safe Place Assignment for reliable railway systems[§]

5.1 Introduction

When a disruption occurs on a railway network, timetables may become infeasible and parts of the network unavailable for inbound trains. This requires decisions to be taken in real-time to mitigate the impact on the overall traffic. Generally train movements have to be re-planned in short time, that is, dispatchers have to take appropriate re-routing and re-scheduling decisions. In the literature the whole process is referred as *disruption management.* Arguably the most critical issue is to avoid creating *deadlocks*, a situation that arises when a group of trains is positioned in such a way that none can move due to other trains in the group blocking their paths. From the text of a law passed by the Kansas legislature at the beginning of the 20th century we can read:

"When two trains approach each other at a crossing, both shall come to a full stop and neither shall start upon again until the other has gone" .

Since different trains may require the same sections of the railway system, one or more trains have to move backward in order to reach again a conflict-free situation. Deadlocks are more prone to occur during disruptions, as the capacity of the network is often abruptly reduced, and dispatchers face unfamiliar and critical circumstances. In order to prevent

[§] This chapter is an extension of a work published on Transportation Science Journal [47]. The original work is co-authored with Prof. Carlo Mannino, from SINTEF (Oslo, Norway), PhD Paolo Ventura from IASI at CNR (Rome, Italy), PhD Veronica Dal Sasso e PhD Leonardo Lamorgese from Optrail (Rome, Italy). [A.L. Croella, V. Dal Sasso, L. Lamorgese, C. Mannino and P. Ventura. Disruption management in railway systems by safe place assignment. Transportation Science, 2021.]

deadlocks and in general to mitigate the negative effects of a disruption, dispatchers are required to implement specific recourse actions, sometimes based on predefined rules that are tailored on the specific railway system. In fact, even though train scheduling problems in different regions of the world share a common core problem, they may also differ significantly in certain aspects. For example, in Europe rail traffic is largely focused on operating passenger services, with the infrastructure manager being a single, public authority, and the train operating companies competing to provide services to passengers and, to a lesser extent, to ship goods. In other markets, such as the North American one, the rail industry is predominantly shaped by freight-hauling companies, with most companies owning and operating their rail network. Freight traffic presents some fairly different challenges from passenger traffic. Firstly, train traffic is often "un-scheduled", that is, no official timetable to adhere to exists. Secondly, railway business rules in America are somewhat more relaxed than in Europe, where trains generally have rigid routes between stations and certain train movements cannot be automated for (passenger) safety concerns. Finally, and perhaps more importantly, freight trains are typically much longer than passenger ones, which drastically amplifies the challenge described above in dealing with disruptions and their aftermath.

Commonly, the key objectives imposed aim:

- to leave, if possible, a viable path in the network with no obstructions, so as to allow work and maintenance trains to reach the disruption site(s);
- to quickly return to the normal traffic regime once the disruption is over;
- to ensure continuity in the service offered to passengers, by taking recourse actions such as minimizing cancellations, adding short-turn trains and substituting missing segments with alternative bus transport, etc.. Note that in general the cancellation of a train service is considered as the last option by dispatchers as it leads to major delays and represents a significantly unpleasant event for passengers.

Most of the contributions presented in the literature on disruption management deal with (some of) these three issues. However, despite quite a large scientific production on disruption management algorithms, we are not aware of existing real-life applications. Note that such practical implementations will become indispensable once Advanced Traffic Management Systems (ATMSs) will be used extensively. An ATMS is a digital train management solution capable of generating in real-time routes and schedules (*movement plans*) for the trains in a given region and planning horizon. ATMSs use on-train technologies and advanced digital communications to enforce safe trains movements improving system safety, efficiency and flexibility.

The model tackled in this chapter follows a real-life protocol adopted by a large Class I North-American railroad company to manage disruptions. The purpose is to find, for each train affected by a disruption, a *safe place*, that is a location on its path where the train can be safely parked during the critical situation, allowing traffic to route around it. Note that the word "safe" does not refer to the risk of accidents or damages to the infrastructure or rolling stock, which is responsibility of the safety layer of the signalling system. Currently, the *safe place assignment* is decided by experienced dispatchers in a greedy fashion. In the following sections we present a mathematical formulation for the Safe Place Assignment Problem (SPAP), later also denoted as *SP-protocol*. We develop an Integer Linear Program (ILP) approach that allows us to find an optimal assignment in a fraction of a second even for large instances and provides a strong support to dispatchers in their task. Indeed, as mentioned above, the application of the SPAP can also be related to the current usage of ATMSs. Since an ATMS solves instances of a hard optimization problem in very limited computational time, it may in general fail to return a plan and this can happen for different reasons. One possibility is that a solution exists, but the algorithm is not able to find it. Another is that the problem is infeasible, but the algorithm is not able to prove it. Infeasible instances correspond to so called *bound-to-deadlock* configurations, that occur when, given the trains' current positions, the system will inevitably end up in a deadlock. Bound-to-deadlock scenarios may vary in accordance with the railway layout in terms of number of trains involved and/or distance between them. Depending on the underlying algorithm used for planning, in these cases the ATMS may still be able to provide a *partial plan*, that is, a (feasible) plan that however spans a shorter planning horizon. Scheduling algorithms based on a "rolling horizon" paradigm, which tackle the problem by solving conflicts in a chronological order[1], are quite common in these applications ([116] [161]). The routes and schedules of such partial plans become the inputs of the SPAP. Thus, finding a solution to the problem for these plans adds an additional level of safety, as traffic controllers are provided with the last safe location for a train to drive within the plan's horizon. Trains would not be automatically let past these locations until a complete plan is available, or only when a dispatcher takes direct control of the situation.

The chapter is organized as follows. Section 5.2 gives an overview of related researches on disruption management. SPAP key elements, basic definitions and properties are presented

[1] This approach mimics the most common planning process of human dispatchers, that can therefore be seen as a particular rolling horizon algorithm.

in Section 5.3 and Section 5.4. Here we introduce the modelling choices adopted for the problem under analysis, then its purely binary programming formulation is described in Section 5.5. Finally, computational results over a set of instances of realistic size and complexity are presented in Section 5.6.

5.2 Literature overview

In recent years disturbance and disruption management have aroused a strong interest in the Operations Research (OR) field. Nielsen ([121]) gives a summary of the causes of disruptions., while in [90] Jespersen-Groth et al. describe the general disruption management process for the European rail system and emphasize the important role that OR models can play in limiting the impact of disruptions and improving railway systems performances. A growing number of algorithms and methods for the real-time re-scheduling have been proposed ever since, showing promising results both in terms of the solutions quality and resolution times.
According to the topography of the infrastructures, the type of transportation or the disturbance/disruption exact location and severity, different strategies can be used to recover from an unexpected event and restore a steady state in the network. Basically, a railway disruption management can be divided in three main sub-problems: timetable adjustment, rolling stock and crew re-scheduling. In the reminder of this section we focus our attention on the literature related to the timetable adjustment challenges and its various solution approaches. In particular, re-timing, short-turning, trains cancellation and stopping trains are just some of the possibilities at hand for the dispatchers that, at present, manually handle the timetable re-scheduling. We underline that most of the works presented on the real-time train re-scheduling, during and after a disruption, consider the railway system at macroscopic level ([27]). Next we report some examples of mathematical models developed in the disruption management context, for an extensive review on train re-scheduling models we refer the reader Section 3.2.

Narayanaswami and Rangaraj introduce in [119] the disruption events in a Mixed Integer Linear Program (MILP) re-scheduling model for a single-track line. Their formulation allows to re-time only the trains affected by the disruption (letting the others keep their schedule) and produces a new conflict-free timetable, to be applied at the end of the disruption. The goal is to minimize total delay of all trains at their respective destinations. However, a set of rigid assumptions limits the validity of the model for real re-scheduling applications.
Paper [142] implements an integrated real-time disruption control model acting at the

macroscopic level. A MILP model is used to decide between waiting, expressing and short-turning strategies to decrease the average passenger waiting time in a single-line system. The work highlights the sensitivity of short-turning to the accuracy of the disruption duration estimate.
Paper [89] develops instead a Mixed Integer Program (MIP) to reinsert the cancelled train services once an adequate level of regularity has been re-established. The objective is to recover the traffic regime scheduled by a periodic timetable as quickly as possible, by deciding which rolling stock units should cover which train services and when. The tests are carried on two lines running between Frederikssund and Farum station in Denmark and show a substantial improvement in the level of service offered to the passengers due to the real-time application of the mathematical formulation.
Paper [116] studies different probabilistic scenarios to generate and select meet-pass plans using a rolling horizon solution approach. Here, Meng and Zhou try to minimize the expected additional delay when restoring the original plan after a disruption event and do consider the uncertainty on the input data and recovery times.
Paper [30] addresses the disruption management problem in rapid transit network. The authors design an integrated approach for both timetable re-scheduling and rolling stock optimal assignment taking into account passenger demand behavior. A MILP model is formulated and tested on realistic instances of of a Spanish rail operator. Computation times account for a few minutes and show that the model is implementable in practice.
In [37] Coor simulates short-turning on a rail transit line (the *Blue Line* of the Massachusetts Bay Transportation Authority) in the morning peak period. Through this thesis work, a more critical use of the short-turning technique to minimize passenger wait times is addressed. In fact, for the rail system under analysis, short-turning appears to be valuable only in the case of severe delays.
For the case of partial or complete blockages in the network, [103] makes use of a MILP model for maximizing the level of service perceived by passengers: this means minimizing the number of canceled trains and the delays of the running trains. In the article Louwerse and Huisman take into account an estimation for the duration of the disruption event and determine a new schedule for the subset of trains that are allowed to run during the blockade. Moreover, they introduce the possibility of delay some trains with a few minutes and show how in this way the number of cancelled train service can be reduced.
The macroscopic model of [103] is extended in [154] by considering rerouting of trains and is formulated through an ILP as an event activity network. Tests on the Dutch railway network show that it can be successfully applied to handle large-scale disruptions. Indeed, a very low computation time (below 90 seconds) demonstrates its applicability in practice.

Paper [39] proposes instead an advanced re-scheduling approach to manage the traffic in a seriously disrupted and busy network with high risk of deadlocks. Both a centralized and a distributed procedure are tested. However, since the model is managed at the microscopic level, good quality solutions can only be found by reducing the time horizon considered (up to one hour).
Paper [17] formulates instead a macroscopic multi-objective ILP that takes into account passengers satisfaction, operational costs and deviation from the original timetable. By including several dispatching measures (cancellation, delay, rerouting, emergency train scheduling) the authors show that it is possible to include a *demand-driven approach* into disruption management with promising results. Yet, the work represents only a first step and is based on several simplified assumptions.
Ghaemi, Cats and Goverde propose in [71] and [72] two MILP models dealing with complete track blockages and describing the network respectively at a microscopic and macroscopic level. [71] model extends the formulation proposed in [125] by including short-turning possibilities. The aim is to find the optimal conflict-free routes for all trains in the net while choosing short-turning stations for a subset of them. The model is applied to two Dutch railway corridors when disruptions impede the normal traffic for a long time. Further, the formulation in [72] considers both sides of a disrupted location and includes short-turning, partial cancellation and re-timing of the trains. The possibility for each train to choose between two candidate short-turn stations is also added. Assuming that the length of the duration is known upon occurrence, the model seems promising in a real-time application.
The work [164] adds (to the dispatching measures to be adopted during disruption events) the possibility of skipping and adding stops and gives trains a full choice on short-turn stations. A macroscopic MILP model is formulated to minimize the re-scheduling impact on passengers satisfaction during complete track blockages. Results on double and single-track railway lines of the Dutch railway system show that the new dispatching options can lead to less passenger delays and knock-on delay effects. However, this is at expense of an increased number of cancelled train and of a higher sensitivity to the disruption length. A trade-off between the quality of the solution and the computation time should be considered.
Paper [166] addresses the case of multiple disruptions connected pairwise by at least one train line. All possible dispatching measures presented so far to handle a disruption appear in the model, that allows for train re-timing, reordering and cancellation, stops addition and flexible short-turning. Both a sequential approach (considering a single disruption at a time, respecting their order of occurrence) and a combined multi-disruptions technique are investigated. The latter, supported by a rolling-horizon method, is able to provide a solution for all experiments tested and appears to be the most efficient providing high-quality

re-scheduling in an acceptable time.
Paper [152] proposes a macroscopic MILP model for the timetable adjustment problem under infrastructure maintenance possession. Apart from re-timing and reordering measures, the model includes short-turning (handled in a preprocessing step) and trains cancellation. Furthermore, to account for station capacity constraints the authors propose a row generation approach able to tackle larger and more complex instances. The model is extended in [153] by integrating flexible short-turning possibilities and tested on large-scale networks in Netherlands. We remark that the validity of this model has been recognized by planners.
Lastly, Josyula et al. in [92] address a framework to classify, evaluate and compare the performances and capabilities of alternative approaches. The authors also applied the proposed evaluation framework as proof-of-concept to two train re-scheduling algorithms.

Depending on the traffic state of the network, Ghaemi et al. divide in [73] the disruption management process into three different phases, each of them marked by an essential decision to be made when handling a disruption. The article also operates a critical review of the models presented in the literature, highlighting the challenges faced by railways operators and traffic controllers. More in detail, the three different stages encountered by the network in case of disruption can be defined as follows:

- *first phase*: traffic decreases, the disruption impact and length are estimated; countermeasures and transition plans are evaluated,
- *second phase*: traffic remains low while a *disruption timetable* is applied;
- *third phase*: traffic and services are restored, a *transition plan* is applied to return to the original timetable.

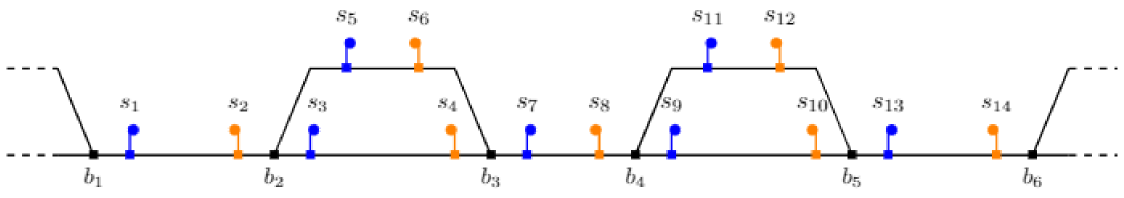
One of the main task of the first phase is the discussion on the adoption of a *contingency plan*. A contingency plan (or *disruption program*) is a predefined solution given by a set of actions that are established by traffic controllers and railway operators and are based on a specific disruption scenario. Countries such as Netherlands, Denmark, Germany, Switzerland and Japan currently use this system to deal with major disruptions and decide which trains should be cancelled or short-turned (and in which stations). Along the lines, the Norwegian infrastructure manager Bane NOR published a list of such rule, called *action cards* (see [10] - text is in Norwegian), defining how to move or where to stop the trains in the affected region during a disruption. Of course, all the designed contingency plans/action cards cannot cover all the disruption events that may arise in different portions of the net. These static solutions always need to be adapted and re-shaped according to the real state

of the system and, unfortunately, there is no universal formula to apply.
In [33] Chu and Oetting underline instead that one of the main cause of delays in the third phase is the queuing of trains at and in front of stations. Thus, they propose a method to better design the contingency plans by considering additional blocking times for trains at the microscopic level. The method is applied to two large German S-Bahn networks and used to assess the feasibility of such contingency plans.

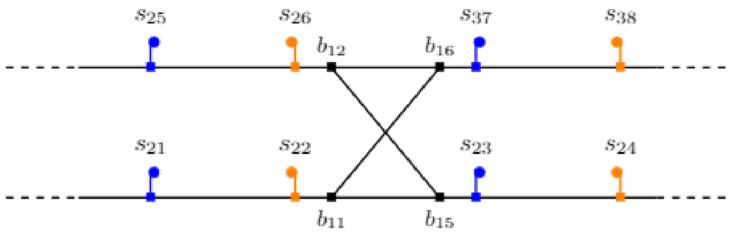
On the other hand, when no contingency plan can be applied, the railways operators first aim is to isolate the disrupted network portion by holding trains in stop positions ([73]). However, despite its significant practical relevance, this approach (in any of its variants) is rarely addressed in the scientific literature and we are only aware of three other papers on the subject.
In case of a long lasting line blockage due to an accident, [86] [160] and [162] refer to the process of establish which train should be stopped and where it should be parked as the *train stop deployment planning.* Hirai et al. ([86]) present a Petri-net-based model to assign trains to parking locations and formulate the problem as an integer program. The input of the model consists in (*i*) possible location where to stop train and (*ii*) a set of track that should be unoccupied. A number of constraints and not mandatory requirements limiting the choice of the trains parking locations is also identified. For example, trains can only stop at block sections with a platform that can fully accommodate it and that allows other trains to pass through the stations. The output of the model not only determines the final stop locations of the trains, but also the sequence of atomic movements that must be performed to reach them. Unfortunately, the resulting algorithm does not scale well and, since the network is described at a microscopic level, it cannot solve real-world instances in a reasonable time (as reported in [162]).
In [160] the train rescheduling management problem is tackled when a complete blockage in the network impedes the train short-turning and forces them to wait at a station until the disruption event is over. The authors formulate a mixed integer program to decide where to stop trains and their precedence in leaving the stations once the service is restored. In addition, the possibility of cancellation of train services is here taken into consideration and it is supposed to be minimized, along with the weighted delays of running trains. The model is macroscopic and does not account for information about the train routes and the signal systems. Nonetheless, it considers the timetable rescheduling for the entire day. The tests are carried on several disruption scenarios in one of the longest and busiest high-speed railway lines in China connecting Beijing to Shanghai. The research shows how the use of the MILP formulation is able to significantly reduce the impact of the disruption and

compares the results with the heuristic approach used in practice by train dispatchers (First Scheduled First Served (FSFS)). However, the complexity of the model necessitates in some experiments a two-step approach addressing the problem in separate steps, before and after the disruption is over.
Zhan et al. present in [162] a MILP model derived from [160]. It aims at re-scheduling and/or canceling and/or assigning trains to stop locations during a major disruption that presents a (complete) blockage for a relatively long period of time. Both long-distance and short-distance trains operating in the Beijing-Shanghai high-speed railway line are taken into account. Besides, unlike the majority of the previous works (and [160]), the authors chose to model at the macroscopic level the trains services with different directions, thus to manage also local and cross-line trains. The key objective is to minimize cancelled trains by deviating as little as possible from the original schedule and ensuring that the halted long-distance trains do not hinder other local trains routes. Given the capacities, the latter requirement is met by imposing that for each station at least a track is always available. Not too surprisingly, computational results show that disruptions of longer duration lead to an increase in both system complexity and computation time. Though, the model is able to find a solution in an acceptable amount of time (100 seconds ca.) for real-time application.

Note that, long high-speed railway lines that use seat reservations, such as the Chinese and the Japanese ones, usually need to stop trains on tracks or in stations for the time needed to recover from the disruption (see [160, 162]). In fact, it can be very inconvenient for the passengers to switch from one train to another. In these cases, the train stop deployment planning (also referred as *disrupted train service waiting strategy*) is one of the input of the train re-scheduling phase ([86]). On the other hand, short-line networks, as the ones in the European countries, where trains can be more easily rescheduled, usually apply trains cancellation and short-turning approaches (see [17, 71, 72, 103, 142, 152–154, 164, 166]).
Other aspects of the disruption management problem are considered in [165] and [167] that take into account the uncertainty on the disruption duration. Indeed, in the current practice, the disruption length is only estimated on a series of predictions based on expert opinion and mechanics' judgment. In [167] a Copula Bayesian Network model is developed to predict the disruption length and assist the Dutch Operational Control Center Rail. The authors take into account both factors affecting latency and repair time length and proved the real-life application of the model to be fast and robust. [165] focuses on the case of a complete blockage of all tracks connecting two stations and uses a two-stage rolling horizon stochastic approach to deal with the disruption duration uncertainty. Several scenarios with different disruption lengths are considered. The innovation lies in the fact that, once the real



(a) Single-track network.



(b) Multiple-track network.

Figure 5.1: Schematic railway networks. Figure 5.1a shows a single-track area, where stations are linked by only one track, while Figure 5.1b shows a portion of a multiple-track network, where more than one track enters/exits a station or a crossover.

disruption duration overtake a fixed range, the model is updated and a new re-scheduling model (based on the one presented in [164]) is solved. Then, new scenarios are accordingly defined. The performances of the procedure in a portion of the Dutch railway network show the primacy of the proposed approach on the deterministic rolling horizon models.
Furthermore, the concept of *resilience* of a railway network is addressed by Bešinović in [15], by considering proactive (planning a robust system) and reactive (efficiently recovering from a disruption) issues. Here, the term resilience is defined as "the ability of a railway system to provide effective services in normal conditions, as well as to resist, absorb, accommodate and recover quickly from disruptions or disasters". The author also gives a comprehensive review of railway resilience papers (relying on [163]) and classifies metrics and approaches, while evaluating the great potential of the mathematical optimization methods in improving the railway systems reliability.

5.3 Modelling the network

In this section we give a formal definition of the key elements of the SPAP, namely the railway network, trains and their movements.

Figure 5.1 shows a schematic representation of a portion of railway line. Here, solid segments represent tracks, while the small squares at the extreme of each track denote different objects: orange and blue squares correspond to *signal points*, locations where a signal is present and trains will stop (assuming a red light). Signal points are partitioned into *east*-bound signal points (where east-bound trains can stop) and *west*-bound signal points (where west-bound trains can stop). The orange squares are the east-bound signal points and are denoted by s_i, with i odd, while the blue squares are the west-bound signal points s_i with i even. The black squares represent *junction points*, where a track splits into two (or two tracks converge) and are instead denoted by b_i, with $i \in \mathbb{N}_+$. Note that the junctions actually identify track portions of different lengths for the different railway branches, but we model them as points. The length difference of using a reverse switch or not will be reflected on the length of occupied track portions, without the need of knowing the exact position of the ideal junction point. The track between two successive points z, f (either signals or junctions) is called a (track) *segment* (or *block section*) and denoted as an unordered pair of extremes $r = \{z, f\}$. Each track segment can accommodate at most one train at time. Example of segments are $\{b_4, s_8\}, \{s_1, s_2\}, \{b_4, s_9\}$, etc.. Following the movement of trains through the line, two "opposite" partial orders can be defined over the points and segments of a rail network: an east-bound order, corresponding to the order in which tracks are traversed by an east-bound train, and a west-bound order. Observe that for two points (segments) s_1, s_2 $(\bar{r}, \hat{r})$ we have that $s_1 \prec s_2$ $(\bar{r} \prec \hat{r})$ in the east-bound order if and only if $s_2 \prec s_1$ $(\hat{r} \prec \bar{r})$ in the west-bound order.

Rail paths. We now give a formal definitions for a rail path.

Definition 5.1. *A* rail path *is an alternating sequence of points and segments of the line that is ordered either west-bound or east-bound.*

In Figure 5.1, $P_W = (s_5,\ \{s_5, s_6\},\ s_6,\ \{s_6, b_3\},\ b_3,\ \{b_3, s_7\},\ s_7,\ \{s_7, s_8\},\ s_8)$ is a west-bound rail path and $P_E = (s_{14},\ \{s_{14},\ s_{13}\},\ s_{13},\ \{s_{13},\ b_5\},\ b_5,\ \{b_5, s_{10}\},\ s_{10})$ is an east-bound rail path. Note that the alternating sequence $(s_5,\ \{s_5, s_6\},\ s_6,\ \{s_6, b_3\},\ b_3,\ \{b_3, s_4\},\ s_4)$ is not a rail path, because it is ordered neither west-bound nor east-bound. Note that the initial position of an X-bound train, from its head to its tail, defines an X-bound rail path, with $X \in \{east, west\}$. We say that two rail paths *intersect* if they share at least a segment.

Train movements. Said I a set of trains, we assume that trains have fixed routes. Let i be an X-bound train, with $X \in \{east, west\}$, the movement of i across the network is identified by the movement of its head. The *route* R_i of a train $i \in I$ is the rail path obtained by concatenating the segments occupied by the train in its initial position, with the X-path followed by the head of i from origin to destination across the line. We let $R = \cup_{i \in I} R_i$. For our purpose we can decompose trains movements into a discrete sequence of elementary movements, or *steps*, from an X-bound signal to the next X-bound signal on R_i, and we can assume that, at the end of each elementary movement, the head of the train is always facing a signal (point) in such direction. After each elementary step, the train can stop at the signal point or proceed with the next step. Trains running on intersecting rail paths can be either *followers*, both running X-bound, or *crossing trains*, traversing the net in opposite directions. In single-track lines networks follower and crossing trains can meet or pass each other at a *siding*, a short track parallel to the main one that is able to accommodate a train. As an example, looking again at Figure 5.1a, segments $\{s_5, s_6\}$ and $\{s_6, b_3\}$ can be considered suitable siding allowing meet-pass events with the segment $\{s_3, s_4\}$.

Conflicts, blocking paths and schedules. As already mentioned, a conflict arises when two or more trains require the access to the same segment at the same time. We underline that, depending on its length from head to tail, a train i occupies a portion of the line which cannot be traversed by other trains. This portion starts at the signal point faced by the head of i and includes all the segments physically occupied by the train from head to tail. However, for safety reason, the portion of the line forbidden to other trains (*blocked* segments) can be extended "backward" on R_i from the segment containing the tail of i until a segment containing a signal point (in any direction) is met. This motivates the following definition:

Definition 5.2. *Say i a X-bound train with $X \in \{east, west\}$, R_i the route of i and assume that the head of i is at the X-bound signal point s. Then we associate with i, s a* blocking path $P(i, s)$ *as follows:*

1. *$P(i, s)$ is a sub-path of R_i with head-point at s,*
2. *The tail-point of $P(i, s)$ is a signal point,*
3. *$P(i, s)$ contains all segments physically occupied by i,*
4. *$P(i, s)$ is minimal.*

Condition 4 implies that any rail path satisfying conditions 1, 2 and 3 contains $P(i, s)$. To better grasp this definition we can look at Figure 5.2. Suppose, the segments $\{s_4, s_3\}$,

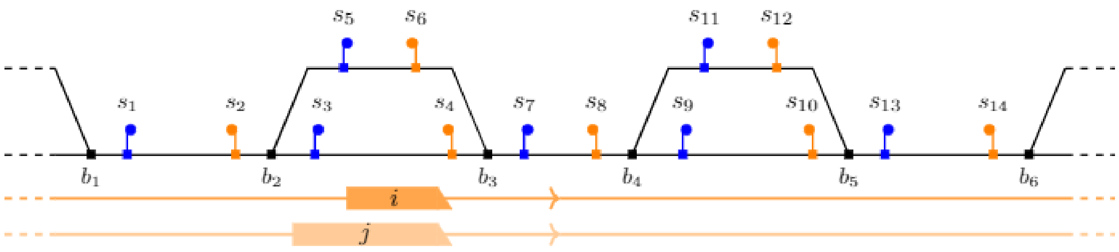


Figure 5.2: Examples of blocking paths occupied by trains. Trains i and j are east-bound trains having their heads at signal point s_4. We have that $P(i, s_4) = (\{s_4, s_3\})$ and $P(j, s_4) = (s_4, \{s_4, s_3\}, s_3, \{s_3, b_2\}, b_2, \{b_2, s_2\}, s_2)$.

$\{s_3, b_2\}$, and $\{b_2, s_2\}$, have length, respectively, 2100, 900 and 400 m and assume that an east-bound train i has its head at signal s_4. Now, if the length of a train i is $\leq$ 2100 m, then i occupies only the segment $(\{s_4, s_3\})$. On the other hand, if a train j has length 2450 m then j blocks the entire blocking path $(s_4, \{s_4, s_3\}, s_3, \{s_3, b_2\}, b_2, \{b_2, s_2\}, s_2)$. Note that the route R_i of i can also be viewed as an ordered sequence of blocking paths, which in turn corresponds to the ordered sequence of elementary movements of i. Also observe that two blocking paths of the same train can overlap on some segments. Summarising, when a train i is at a signal point s, it is actually blocking, or virtually occupying, all segments in the blocking path $P(i, s)$. Blocking paths play a central role when coordinating trains through the network since two trains can never occupy overlapping blocking paths at the same time.

Now, whit a *train plan* we specify the route for each train i and the schedule of i, i.e. the time train i enters each blocking path in its route. If P_i is a blocking path on the route R_i of i and P_j is a blocking path on the route R_j of another train j, and P_i and P_j intersect, we say that there is a *potential conflict* between i and j, and either i moves through P_i before j moves through P_j, or vice versa. Any feasible schedule for the trains in the network will hence establishes which train goes first for any potential conflicting pair of blocking paths.

5.4 The Safe Place Assignment Problem

One of the main goals of disruption management is to avoid deadlocks when trains resume their trip after a blockage event. Disruptions reduce the capacity of the network and may force dispatchers to hold a subset of trains, as their final destination is no longer reachable. As these disruptions may last for a long time, it is crucial to avoid that trains "pile" around the interested area, as trains should be able to reach their destination once the disruption is lifted. Moreover, depending on the type of disruption, it may be required that emergency vehicles reach the disruption location. These requirements introduce the idea of assigning *safe* places to the trains, that are locations where the trains can stop without impeding entirely the traffic flow.

In this section we describe an optimization model to support the SP-protocol for disruption management developed by a Class I North-American railway operator. We recall that when a major disruption occurs dispatchers may not be able to stick to the (real-time) plan they initially established. The SP-protocol enforces that affected trains hold at some location along their planned routes until the disruption is over and a regular traffic is restored. Now, an X-bound train i can only stop at an X-bound signal, and therefore the set of potential holding places is in correspondence to the X-bound signal points in R_i. Moreover, because i will indeed virtually occupy an entire blocking path, we can say that the set of (potential) safe places for i is the set $\mathcal{P}(i) = \{P(i,s) : s \in R_i\}$ of its blocking paths. Thus, the Safe Place Assignment Problem consists in assigning a safe place $\overline{P}_i \in \mathcal{P}(i)$ to each train i. In finding the best possible holding locations, i.e. the safe places, the SP-protocol pursues three distinct and somehow conflicting objectives:

O1 The primary goal is to avoid that when the disruption is lifted trains are bound-to-deadlock.

O2 Secondly, there is the need to leave some paths free of trains so that worker trains can reach the disrupted area.

O3 Finally, the aim is to "push" trains as far along their routes as possible, so as to mitigate the impact of the disruption in terms of train delays.

Objectives *O1* and *O2* are conditions imposed on the feasible assignments and will be translated into constraints of the ILP formulation in Section 5.5. In contrast *O3* only affects the objective function of the formulation. Notice that this SP-protocol has been defined according to the requirements of one railway operator. The particular type of trains and layout of the railway network considered may affect the definition of the preferred, or even of the feasible, safe places assignments. However, the formulation presented in this thesis is flexible and can be adapted to different requirements. As an example, if the preferred policy is not to push all the trains as far as possible, but there is a special focus on not stopping passengers trains outside of a station, this could be easily integrated in the model by properly modifying the objective function.
Next we show how the disjunctive graph theory (see Section 3.4.1) can be exploited in order to find a suitable set of safe places for the trains.

5.4.1 Implementing goal *O1*

We start the discussion with how to pursue goal *O1*. We assume that, before the disruption occurs, trains run through the network according to a feasible plan which specifies the routing and the schedule τ for all trains. As described in Section 3.4.1, we can associate with such a plan a disjunctive graph $G = (N, E, D)$ with arc lengths l, which represents our instance at the moment the disruption strikes. Note that the planned schedule τ induces a selection $\overline{S} = S(\tau)$, and $G(\overline{S})$ is acyclic. However, as a consequence of the disruption, trains have to hold somewhere on their routes, the scheduling instance changes and (in general) the schedule τ is no longer feasible.
Therefore, let ϕ be the function that, when the disruption occurs and according to the SP-protocol, assigns each train i a safe place $\overline{P}_i \in \mathcal{P}(i)$. Trains will arrive at their assigned safe place and will hold there until the disruption is lifted. Now, to represent the new scheduling instance associated with the disruption event and the additional constraints imposed on the movement of trains by the SP-protocol, we derive from the original instance $G = (N, E, D)$ a new disjunctive graph $G'(\phi) = (N', E', D')$ and distance function l'. The node set $N' = N \cup \{\Delta\}$ contains a new element Δ that represents the end of the disruption. Next, the set E' properly contains the original set $E \subset E'$, with their original lengths. Moreover, since we assume trains arrive at their assigned safe places before the disruption ends, we have $(\overline{P}_i, \Delta) \in E'$, with $l'_{\overline{P}_i\Delta} = 0$, for $i \in I$. Also, since the trains can only move away from their safe places after the disruption ends, we have $(\Delta, \sigma(\overline{P}_i)) \in E'$, with $l'_{\Delta\sigma(\overline{P}_i)} = 0$, for $i \in I$. Finally, because train routes have not changed and there are no new blocking paths, we have $D' = D$, and the edges in each disjunction maintain their original lengths.
Summarizing, the disjunctive graph $G'(\phi) = (N', E', D')$ with edge lengths l', associated with an assignment of safe places $\phi = \{\overline{P}_i \in \mathcal{P}(i) : i \in I\}$, is defined as follows:

- Node set $N' = N \cup \{\Delta\}$. Node w represents the event *end of disruption*.
- Edge set $E' = E \cup E^{in}_{\Delta} \cup E^{out}_{\Delta}$, where $E^{in}_{\Delta} = \{(\overline{P}_i, \Delta) : i \in I\}$, representing the fact that trains enter their safe place before the end of the disruption, and $E^{out}_{\Delta} = \{(\Delta, \sigma(\overline{P}_i)) : i \in I\}$, representing the fact that trains exit their safe place after the end of the disruption. Moreover, $l'_e = l_e$ if $e \in E$, and $l'_e = 0$ otherwise.
- Disjunctive arcs $D' = D$ and $l'_e = l_e$ for $e \in D'$.

Note that, since $D' = D$, the selection $\overline{S} = S(\tau)$ of G associated with the original schedule τ is also a (not necessarily acyclic) selection of $G'(\phi)$, although τ is not a feasible schedule for $G'(\phi)$. Observe now that, in order to achieve objective *O1*, we need to find an assignment of safe places ϕ and a selection S such that $G'(\phi)(S)$ is acyclic. We now present a theorem that provides a way to find such an assignment and a selection in a natural and effective way by exploiting $\bar{S}$. In the proof we make use of the following well known results of graph theory (see [36]):

Definition 5.3. *Given a directed graph $G = (N, E)$, a topological ordering is a function $f : N \to Z_+$ which associates with every node a non-negative integer such that $f(v) > f(u)$ for all $(u, v) \in E$.*

Theorem 5.1. *A directed graph $G(N, E)$ admits a topological ordering f if and only if it does not contain a directed cycle.*

Next theorem hence provides a necessary and sufficient condition on ϕ so that $G'(\phi)(\overline{S})$ is acyclic, i.e. so that the selection induced by the original schedule τ is still feasible after the disruption. This result is the basis for our ILP formulation in Section 5.5. In the following, for any $i \in I$ and $P, Q \in \mathcal{P}(i)$, we let $P \preceq Q$ ($P \prec Q$) if Q does not precede P (Q strictly follows P) on the route R_i of train i.

Theorem 5.2. *Let $G(\overline{S}) = (N, E \cup \bar{S})$ be acyclic and let $\overline{\phi}$ assign each train i a safe place $\overline{P}_i \in \mathcal{P}(i)$. Then the following claims are equivalent*

i) $G'(\overline{\phi})(\overline{S}) = (N', E' \cup \bar{S})$ is acyclic;

ii) for each $(\sigma(P_i), P_j) \in \overline{S}$ with $\overline{P}_i \preceq P_i$, we have that $\overline{P}_j \prec P_j$.

Proof. We first show that $i)$ implies $ii)$. In particular, we prove that if $\overline{S}$ contains an edge $(\sigma(P_i), P_j)$ such that $\overline{P}_i \preceq P_i$ with $\overline{P}_j \succeq P_j$, then $G'(\phi)(\overline{S})$ admits a cycle C. Indeed, since $\overline{P}_i \preceq P_i$, then $\sigma(\overline{P}_i) \preceq \sigma(P_i)$. Therefore, $\sigma(P_i)$ follows $\sigma(\overline{P}_i)$ on the route of train i and there is a path in $G'(\phi)$ from $\sigma(\overline{P}_i)$ to $\sigma(P_i)$, say Q_i (see Figure 5.3). Similarly, as $P_j \preceq \overline{P}_j$, $G'(\phi)$ also contains a directed path, say Q_j, from P_j to $\overline{P}_j$. Now, we get C by concatenating $(\Delta, \sigma(\overline{P}_i))$, Q_i, $(\sigma(P_i), P_j)$, Q_j, and $(\overline{P}_j, \Delta)$.
Now we prove that $ii)$ implies $i)$. First observe that, since $G(S)$ is acyclic, there is a topological ordering $f : N \to \{1, \ldots, n\}$ of its nodes. Now, let $B = \{P_i \succ \overline{P}_i : i \in I\}$, and let $A = N \setminus B$. Then we have $N' = A \cup B \cup \{\Delta\}$. We define $f' : N' \to \mathbb{Z}_+$ as follows:

$$f'(u) = \begin{cases} f(u) & \text{if } u \in A \quad ; \\ f(u) + n + 1 & \text{if } u \in B \quad ; \\ n + 1 & \text{if } u = \Delta \quad . \end{cases}$$

Figure 5.4 show graph $G'(\phi)$. Here, red filled nodes represent the chosen *safe place* assignment ϕ and black solid edges belong to the original set of arcs E, while edges belonging to

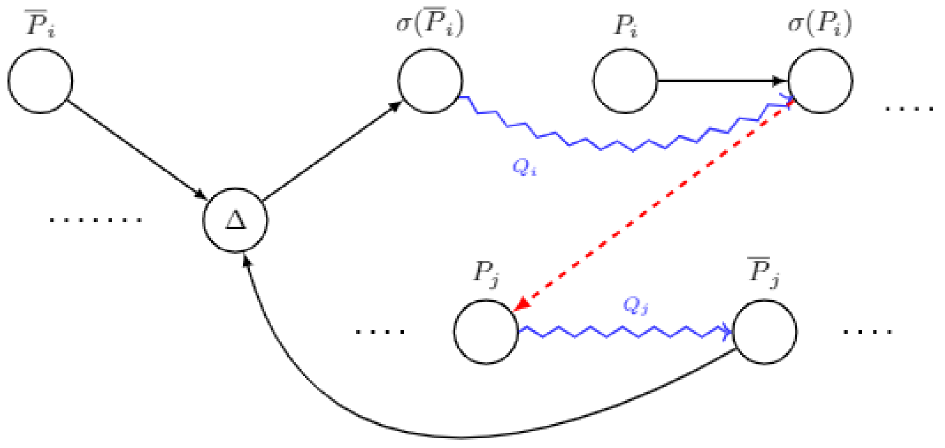


Figure 5.3: Proof of Theorem 5.2 i) -> ii).

$E' \setminus E$ are drawn in cyan (solid arrows for E^{in}_{Δ} and dashed arrows for E^{out}_{Δ}). Then, we show that f' is a topological ordering of $G'(\phi)$.
Suppose not, then there is an edge $(u, v) \in E' \cup \overline{S}$ such that $f'(u) > f'(v)$.

- If $u = \Delta$, then $v = \sigma(\overline{P}_j)$ for some $j \in I$. But then $\sigma(\overline{P}_j) \in B$, $f'(\sigma(\overline{P}_j)) \geq n+2$ and we have $f'(u) = f'(\Delta) = n+1 < f'(\sigma(\overline{P}_{\bar{i}})) = f'(v)$, a contradiction.
- If $v = \Delta$, then $u = \overline{P}_j$ for some $j \in I$. But then $\overline{P}_j \in A$, $f'(\overline{P}_j) \leq n$ and we have $f'(u) = f'(\overline{P}_j) \leq n < n+1 = f'(\Delta) = f'(v)$, a contradiction.
- Finally, consider the case with $u, v \neq \Delta$. Then arc (u, v) belongs also to G, implying $f(u) < f(v)$. Let $u = \sigma(P_i) \in \mathcal{P}(i)$, for a $i \in I$ and $v = P_j \in \mathcal{P}(j)$, for a $j \in I$. Since $f'(u) > f'(v)$, but $f(u) < f(v)$, we must have $u = \sigma(P_i) \in B$ and $v = P_j \in A$. But this implies that $i \neq j$, and $(\sigma(P_i), P_j) \in \overline{S}$. Then we have $\overline{P}_i \prec \sigma(P_i)$ and $P_j \preceq \overline{P}_j$, a contradiction.

□

Thanks to Theorem 5.2 we can ensure that the safe place assignment complies with all meet-pass events scheduled by the original plan. As, an example consider the railway area in Figure 5.5a. Two west-bound trains i and j are traversing the net when a disruption occurs. Their path are drawn in solid colored lines. The blocking paths are defined as follows:

- $\mathcal{P}(i) = \{P(i, s_2) = (\{s_1, b_1\}, \{b_1, s_2\}), P(i, s_4) = (\{s_2, b_2\}, \{b_2, s_2\}), P(i, s_5) = (\{s_4, b_3\}, \{b_3, s_5\}, P(i, s_6) = (\{s_5, s_6\}\}, P(i, s_7) = (\{s_6, s_7\}\}$;

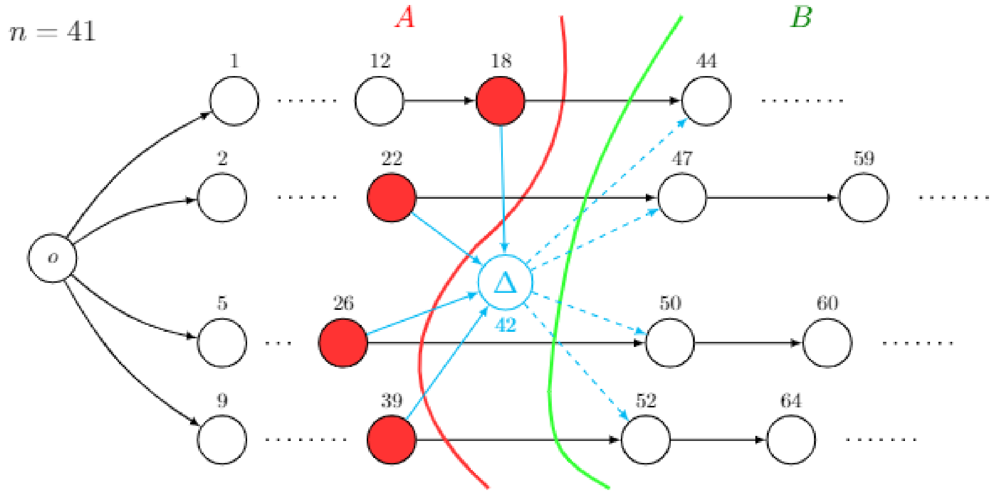


Figure 5.4: Proof of Theorem 5.2 ii) -> i).

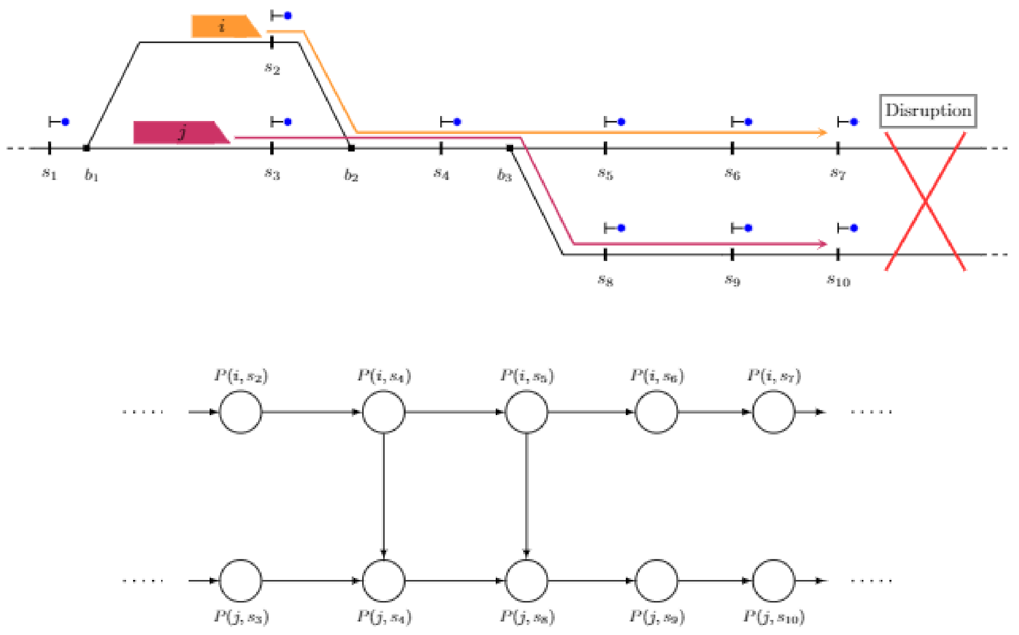


Figure 5.5: Example of a net portion traversed by two trains (above) and its disjunctive graph representation (below). Train i and j are shown in their current positions. Train i precedes train j on the rail segments $(\{b_2, s_4\}, \{s_4, b_3\})$ in the original timetable.

- $\mathcal{P}(j) = \{P(j, s_3) = (\{s_1, b_1\}, \{b_1, s_3\}), P(j, s_4) = (\{s_3, b_25\}, \{b_2, s_4\}), P(j, s_8) = (\{s_4, b_3\}, \{b_3, s_8\}, P(j, s_9) = (\{s_8, s_9\}, P(j, s_{10}) = (\{s_9, s_{10}\}\}$.

According to the scheduled plan, train i precedes train j on the rail segments ($\{b_2, s_4\}$ and $\{s_4, b_3\}$). This precedence constraints are represented by a selection of arcs on the associated disjunctive graph G, as shown in Figure 5.5. Then, Figure 5.6a and Figure 5.6c present respectively a feasible safe place assignment and an infeasible selection of safe place positions for the two trains. For both solutions we report in Figure 5.6b and 5.6d the new disjunctive graphs G' associated with these assignments. Note that, the infeasible assignment generates a cycle $(P(i, s_5), P(j, s_8)) \to (P(j, s_8), P(j, s_9)) \to (P(j, s_9), \Delta) \to (\Delta, P(i, s_5))$ in the associated disjunctive graph (Figure 5.6d) while int the feasible assignment graph presents no cycle (Figure 5.6b).

As a consequence of Theorem 5.2 we have that, if $G(\overline{S})$ is acyclic, then we can always assign all trains the safe places associated with their starting positions. Indeed, the following holds true.

Corollary 5.1. *Let $G(\overline{S})$ be acyclic and let $\tilde{\phi}$ assign each train i a safe place $\bar{P}_i$ that corresponds to its starting position P_i^0. Then $G'(\tilde{\phi})(\overline{S})$ is acyclic.*

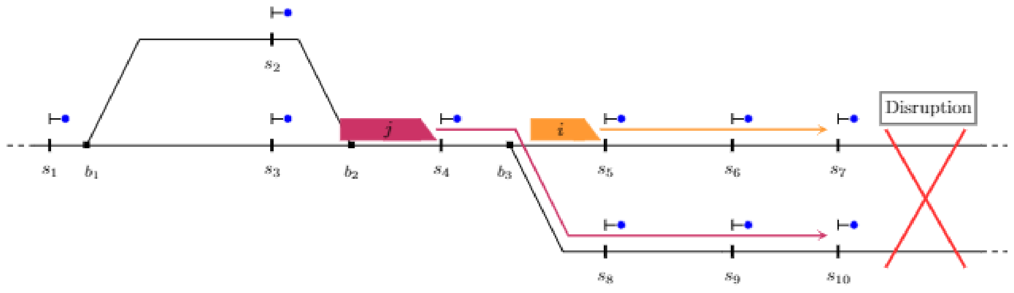
Proof. At the beginning of the planning horizon, every train $i \in I$ is in its starting position P_i^0 implying $\bar{P}_i = P_i^0 \preceq P_i$, for all $P_i \in \mathcal{P}(i)$. Suppose now $(\sigma(P_i), P_j) \in \overline{S}$ for some $\sigma(P_i) \in \mathcal{P}(i), P_j \in \mathcal{P}(j)$, with $i \neq j$. Since no train can enter (a segment of) P_j^0 before j, we must have $P_j \neq P_j^0$, and so $P_j^0 \prec P_j$, thus verifying condition *ii*) of Theorem 5.2.

□

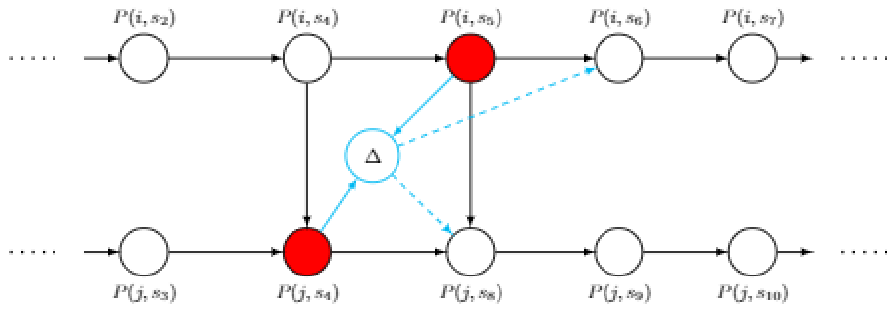
In our ILP model (see Section 5.5), we assume that a newly computed schedule that factors in the safe place assignment will also agree with the selection $\overline{S}$ associated with the original schedule τ. Then, we have constraints imposing that the safe place assignment $\tilde{\phi}$ satisfies the condition *ii*) of Theorem 5.2.

5.4.2 Implementing goal *O2*

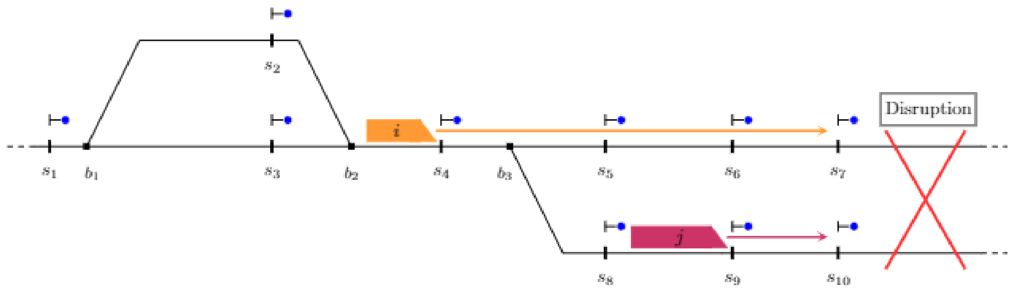
The purpose of goal *O2* is to allow worker trains to reach the disruption, bypassing trains that are parked in safe places. In order to model this requirement, we need first to give a formal definition for a rail path Q to bypass a blocking path P.



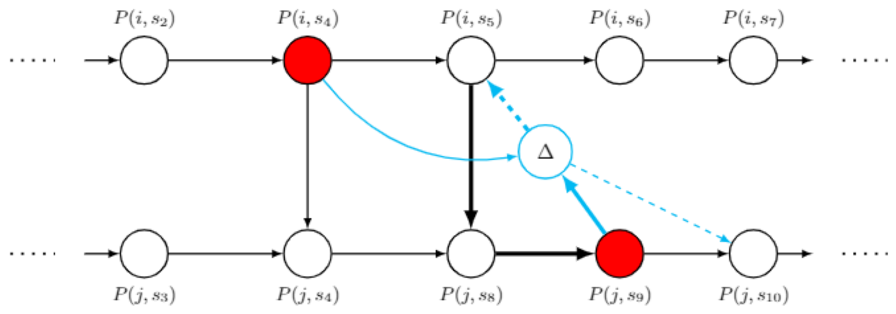
(a) Feasible safe place assignment for the net in Figure 5.5.



(b) Disjunctive graph associated with the feasible safe place assignment shown in Figure 5.6a.



(c) Infeasible safe place assignment for the net in Figure 5.5.



(d) Disjunctive graph associated with the infeasible safe place assignment shown in Figure 5.6c. The set of ticker arcs $\{(P(i, s_5), P(j, s_8))$, $(P(j, s_8), P(j, s_9))$, $(P(j, s_9), \Delta)$, $(\Delta, P(i, s_5))\}$ form a directed cycle..

Figure 5.6: Examples of a feasible and an infeasible safe place assignment for the net portion in Figure 5.5.

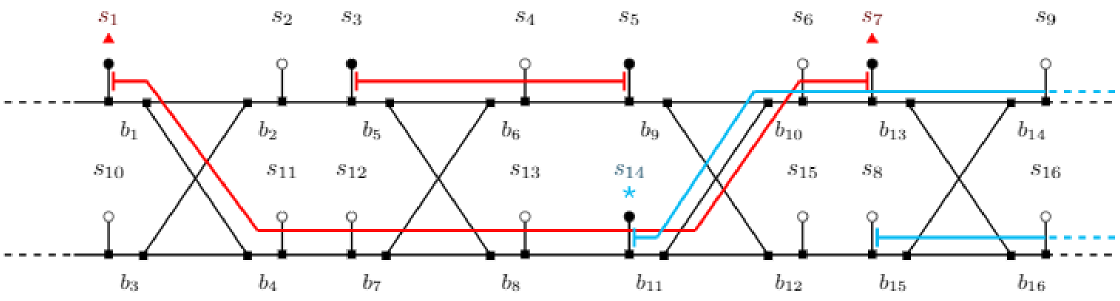


Figure 5.7: Example of paths that can be used to bypass a train parked in a safe place. Signals with the same token are endpoints signals for the paths that allow to bypass the safe place segment marked with the same color.

Definition 5.4. *Let P be a blocking path and Q be a rail path with endpoints s and s'. Then we say that Q* bypasses *P if*

(i) Q does not contain any segment of P,

(ii) there exists a rail path between s and s' that contains (all the segments of) P.

Informally, condition (ii) states that P "lies" between s and s' and condition (i) ensures that there is a path (Q) between s and s' which avoids P. The immediate example of a bypassing route in a network with single-track lines is the presence of a (sufficiently "long") siding that allows meet-pass events between trains. As an example for a multiple-track network, in Figure 5.7, path $P = (s_3, \{s_3, b_5\}, b_5, \{b_5, b_6\}, b_6, \{b_6, s_4\}, s_4, \{s_4, s_5\}, s_5)$ is bypassed by path $Q = (s_1, \{s_1, b_1\}, b_1, \{b_1, b_4\}, b_4, \{b_4, s_{11}\}, s_{11}, \{s_{11}, s_{12}\}, s_{12}, \{s_{12}, b_7\}, b_7, \{b_7, b_8\}, b_8, \{b_8, s_{13}\}, s_{13}, \{s_{13}, s_{14}\}, s_{14}, \{s_{14}, b_{11}\}, b_{11}, \{b_{11}, b_{10}\}, b_{10}, \{b_{10}, s_6\}, s_6, \{s_6, s_7\}, s_7)$.
Therefore, goal *O2* can be implemented by requiring that each train, once stopped in a safe place (i.e. in a blocking path), has to be bypassed by some "short" rail path that will not be occupied by other trains parked in their safe places. The maximum length of such short rail path is a requirement that can be defined from time to time according to some rules fixed by the rail company. For example, a path can be considered short if it does not exceed a given length or, alternatively, if it is composed by at most a given number of segments. All these alternatives can be easily implemented in the proposed model.

In the *SP*-protocol considered here, the rail company directly identifies the set $\mathcal{U}$ of paths that can be used to bypass the trains parked in the safe places, which is thus an input to our model. In the following, for each possible safe place $P \in \mathcal{P}(i)$ for some train $i \in I$, we will denote by U^P the set of paths of $\mathcal{U}$ bypassing P. Such sets U^P can be easily calculated off-line for each P, once the family $\mathcal{U}$ is given. Then, in our model condition *O2* is satisfied if the following holds: for each train i assigned to a safe place $P \in \mathcal{P}(i)$, at least one path Q in U^P is "free", i.e. it does not intersect with any safe place assigned to some train. In other words, there exists a path Q in U^P whose tracks are not occupied or blocked by the train itself or any other train assigned to a safe place on the network. We say that, in this case, Q is *reserved* (because of P).

5.4.3 Implementing goal *O3*

The last goal *O3* does not require further discussion, as it relates to the desirability of train assignment choices and only affects the objective function on the ILP model.

5.4.4 A remedy to infeasibility

Since feasibility and precedence requests among trains can be hard to satisfy, we must consider the case when no feasible assignment of blocking paths to trains exists. In this situation, the SP-protocol introduces the possibility for any train $i \in I$ to hold in the first blocking path P_i^0 of its route, while eventually violating the constraints associated with goal *O2*. We define such trains as *halted*. Note that the solution in which all trains are halted is conflict-free (as trains are physically occupying these positions in the instance) and satisfies the conditions imposed by goal *O1* (thanks to Corollary 5.1). Clearly, this choice will be the least valuable and we want to minimize the number of halted trains. The latter requirement is thus taken into account in the objective function.
It must be noted that when the number of halted trains is larger than one, some conditions must be verified before the safe place assignment can be applied automatically in the case of a partial plan. Namely, the assignment can be carried out if, for every pair of halted trains, these are either trailing, i.e. travelling in the same direction, or have no "interaction", that is they do not risk having conflicts on any resource potentially reachable within the planning horizon. These conditions are often satisfied in North-American freight railways, where traffic is relatively sparse in lower capacity areas in which trains are more prone to being "halted". In denser, typically multiple-track areas instead, the additional capacity available will generally have fewer/no halted trains altogether. We remark that in those rare cases of partial plans in which these conditions do not apply, the SP-protocol requires that the dispatcher overseeing operations take control of the involved trains and "manually" assign trains to their desired safe place (if any).

5.5 A binary linear programming formulation

The purpose of the SPAP is to find an assignment of blocking paths to trains and a conflict-free schedule pursuing objective *O3* while conditions *O1* and *O2* are satisfied. In this section we formally describe a mathematical formulation for it.

5.5.1 Variables and constraints

We recall that, given a set of east-bound and west-bound trains I, for each train i we let $\mathcal{P}(i)$ be the set of its candidate safe places, ordered from the origin to the destination. We denote with P_i^0 the first blocking path on the route of train i (starting position), that is the only assignable blocking path for a halted train. Next, we introduce the following set of binary variables:

$$z_i^P = \begin{cases} 1 & \text{if train } i \text{ is assigned to safe place } P \\ 0 & \text{otherwise} \end{cases} \qquad i \in I,\ P \in \mathcal{P}(i)$$

$$h_i = \begin{cases} 1 & \text{if train } i \text{ is halted in its starting position } P_i^0 \\ 0 & \text{otherwise} \end{cases} \qquad i \in I \quad .$$

A feasible solution for the SPAP implies that each train is either halted in its initial position or assigned to a safe place. This can be ensured thanks to a set of linear constraints:

$$h_i + \sum_{P \in \mathcal{P}(i)} z_i^P = 1 \qquad i \in I \quad . \tag{5.1}$$

Note that the two conditions $h_i = 1$ and $z_i^{P_i^0} = 1$, although train i is physically parked in the same blocking path (the first), actually represent very different situations. Indeed, a halted train does not need to satisfy condition *O2*. Precisely to distinguish these cases, we define $P \in \mathcal{P}(i)$ to be a safe place for i if and only if $z_i^P = 1$.

Clearly, if a train is parked (either halted or in safe place) in a blocking path P, then no other train can be parked in a blocking path P' sharing a segment $r \in P \cap P'$. This is guaranteed by the following two sets of constraints.

$$h_i + \sum_{j \in I} \sum_{P' \in \mathcal{P}(j): r \in P'} z_j^{P'} \leq 1 \qquad i \in I,\ r \in P_i^0 \quad , \tag{5.2}$$

$$\sum_{j \in I} \sum_{P' \in \mathcal{P}(j): r \in P'} z_j^{P'} \leq 1 \qquad i \in I,\ P \in \mathcal{P}(i),\ r \in P \quad . \tag{5.3}$$

Once again, note that, since the input plan is feasible, we have that $P_i^0 \cap P_j^0 = \emptyset$ for any pair of distinct trains i, j.

In order to fulfill objective *O1*, we want the solution to satisfy the condition *ii*) of Theorem 5.2. Therefore, for each oriented arc $(\sigma(P_i), P_j)$ in the current selection $\overline{S}$, we forbid that the conditions $\overline{P}_i \preceq P_i$ and $\overline{P}_j \succeq P_j$ both hold true.

$$h_i + \sum_{P \in \mathcal{P}(i) | P \preceq P_i} z_i^P + \sum_{P \in \mathcal{P}(j) | P \succeq P_j} z_j^P \leq 1 \qquad (\sigma(P_i), P_j) \in \overline{S} \quad . \tag{5.4}$$

Now, to implement condition *O2* we need to introduce the binary variables that define the activation of the paths in $\mathcal{U}$. A path Q is defined as *active* if it intersects with at least a safe place assigned to some train. We have:

$$y^Q = \begin{cases} 1 & \text{if path } Q \text{ is active} \\ 0 & \text{otherwise} \end{cases} \qquad Q \in \mathcal{U} \quad .$$

The following constraints imply that, whenever a train is assigned to safe place P, then at least one path in U^P must be reserved.

$$\sum_{Q \in U^P} y^Q - z_i^P \geq 0 \qquad i \in I,\ P \in \mathcal{P}(i) \quad . \tag{5.5}$$

Then, we need to ensure that all reserved paths do not intersect a safe place assigned to some train.

$$y^Q + z_i^P \leq 1 \qquad Q \in \mathcal{U},\ i \in I,\ P \in \mathcal{P}(i) : P \cap Q \neq \emptyset \quad .$$

Since two assigned safe places cannot share a common segment, we can strengthen the previous constraints as follows:

$$y^Q + \sum_{i \in I} \sum_{P \in \mathcal{P}(i): s \in Q} z_i^P \leq 1 \qquad Q \in \mathcal{U},\ s \in Q \quad . \tag{5.6}$$

Note that there is no analogous of constraints (5.5) and (5.6) involving the variables h instead of variables x, as halted trains do not need to satisfy condition *O2*.

The set of constrains (5.1)-(5.6), plus the binary stipulation on the variables z, h, y, defines the feasible set of solutions to the SPAP-ILP model. We underline that this set is non-empty since the solution $h_i = 1$ for all $i \in I$ (*all halted*) is always feasible.

5.5.2 Objective function

To measure the cost of a feasible solution, we consider two components. First recall that we want the trains to be parked in the farthest possible safe place (objective *O3*). To this end, for each train i, and each potential safe place $P \in \mathcal{P}(i)$ we associated a cost ς_i^P that is proportional to the distance of P from the final destination of i. In other words, $\varsigma_i^{P'} < \varsigma_i^P$ for any $P' \in \mathcal{P}(i) : P \prec P'$. Then, we have a cost $\bar{\varsigma}_i$ for every halted train i. Since we want to minimize the number of halted trains, we choose $\bar{\varsigma}_i >> \varsigma_i^P$ for any $P \in \mathcal{P}(i)$. Thus, the objective function of the SPAP-ILP writes as the following cost minimization:

$$\min \sum_{i \in I} \bar{\varsigma}_i \, h_i + \sum_{i \in I} \sum_{P \in \mathcal{P}(i)} \varsigma_i^P \, z_i^P . \tag{5.7}$$

Modelling the cost of a solution in this way allows us to achieve goal *O3*.
As anticipated while defining the three objectives, this formulation can be modified to incorporate different requirements. For example, we may discourage to stop a passengers train i on a potential safe place $\bar{P}$ where there is no available platform by increasing costs $\varsigma_i^{\bar{P}}$. Instead, consider a train having its starting position in a busy network portion, we may foster the choice of a safe place on its route choosing a very large value for the coefficient $\bar{\varsigma}_i$.

To conclude, note that all but one (family of) constraints are shared with the integer programming formulation of a Maximal Stable Set Problem (MSSP), namely the *node-packing formulation* (see Section 2.3.3). Although ILPs are in general hard to solve in practice and in theory (see [70]), the associated integer polytope has been much investigated in the literature, and most commercial solvers can exploit the special structure of its constraint matrix. The next section shows indeed that all instances can be solved in very short computing time.

5.6 Computational experiments

We now present the results obtained by applying the SPAP formulation on sixty realistic instances based on data provided by a Class I North-American railroad. The underlying

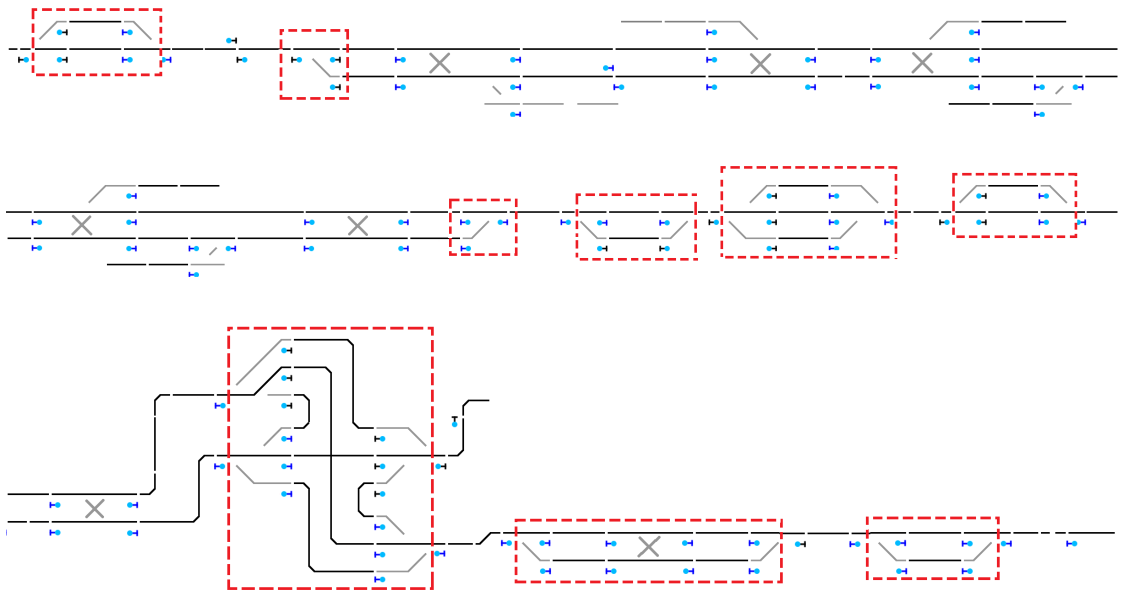


Figure 5.8: Three sample portions of railway network. Stations (within dotted lines) can be connected by a single or multiple (bidirectional) tracks. Stations can have a different number of tracks. There can be junctions between different lines.

network alternates single-track and multiple-track stretches. Tracks are bidirectional. The number of routes within a station may vary: most stations have two bidirectional tracks, but some stations may have three or more of them. Data come from a wide area of the railway network, which comprises almost 300 stations. Note that not all stations appear in every instances, as some stations are not traversed by any train during the time interval considered. In Figure 5.8 we show three sample portions of the net as examples. In particular, note that the network also includes junctions between different lines. Distance between adjacent stations may vary considerably, from 6000 to 100000 ft in single-track areas, while in multiple-track areas they are usually shorter (mainly spanning from 2000 to 20000 ft). The trains in the network are mostly freight trains.

5.6.1 Instances

All instances considered refer to a network with both single and multiple track areas. As already mentioned, here the set $\mathcal{U}$ of paths that can be used to bypass the trains parked in the safe places is given explicitly. Moreover, given the planned route of each train i, we can easily compute the set $\mathcal{P}(i)$ of the potential safe places of i: it is sufficient to enumerate all blocking paths traversed by i along its scheduled route. Then, for each $P \in \mathcal{P}(i)$, we

Instance	$\lvert I\rvert$	$\mathbb{E}[U^P]$	κ	$\mathbb{E}[\mathcal{P}(i)]$	$i_{\mathcal{P}(i)=0}$	$i_{\mathcal{P}(i)>0}$	$i_{\mathcal{P}(i)>1}$	$i_{\mathcal{P}(i)>5}$	$i_{\mathcal{P}(i)>10}$	$i_{\mathcal{P}(i)>15}$
$\mathcal{I}_1$	7	3	15	7	0 %	100 %	100 %	43 %	29 %	0 %
$\mathcal{I}_2$	4	1	13	9	0 %	100 %	100 %	100 %	25 %	0 %
$\mathcal{I}_3$	6	2	15	8	0 %	100 %	100 %	83 %	33 %	0 %
$\mathcal{I}_4$	4	1	13	8	0 %	100 %	100 %	50 %	25 %	0 %
$\mathcal{I}_5$	5	1	14	10	0 %	100 %	100 %	100 %	40 %	0 %
$\mathcal{I}_6$	4	2	15	10	25 %	75 %	75 %	75 %	75 %	0 %
$\mathcal{I}_7$	5	2	15	9	0 %	100 %	100 %	100 %	40 %	0 %
$\mathcal{I}_8$	4	3	15	10	0 %	100 %	100 %	75 %	50 %	0 %
$\mathcal{I}_9$	4	2	12	9	0 %	100 %	100 %	100 %	25 %	0 %
$\mathcal{I}_{10}$	5	2	11	6	0 %	100 %	80 %	60 %	0 %	0 %

Table 5.1: Dataset $\mathcal{S}_1$: relevant statistics of test instances.

construct $U^P \subseteq \mathcal{U}$ by considering all paths of $\mathcal{U}$ that bypass P (see goal *O2* in Section 5.4). Observe that, in a directed graph G, a u-v path Q bypasses P if $G \setminus Q$ admits a u-v path that contains P.

Our test-bed allows us to highlight the proactive and reactive aspect of our formulation. As discussed in the introduction, we encounter the SPAP mainly in two application contexts:

(i) *disruption management*, when a major blockage occurs in the network;

(ii) *partial plan recovering*, when an ATMS fails to produce a solution for the entire planning horizon.

Accordingly, we generated two sets of instances: $\mathcal{S}_1$ and $\mathcal{S}_2$. Some statistics for the two dataset are summarized respectively in Table 5.1 and Table 5.2. For each instances we present the number of trains ($\lvert I\rvert$), the average number of bypassing paths per safe place position ($\mathbb{E}[U^P]$), the average number of safe places available for each train ($\mathbb{E}[\mathcal{P}(i)]$) and the maximum number of available safe place positions among all trains (defined as κ). Moreover, for every instance we report the percentage of trains that have either no safe place or more than $\{0, 1, 5, 10, 15\}$ potential safe place locations (columns $i_{\mathcal{P}(i)=0}$, $i_{\mathcal{P}(i)>0}$, $i_{\mathcal{P}(i)>1}$, $i_{\mathcal{P}(i)>5}$, $i_{\mathcal{P}(i)>10}$, $i_{\mathcal{P}(i)>15}$).

The first dataset $\mathcal{S}_1 = \{\mathcal{I}_1, ..., \mathcal{I}_{10}\}$ consists of ten instances related to the case (i). Each instance relates to a small set of trains approaching an area affected by a disruption (on average 4.8 trains per instance), in a limited time frame of two hours. For each instance, the average number of potential safe place per train is 8.6. Note that trains have at least a safe place position in their scheduled path in all instances except one. Indeed, instance $\mathcal{I}_6$ presents a train with no safe place position available. Of course, this train will be halted in the optimal solution (see Section 5.6.3). Moreover, due to the presence of a major disruption in the net we can count on a very low number of bypassing path per safe place position

Instance	$\lvert I \rvert$	$\mathbb{E}[U^P]$	κ	$\mathbb{E}[\mathcal{P}(i)]$	$i_{\mathcal{P}(i)=0}$	$i_{\mathcal{P}(i)>0}$	$i_{\mathcal{P}(i)>1}$	$i_{\mathcal{P}(i)>5}$	$i_{\mathcal{P}(i)>10}$	$i_{\mathcal{P}(i)>15}$
$\mathcal{I}_{11}$	54	7	14	7	0 %	100 %	98 %	76 %	9 %	0 %
$\mathcal{I}_{12}$	47	7	23	12	0 %	100 %	98 %	81 %	62 %	34 %
$\mathcal{I}_{13}$	50	9	19	10	0 %	100 %	96 %	88 %	40 %	6 %
$\mathcal{I}_{14}$	45	7	23	13	2 %	98 %	96 %	87 %	64 %	31 %
$\mathcal{I}_{15}$	50	7	17	8	0 %	100 %	100 %	76 %	14 %	2 %
$\mathcal{I}_{16}$	54	7	22	10	2 %	98 %	98 %	89 %	48 %	9 %
$\mathcal{I}_{17}$	46	7	11	6	2 %	98 %	98 %	57 %	0 %	0 %
$\mathcal{I}_{18}$	54	8	19	10	0 %	100 %	98 %	89 %	57 %	7 %
$\mathcal{I}_{19}$	53	7	9	4	0 %	100 %	98 %	11 %	0 %	0 %
$\mathcal{I}_{20}$	53	7	7	3	0 %	100 %	98 %	2 %	0 %	0 %
$\mathcal{I}_{21}$	52	7	21	11	0 %	100 %	96 %	83 %	56 %	19 %
$\mathcal{I}_{22}$	43	7	21	12	0 %	100 %	100 %	88 %	67 %	21 %
$\mathcal{I}_{23}$	42	8	20	12	0 %	100 %	100 %	86 %	71 %	24 %
$\mathcal{I}_{24}$	44	6	21	11	0 %	100 %	98 %	86 %	55 %	23 %
$\mathcal{I}_{25}$	59	3	12	6	2 %	98 %	98 %	54 %	3 %	0 %
$\mathcal{I}_{26}$	51	2	12	6	2 %	98 %	96 %	65 %	2 %	0 %
$\mathcal{I}_{27}$	47	3	13	6	0 %	100 %	98 %	49 %	4 %	0 %
$\mathcal{I}_{28}$	46	2	18	11	0 %	100 %	100 %	89 %	57 %	7 %
$\mathcal{I}_{29}$	41	2	19	11	0 %	100 %	100 %	85 %	63 %	10 %
$\mathcal{I}_{30}$	57	3	19	11	0 %	100 %	98 %	91 %	65 %	7 %
$\mathcal{I}_{31}$	53	2	19	10	0 %	100 %	98 %	83 %	51 %	9 %
$\mathcal{I}_{32}$	57	3	21	11	0 %	100 %	96 %	86 %	61 %	12 %
$\mathcal{I}_{33}$	43	3	22	12	0 %	100 %	100 %	93 %	63 %	16 %
$\mathcal{I}_{34}$	45	3	26	11	0 %	100 %	100 %	82 %	64 %	9 %
$\mathcal{I}_{35}$	53	3	23	9	0 %	100 %	94 %	75 %	42 %	6 %
$\mathcal{I}_{36}$	48	2	14	6	0 %	100 %	98 %	56 %	2 %	0 %
$\mathcal{I}_{37}$	40	3	24	12	0 %	100 %	100 %	90 %	63 %	20 %
$\mathcal{I}_{38}$	42	3	25	10	0 %	100 %	100 %	81 %	45 %	12 %
$\mathcal{I}_{39}$	49	5	17	11	2 %	98 %	98 %	86 %	57 %	4 %
$\mathcal{I}_{40}$	40	5	19	11	0 %	100 %	98 %	88 %	63 %	8 %
$\mathcal{I}_{41}$	45	5	20	11	2 %	98 %	98 %	89 %	58 %	7 %
$\mathcal{I}_{42}$	49	5	19	10	0 %	100 %	100 %	90 %	57 %	2 %
$\mathcal{I}_{43}$	59	6	17	11	3 %	97 %	97 %	92 %	66 %	3 %
$\mathcal{I}_{44}$	46	6	17	11	2 %	98 %	98 %	91 %	72 %	7 %
$\mathcal{I}_{45}$	49	5	18	11	2 %	98 %	98 %	90 %	67 %	6 %
$\mathcal{I}_{46}$	50	6	13	6	0 %	100 %	100 %	56 %	2 %	0 %
$\mathcal{I}_{47}$	23	5	6	2	13 %	87 %	52 %	0 %	0 %	0 %
$\mathcal{I}_{48}$	29	2	13	4	14 %	86 %	76 %	48 %	3 %	0 %
$\mathcal{I}_{49}$	11	3	9	5	0 %	100 %	82 %	45 %	0 %	0 %
$\mathcal{I}_{50}$	31	2	11	5	0 %	100 %	90 %	42 %	0 %	0 %
$\mathcal{I}_{51}$	31	2	14	6	0 %	100 %	90 %	61 %	6 %	0 %
$\mathcal{I}_{52}$	29	2	10	5	0 %	100 %	86 %	45 %	0 %	0 %
$\mathcal{I}_{53}$	26	2	15	7	0 %	100 %	85 %	65 %	27 %	0 %
$\mathcal{I}_{54}$	32	2	11	6	0 %	100 %	100 %	69 %	0 %	0 %
$\mathcal{I}_{55}$	27	5	19	12	0 %	100 %	100 %	85 %	70 %	19 %
$\mathcal{I}_{56}$	44	2	10	5	2 %	98 %	93 %	55 %	0 %	0 %
$\mathcal{I}_{57}$	33	1	14	7	0 %	100 %	85 %	67 %	6 %	0 %
$\mathcal{I}_{58}$	15	2	12	6	0 %	100 %	100 %	60 %	7 %	0 %
$\mathcal{I}_{59}$	16	2	11	4	6 %	94 %	81 %	38 %	0 %	0 %
$\mathcal{I}_{60}$	27	2	10	3	11 %	89 %	70 %	15 %	0 %	0 %

Table 5.2: Dataset $\mathcal{S}_2$: relevant statistics of test instances.

(from 1 to 3).
The second set $\mathcal{S}_2 = \{\mathcal{I}_{11}, ..., \mathcal{I}_{60}\}$ refers to the case (*ii*). It includes fifty instances for which the ATMS only provides a plan with a time horizon that is smaller than the default planning horizon. The number of trains per instance considered ranges from a minimum of 11 to a maximum of 59 trains (42.6 on average). Here, the average number of possible safe place positions per train varies greatly from instance to instance and from train to train, and ranges from 2 to 13 (8.4 on average). Finally, the number of bypassing paths per safe place position for this dataset is on average 4.3.

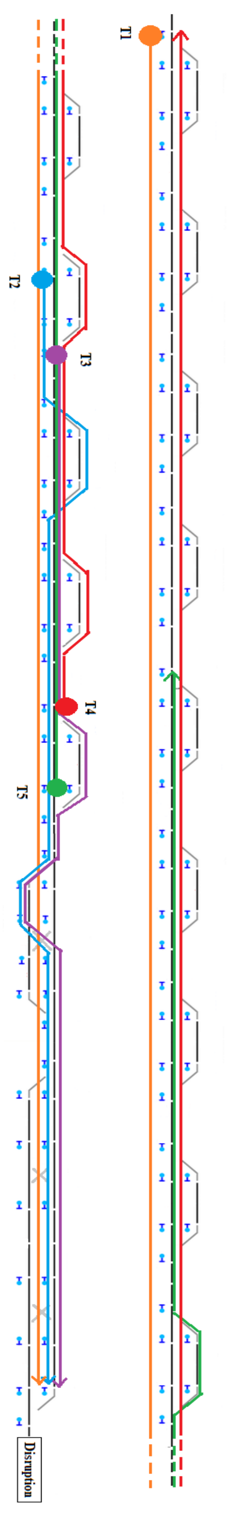
5.6.2 Settings

Since we want to park the trains as close as possible to their final destination (goal *O3*), we let $\varsigma_i^P = \kappa \cdot (\kappa - \rho_i^P)$, where ρ_i^P indicates the index position of the safe place segment P along the route of train i (the lower the index, the closer P is to P_i^0). Furthermore, as our main objective is to minimize the number of halted trains, in the numerical experiments we set the value $\bar{\varsigma}_i = \kappa^2 \cdot |I|/2$.
The experiments are performed using an x64 MS Windows 10 machine with an Intel (R) Core (TM) i7-10510U CPU and 16 GB of RAM. The problem is implemented in Java and uses *IBM ILOG CPLEX v12.10* (all default settings) as MIP optimizer.

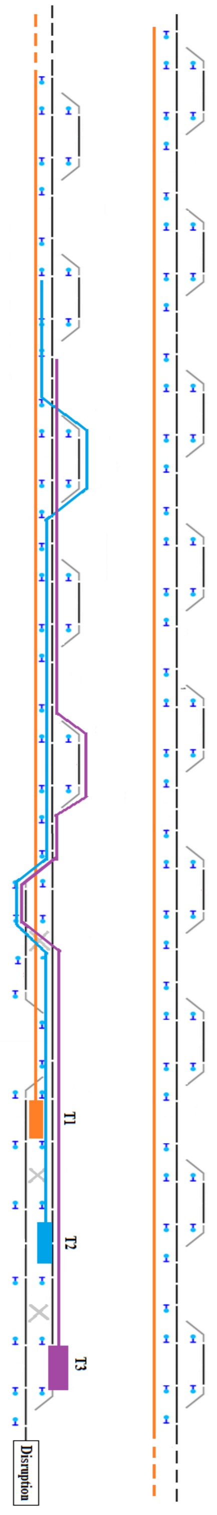
5.6.3 Computational results

Figure 5.9 gives a graphical representation of the input and output of the SPAP for instance $\mathcal{I}_7$. As input (see Figure 5.9a) we have the scheduled plan for each train belonging to the instance and its current position in the net (denoted with solid colored points). Three out of five trains are obstructed by a disruption and need to be assigned to a safe place position, while the remaining trains are moving away from the blockage. The optimal safe place assignment for the instance (in solid colored rectangles) is shown in Figure 5.9b.

In Table 5.3 and Table 5.4, we report the computational results obtained solving our formulation on the test instances of the set $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. The rows of the tables are associated with the instances of the corresponding dataset. Then, in the first columns, we give some detail about the instance - reporting the number of trains ($|I|$) and the average number of safe places per train ($\mathbb{E}[\mathcal{P}(i)]$) - and the size of the corresponding ILP model, with the number of variables ($\#Var$) and constraints ($\#Con$). Finally, in columns $\#Halted$ and *Time (ms)* we report respectively the number of halted trains in the optimal solution and the CPU time needed to solve the formulation (expressed in milliseconds).



(a) Schedule.



(b) Safe places assignment.

Figure 5.9: Instance $\mathcal{I}_7$: the schedule for each train (5.9a) and the safe places assignment in the optimal solution (5.9b).

Instance	$\lvert I \rvert$	$\mathbb{E}[\mathcal{P}(i)]$	$\#Var$	$\#Con$	$\#Halted$	*Time (ms)*
$\mathcal{I}_1$	7	7	278	550	0	9
$\mathcal{I}_2$	4	9	139	288	0	7
$\mathcal{I}_3$	6	8	189	425	1	8
$\mathcal{I}_4$	4	8	156	232	0	6
$\mathcal{I}_5$	5	10	221	367	0	7
$\mathcal{I}_6$	4	10	148	302	1	3
$\mathcal{I}_7$	5	9	236	546	0	9
$\mathcal{I}_8$	4	10	206	390	0	3
$\mathcal{I}_9$	4	9	168	264	0	3
$\mathcal{I}_{10}$	5	6	149	268	1	3

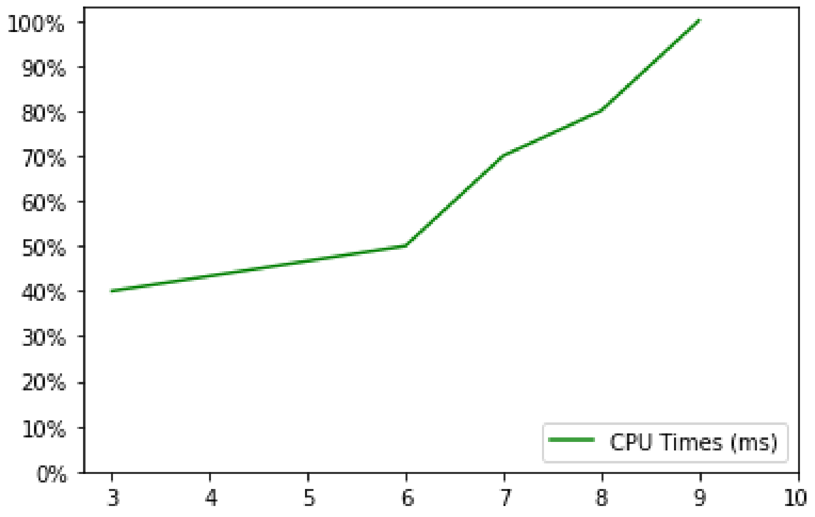
Table 5.3: Dataset $\mathcal{S}_1$: metrics and computational times results.

We first observe that, for the instances of $\mathcal{S}_1$, the CPU computational times are negligible. While about 90% of the instances of $\mathcal{S}_2$ require around 200 milliseconds to be solved, only five need more than 300 milliseconds to produce the final safe place assignments. This proves that the proposed method can be very efficiently applied in a real-time setting. The cumulative distribution of the computational times of the $\mathcal{S}_1$ and $\mathcal{S}_2$ instances are presented in Figure 5.10. We also want to remark that all instances are solved at the root node of the Branch&Bound approach implemented by the solver used. This proves the good quality of the linear relaxation of the ILP model, that is the strength of our formulation.

In Figure 5.11, we show, for the two set $\mathcal{S}_1$ and $\mathcal{S}_2$, the cumulative distribution of the instances according to the number of halted trains in the optimal solution. In particular, for dataset $\mathcal{S}_1$, the optimal solutions of about 70% of the instances do not halt any train. This is somehow surprising, since such instances are derived from network disruption situations. In contrast, this percentage reaches about 50% when considering $\mathcal{S}_2$ instances. These data could be explained by the fact that the number of involved trains is much smaller in the $\mathcal{S}_1$ instances, which in turn implies: *i*) that the number of interactions (meets-pass events), hence of precedence constraints is smaller and *ii*) that the network is less congested, which increases the probability of satisfying condition *O2*. Besides, in most of the instances of $\mathcal{S}_2$ (i.e. instances derived from a partial plan), the trains set as halted by the optimal solution have a very small number of available safe places. We also point that, aside from instances $\mathcal{I}_{47}$ and $\mathcal{I}_{48}$, in all other cases the halted trains comply with the conditions of either been trailing trains or not having potential interactions with each other. As described in Section 5.4, in the other two special cases the dispatchers will monitor the output of the SP-protocol and "safely" assign a position to the involved trains.

Instance	$\|I\|$	$\mathbb{E}[\mathcal{P}(i)]$	$\#Var$	$\#Con$	$\#Halted$	*Time (ms)*
$\mathcal{I}_{11}$	54	7	2873	6628	0	429
$\mathcal{I}_{12}$	47	12	3261	9020	0	246
$\mathcal{I}_{13}$	50	10	3347	8648	0	103
$\mathcal{I}_{14}$	45	13	3027	8265	1	136
$\mathcal{I}_{15}$	50	8	2835	6683	0	538
$\mathcal{I}_{16}$	54	10	3203	8552	1	165
$\mathcal{I}_{17}$	46	6	2408	5188	1	44
$\mathcal{I}_{18}$	54	10	3176	8637	0	159
$\mathcal{I}_{19}$	53	4	1828	3674	0	600
$\mathcal{I}_{20}$	53	3	1920	3692	0	42
$\mathcal{I}_{21}$	52	11	3946	10596	0	180
$\mathcal{I}_{22}$	43	12	3911	10149	0	156
$\mathcal{I}_{23}$	42	12	4071	11170	0	143
$\mathcal{I}_{24}$	44	11	3221	7904	0	182
$\mathcal{I}_{25}$	59	6	1231	3455	1	314
$\mathcal{I}_{26}$	51	6	1149	3085	1	308
$\mathcal{I}_{27}$	47	6	1112	2732	0	237
$\mathcal{I}_{28}$	46	11	1987	4229	0	31
$\mathcal{I}_{29}$	41	11	2036	4271	0	32
$\mathcal{I}_{30}$	57	11	2654	6685	0	70
$\mathcal{I}_{31}$	53	10	2174	5105	0	47
$\mathcal{I}_{32}$	57	11	2502	6895	0	91
$\mathcal{I}_{33}$	43	12	2291	5320	0	42
$\mathcal{I}_{34}$	45	11	1956	5779	1	75
$\mathcal{I}_{35}$	53	9	2147	5777	0	62
$\mathcal{I}_{36}$	48	6	1479	2863	0	28
$\mathcal{I}_{37}$	40	12	1864	5394	0	47
$\mathcal{I}_{38}$	42	10	2075	4933	0	45
$\mathcal{I}_{39}$	49	11	2686	7075	1	135
$\mathcal{I}_{40}$	40	11	2458	5831	0	77
$\mathcal{I}_{41}$	45	11	2467	6469	1	121
$\mathcal{I}_{42}$	49	10	2644	7291	0	100
$\mathcal{I}_{43}$	59	11	3079	8742	2	139
$\mathcal{I}_{44}$	46	11	2610	7128	1	138
$\mathcal{I}_{45}$	49	11	2765	7398	1	157
$\mathcal{I}_{46}$	50	6	2091	4529	0	61
$\mathcal{I}_{47}$	23	2	407	743	7	6
$\mathcal{I}_{48}$	29	4	730	1183	5	19
$\mathcal{I}_{49}$	11	5	298	621	2	85
$\mathcal{I}_{50}$	31	5	997	1588	2	12
$\mathcal{I}_{51}$	31	6	1041	1747	2	22
$\mathcal{I}_{52}$	29	5	707	1423	6	18
$\mathcal{I}_{53}$	26	7	1169	2064	3	16
$\mathcal{I}_{54}$	32	6	1073	1807	2	13
$\mathcal{I}_{55}$	27	12	2021	6325	1	38
$\mathcal{I}_{56}$	44	5	1229	2070	4	27
$\mathcal{I}_{57}$	33	7	1039	1847	3	18
$\mathcal{I}_{58}$	15	6	559	973	0	16
$\mathcal{I}_{59}$	16	4	405	684	2	11
$\mathcal{I}_{60}$	27	3	596	983	12	8

Table 5.4: Dataset $\mathcal{S}_2$: metrics and computational times results.



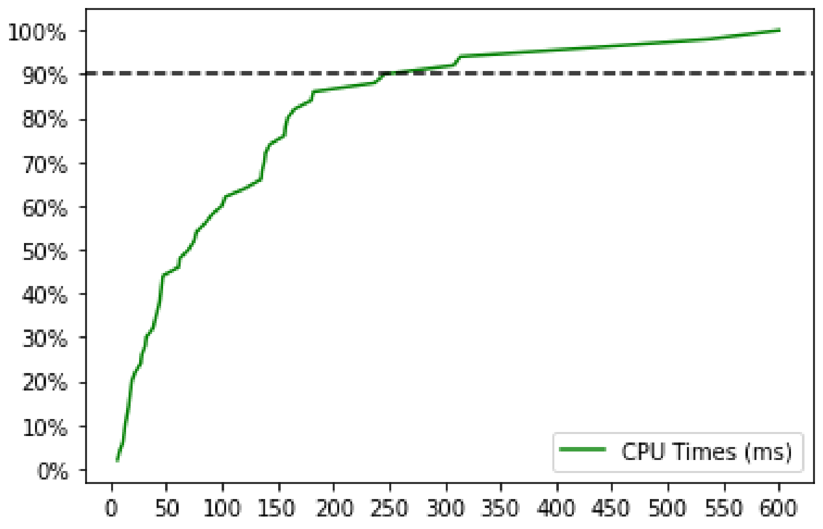
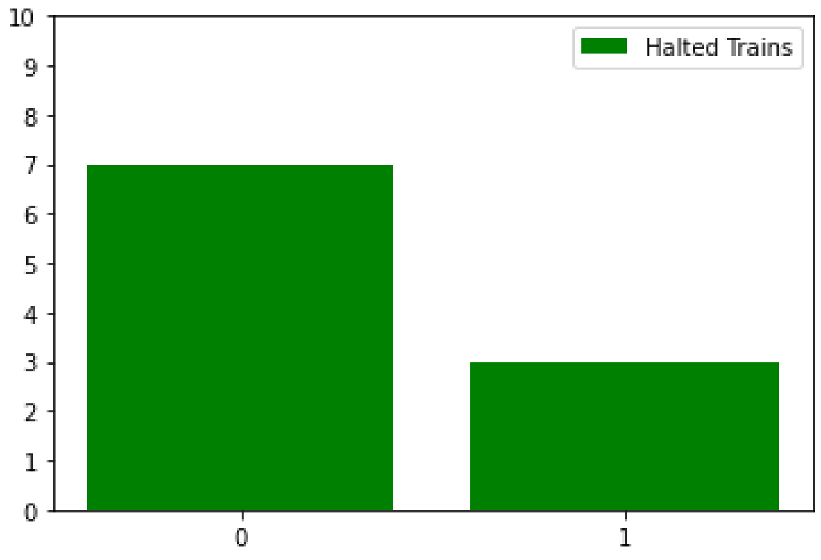


Figure 5.10: Computational times cumulative distribution for dataset $\mathcal{S}_1$ (above) and $\mathcal{S}_2$ (below).



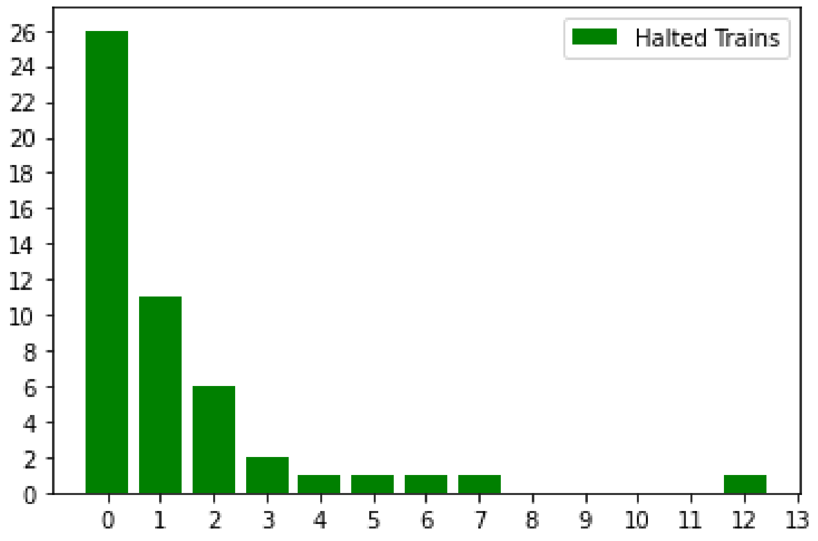


Figure 5.11: Distributions of halted trains per instance for dataset $\mathcal{S}_1$ (above) and $\mathcal{S}_2$ (below).

5.7 Conclusions

In this chapter, we present a non-naive formulation for the SPAP, a problem that arises in the context of railway disturbance and disruption management. Trains must be assigned a location to "safely" park under certain circumstances, such as when a disruption occurs on the network or when the ATMS produces a plan that can only partially be carried out. In Section 5.6 we showed that realistic instances can be solved in a very small amount of time, giving evidence that the proposed method can be effectively applied also in a practical setting, where (quasi) real-time performance is required. It is perhaps worth noting that the special structure of the model could be among the reasons behind the extremely low computational time. Indeed, as we observed in Section 5.5, the formulation shares all constraints except one with a MSSP, it therefore presents a structure that most math programming solvers are able to exploit to improve performances.

As possible future research directions, we consider the case of modelling tracks with a capacity of two or more trains as well as the possibility of partially rerouting some of the trains. Then, we can model the possibility to stop trains on block sections that do not appear in their original schedules. Thus, in this scenario we aim to further reduce the number of halted trains, combining with the SPAP the resolution of a train re-scheduling problem after the end of the disruption. Moreover, by investigating the disparate requirements and policies of different rail operators, more complex objectives can be addressed. The relative closeness to the much-studied MSSP might also inspire alternative solution methods and further work to strengthen the formulation.

Chapter 6

Conclusions

6.1 Main findings

In this thesis we demonstrated the necessity of the use of rigorous mathematics tools in train dispatching. We provided an overview of the state of the art in the field of Real-time Train Re-scheduling and proposed a new re-scheduling model with the aim of overcoming the main drawbacks of the standard optimization models used. We then focused on a special aspect of the disruption management: finding locations, named *safe places*, where to park trains in case of a major perturbation of the system.
The research subjects are both innovative and of high relevance in the railway sector and beyond. Particular attention should be given to the designed re-scheduling procedure as it can be applied to any job-shop scheduling problem. On the other hand, the topic of assigning safe locations (and the related mathematical formulation) has been little addressed in the literature and not so many papers can be found. The outputs of the present study are two different optimization models that trace two opposite but complementary issues of the real-time train re-scheduling: the *reactive scheduling*, allowing to efficiently recover from unexpected events, and the *proactive planning*, aiming at building robust railways systems.

On the reactive side, we designed, implemented and evaluated a novel technique to re-schedule trains and generate a consistent and conflict-free timetable. We relied on a recently introduced paradigm, the Dynamic Discretization Discovery (DDD). We provided a tailor-made implementation of the algorithm that iteratively solves a series of pure binary linear problems until a feasible solution has been found. We compared the result obtained on 20 real instances with the performance of the two most widely used optimization models, namely the big-M and time-indexed formulations. The results gave evidence that the

procedure can be used efficiently in a dynamic environment and has great potential to be adopted for real-time applications.
In the context of railway disturbance and disruption management, we have defined and mathematically formulated the *Safe Place Assignment Problem* (SPAP). Such a problem may be addressed in two cases: when a disruption occurs on the network or when an Advanced Traffic Management System (ATMS) is not able to produce a complete feasible plan and ends up with a schedule that spans a shorter time horizon. Thus, its deployment area covers both a reactive and proactive aspect. The model has been developed in close cooperation with a large Class I North-American railroad company and the results have been very welcomed by dispatchers. We tested the binary linear program we devised on two real-world data sets which allowed us to highlight the suitability of the model for both facets of disruption management. The computational experiments indicated that a very small amount of time is needed to solve the model, demonstrating its great potential for application in an on-line context.

Note that the implementation of the two procedures developed strongly depends on the disturbance/disruption length estimation and the knowledge on the exact location of all rolling stocks when the unwanted event occurs. Up to date and accurate data are therefore an essential prerequisite for their correct application in real life.

6.2 Recommendations for future research

Since both research projects debated in this thesis are off the beaten track, they leave a wide room for further improvements and offer different research directions.

A first step towards the improvement of the DDD technique is to investigate on how the discover of new time points to refine the intervals can be improved. In fact, this aspect can be particularly time consuming since at each iteration we may have a non-trivial number of candidate time points to add for a single resource. This is especially the case of a conflicting resource involving more than two trains. One might want to find the best partition that eliminates all possible pair-wise conflicting events at once, avoiding unnecessary iterations. Moreover, it seems that the difficulty of this problem may vary significantly depending on the different scenarios, criteria and conditions. In particular, a study on the influence of the number of potentially conflicting resource pairs should be addressed. A potential time-saving procedure introducing a concurrent use of macro-micro approaches to model the network can also be tested. For example, one can explicitly model a conflicting pair only when it shows an incompatibility, and leave the associated resource in an aggregate form when them do not interfere with the feasibility of the schedule. On the other hand, more details can be considered in the model, such as trains cancellations, re-routing, optimal train speed and efficient driving. Finally, the dynamic potential of the proposed technique can also be exploited transforming the problem in a Maximum Satisfiability (MaxSAT) problem and using the same idea of abstraction refinement with a unary-number-like representations of time-domain variables.

There are also many insights for future research on the SPAP. As for the DDD case, the model can include more details, such as capacity on tracks and at the stations. Re-routing trains through a safe location that does not appear on the original planned route could also be a way to make the network less congested on a disruption occurrence. Furthermore, the strength of the binary formulation and its special structure resembling a maximum stable set should be studied. Lastly, we underline that current models and systems usually disregard passengers discomfort and focus their attention on minimizing the overall delay or the number of delayed trains. Future research should therefore include, for both projects, the passengers role and their expectations.

Bibliography

[1] J. Akker, C. Hurkens, and M. Savelsbergh. Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing*, 12:111–124, 05 2000.

[2] ALSTOM. Advanced computer-aided dispatching for centralized train control, 2000. `https://www.yumpu.com/en/document/read/8277621/tms-alstom` [Online. Last Visited: 29 December, 2021].

[3] D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *INFORMS Journal on Computing*, 3:149–156, 05 1991.

[4] Aurizon - Railway Gazette International. Aurizon deploys ge transportation movement planner tool, 2016. `https://www.railwaygazette.com/australasia/aurizon-deploys-ge-transportation-movement-planner-tool/42988.article` [Online. Last Visited: 29 December, 2021].

[5] S. Azem, R. Aggoune, and S. Dauzère-Pérès. Disjunctive and time-indexed formulations for non-preemptive job shop scheduling with resource availability constraints. In *2007 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 787 – 791, 01 2008.

[6] E. Balas. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations research*, 17(6):941–957, 1969.

[7] E. Balas. *On the facial structure of scheduling polyhedra*, pages 179–218. Springer Berlin Heidelberg, Berlin, Heidelberg, 1985.

[8] E. Balas. *Disjunctive Programming.* Springer Publishing Company, Incorporated, 2018.

[9] BaneNOR. The digital railway – bane nor signs contract with thales, 2018. `https://www.banenor.no/Prosjekter/prosjekter/ertms/innhold/2018/`

`the-digital-railway--bane-nor-signs-contract-with-thales/` [Online. Last Visited: 29 December, 2021].

[10] BaneNOR. Bane nor action cards, 2021. `https://orv.banenor.no/orv/doku.php?id=sidebar&id=avvikshandtering:aksjonskort/` [Online. Last Visited: 29 December, 2021].

[11] P. Baptiste and R. Sadykov. On scheduling a single machine to minimize a piecewise linear objective function: A compact MIP formulation. *Naval Research Logistics (NRL)*, 56(6):487–502, September 2009.

[12] M. Barah, A. Seifi, and J. Ostrowski. Decomposing the train-scheduling problem into integer-optimal polytopes. *Transportation Science*, 03 2019.

[13] M. A. Bender, J. T. Fineman, S. Gilbert, and R. E. Tarjan. A new approach to incremental cycle detection and related problems. *ACM Transactions on Algorithms (TALG)*, 12(2):14, 2016.

[14] A. Bettinelli, A. Santini, and D. Vigo. A real-time conflict solution algorithm for the train rescheduling problem. *Transportation Research Part B: Methodological*, 106:237–265, 2017.

[15] N. Bešinović. Resilience in railway transport systems: a literature review and research agenda. *Transport Reviews*, 40(4):457–478, 2020.

[16] N. Beinovi, R. Tang, Z. Lin, R. Liu, T. Tang, L. De Donato, V. Vittorini, Z. Wang, F. Flammini, M. Pappaterra, and R. Goverde. Rails project deliverable d1.2: Summary of existing relevant projects and state-of-the-art of ai application in railways. Technical report, 03 2021.

[17] S. Binder, Y. Maknoon, and M. Bierlaire. The multi-objective railway timetable rescheduling problem. *Transportation Research Part C: Emerging Technologies*, 78:78–94, 2017.

[18] M. Boccia, C. Mannino, and I. Vasilyev. The dispatching problem on multitrack territories: Heuristic approaches based on mixed integer linear programming. *Networks*, 62(4):315–326, 2013.

[19] N. Boland, R. Clement, and H. Waterer. A Bucket Indexed Formulation for Non-preemptive Single Machine Scheduling Problems. *INFORMS Journal on Computing*, 28(1):14–30, February 2016.

[20] N. Boland, M. Hewitt, L. Marshall, and M. Savelsbergh. The continuous-time service network design problem. *Operations research*, 65(5):1303–1321, 2017.

[21] N. Boland, M. Hewitt, L. Marshall, and M. Savelsbergh. The price of discretizing time: a study in service network design. *EURO Journal on Transportation and Logistics*, 8(2):195–216, June 2019.

[22] N. L. Boland and M. W. Savelsbergh. Perspectives on integer programming for time-dependent models. *Top*, 27(2):147–173, 2019.

[23] S. Bollapragada, R. Markley, H. Morgan, E. Telatar, S. Wills, M. Samuels, J. Bieringer, M. Garbiras, G. Orrigo, F. Ehlers, et al. A novel movement planner system for dispatching trains. *Interfaces*, 48(1):57–69, 2018.

[24] R. Borndörfer, T. Klug, L. Lamorgese, C. Mannino, M. Reuther, and T. Schlechte. Recent success stories on integrated optimization of railway systems. *Transportation Research Part C: Emerging Technologies*, 74:196 – 211, 2017.

[25] C. Borraz-Sánchez, D. Klabjan, and A. Uygur. An approach for the railway multiterritory dispatching problem. *Transportation Science*, 54(3):721–739, 2020.

[26] J. Baewicz, E. Pesch, and M. Sterna. The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, 127(2):317–331, 2000.

[27] V. Cacchiani, D. Huisman, M. Kidd, L. Kroon, P. Toth, L. Veelenturf, and J. Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15 – 37, 05 2014.

[28] V. Cacchiani and P. Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3):727–737, 2012.

[29] V. Cacchiani and P. Toth. Robust train timetabling. In *Handbook of Optimization in the Railway Industry*, pages 93–115. Springer, 2018.

[30] L. Cadarso, Ángel Marín, and G. Maróti. Recovery of disruptions in rapid transit networks. *Transportation Research Part E: Logistics and Transportation Review*, 53:15–33, 2013.

[31] G. Caimi, F. Chudak, M. Fuchsberger, M. Laumanns, and R. Zenklusen. A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation Science*, 45:212–227, 05 2011.

[32] G. Caimi, M. Fuchsberger, M. Laumanns, and M. Lüthi. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research*, 39(11):2578–2593, 2012.

[33] F. Chu and A. Oetting. Modeling capacity consumption considering disruption program characteristics and the transition phase to steady operations during disruptions. *Journal of Rail Transport Planning & Management*, 3(3):54–67, 2013. Robust Rescheduling and Capacity Use.

[34] V. Chvatal. *Linear programming*. Macmillan, 1983.

[35] M. Conforti, G. Cornuejols, and G. Zambelli. *Integer Programming*. Springer Publishing Company, Incorporated, 2014.

[36] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley &amp; Sons, Inc., USA, 1998.

[37] G. T. Coor. Optimization and simulation models applied to railway traffic. Master's thesis, Dept. of Civil and Environmental Engineering, Massachusetts, 1997.

[38] J. F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science*, 32(4):380–404, November 1998.

[39] F. Corman, A. D'Ariano, I. A. Hansen, D. Pacciarelli, and M. Pranzo. Dispatching trains during seriously disrupted traffic situations. In *2011 International Conference on Networking, Sensing and Control*, pages 323–328, 2011.

[40] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. Evaluation of green wave policy in real-time railway traffic management. *Transportation Research Part C: Emerging Technologies*, 17:607–616, 12 2009.

[41] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. Centralized versus distributed systems to reschedule trains in two dispatching areas. *Public Transport*, 2:219–247, 08 2010.

[42] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B: Methodological*, 44:175–192, 01 2010.

[43] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo. Dispatching and coordination in multi-area railway traffic management. *Computers & Operations Research*, 44:146–160, 2014.

[44] F. Corman, A. D'Ariano, M. Pranzo, and I. Hansen. Effectiveness of dynamic reordering and rerouting of trains in a complicated and densely occupied station area. *Transportation Planning and Technology*, 34, 06 2011.

[45] F. Corman, A. D' Ariano, D. Pacciarelli, and M. Pranzo. Optimal inter-area coordination of train rescheduling decisions. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):71–88, 2012. Select Papers from the 19th International Symposium on Transportation and Traffic Theory.

[46] F. Corman and L. Meng. A review of online dynamic models and algorithms for railway traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1274–1284, 2015.

[47] A. L. Croella, V. Dal Sasso, L. Lamorgese, C. Mannino, and P. Ventura. Disruption management in railway systems by safe place assignment. *Transportation Science*, online, 2022.

[48] V. Dal Sasso, L. Lamorgese, C. Mannino, A. Onofri, and P. Ventura. The tick formulation for deadlock detection and avoidance in railways traffic control. *Journal of Rail Transport Planning & Management*, 17:100239, 2021.

[49] A. D'Ariano. Innovative decision support system for railway traffic control. *Intelligent Transportation Systems Magazine, IEEE*, 1:8 – 16, 02 2009.

[50] A. D'Ariano, F. Corman, D. Pacciarelli, and M. Pranzo. Reordering and local rerouting strategies to manage train traffic in real time. *Transportation Science*, 42:405–419, 11 2008.

[51] A. D'Ariano, D. Pacciarelli, and M. Pranzo. Assessment of flexible timetables in real-time traffic management of a railway bottleneck. *Transportation Research Part C: Emerging Technologies*, 16:232–245, 04 2008.

[52] A. D'Ariano and M. Pranzo. An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. *Networks and Spatial Economics*, 9(1):63–84, 2009.

[53] S. Dash, O. Günlük, A. Lodi, and A. Tramontani. A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(1):132–147, 2012.

[54] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[55] DigiTwin project - Railway Gazette International. Automated wagon inspection project launched, 2021. `https://www.railwaygazette.com/technology/automated-wagon-inspection-project-launched/59167.article` [Online. Last Visited: 29 December, 2021].

[56] DSB - Railway Gazette International. Second ertms pilot line commissioned, 2019. `https://www.railwaygazette.com/infrastructure/second-ertms-pilot-line-commissioned/55385.article` [Online. Last Visited: 29 December, 2021].

[57] M. E. Dyer and L. A. Wolsey. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, 26(2):255–270, 1990.

[58] A. D' Ariano, D. Nucci, D. Pacciarelli, M. Rosti, and A. F. S. B. Italia. Intelligent real-time traffic management system for complex and busy railway networks. In *Proceedings of the 11th world congress on railway research—WCRR*, 01 2016.

[59] A. D' Ariano, D. Pacciarelli, and M. Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, 2007.

[60] European Commission. Moving block and traffic management in railway applications: the EC project COMBINE, 2000. `https://www.witpress.com/Secure/elibrary/papers/CR00/CR00001FU.pdf` [Online. Last Visited: 29 December, 2021].

[61] European Commission - CORDIS EU research results. Algorithms for robust and on-line railway optimisation: Improving the validity and reliability of large-scale systems, 2006-2009. `https://cordis.europa.eu/project/id/021235-2` [Online. Last Visited: 29 December, 2021].

[62] European Commission - CORDIS EU research results. Optimal networks for train integration management across europe, 2011-2014. `https://cordis.europa.eu/project/id/285243/reporting` [Online. Last Visited: 29 December, 2021].

[63] European Railway Agency (ERA). European rail traffic management system (ERTMS), 2021. `https://www.era.europa.eu/activities/european-rail-traffic-management-system-ertms_en` [Online. Last Visited: 29 December, 2021].

[64] W. Fang, S. Yang, and X. Yao. A survey on problem models and solution approaches to rescheduling in railway networks. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2997–3016, 2015.

[65] F. Fischer and C. Helmberg. Dynamic Graph Generation and Dynamic Rolling Horizon Techniques in Large Scale Train Timetabling. In T. Erlebach and M. Lübbecke, editors, *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10)*, volume 14 of *OpenAccess Series in Informatics (OASIcs)*, pages 45–60, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[66] F. Fischer and C. Helmberg. Dynamic graph generation for the shortest path problem in time expanded networks. *Mathematical Programming*, 143(1-2):257–297, 02 2014. Copyright - Springer-Verlag Berlin Heidelberg and Mathematical Optimization Society 2014; Last updated - 2021-09-11; CODEN - MHPGA4.

[67] M. Fischetti and M. Monaci. Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. *European Journal of Operational Research*, 05 2017.

[68] M. Flamini and D. Pacciarelli. Real time management of a metro rail terminus. *European Journal of Operational Research*, 189:746–761, 09 2008.

[69] S. Foglietta, G. Leo, C. Mannino, P. Perticaroli, and M. Piacentini. An optimized, automatic tms in operations in roma tiburtina and monfalcone stations. *WIT Transactions on The Built Environment*, 135:635–647, 2014.

[70] M. R. Garey and D. S. Johnson. *Computers and intractability. A guiged to the theory of NP-completeness.* WH Freman and Company, San Francisco, 1979.

[71] N. Ghaemi, O. Cats, and R. M. Goverde. A microscopic model for optimal train short-turnings during complete blockages. *Transportation Research Part B: Methodological*, 105:423 – 437, 2017.

[72] N. Ghaemi, O. Cats, and R. M. Goverde. Macroscopic multiple-station short-turning model in case of complete railway blockages. *Transportation Research Part C: Emerging Technologies*, 89:113–132, 2018.

[73] N. Ghaemi, O. Cats, and R. M. P. Goverde. Railway disruption management challenges and possible solution directions. *Public Transport*, 9(1):343–364, July 2017.

[74] F. Ghofrani, Q. He, R. M. Goverde, and X. Liu. Recent applications of big data analytics in railway transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, 90:226–246, 2018.

[75] F. Gnegel and A. Fügenschuh. Branch-and-refine for solving time-dependent problems. In *ottbus Mathematical Preprint COMP#13*, 2020.

[76] E. Goddard. Overview of signalling and train control systems. In *The 9th Institution of Engineering and Technology Professional Development Course on Electric Traction Systems*, pages 336–350, 2006.

[77] R. M. Goverde. A delay propagation algorithm for large-scale railway traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(3):269–287, 2010. 11th IFAC Symposium: The Role of Control.

[78] S. Harrod. Modeling network transition constraints with hypergraphs. *Transportation Science*, 45(1):81–97, 2011.

[79] S. Harrod. A tutorial on fundamental model structures for railway timetable optimization. *Surveys in Operations Research and Management Science*, 17(2):85–96, 2012.

[80] S. Harrod and T. Schlechte. A direct comparison of physical block occupancy versus timed block occupancy in train timetabling formulations. *Transportation Research Part E Logistics and Transportation Review*, 54:50 – 66, 08 2013.

[81] E. He, N. Boland, G. Nemhauser, and M. Savelsbergh. *A Dynamic Discretization Discovery Algorithm for the Minimum Duration Time-Dependent Shortest Path Problem*, pages 289–297. Springer International Publishing, Cham, 2018.

[82] Y. He, F. Lehuédé, and O. Péton. A dynamic discretization approach to the integrated service network design and vehicle routing problem. In *VeRoLog 2019 : seventh annual workshop of the EURO Working Group on Vehicle Routing and Logistics Optimization*, Sevilla, Spain, June 2019.

[83] M. Hewitt. Enhanced dynamic discretization discovery for the continuous time load plan design problem. *Transportation Science*, 53(6):1731–1750, 2019.

[84] A. Higgins, E. Kozan, and L. Ferreira. Optimal scheduling of trains on a single line track. *Transportation Research Part B: Methodological*, 30(2):147–161, 1996.

[85] A. J. Higgins, E. Kozan, and L. Ferreira. Heuristic techniques for single line train scheduling. *Journal of Heuristics*, 3:43–62, 1997.

[86] C. Hirai, T. Kunimatsu, N. Tomii, S. Kondou, and M. Takaba. A train stop deployment planning algorithm using a petri-net-based modelling approach. *Quarterly Report of Rtri*, 50:8–13, 01 2009.

[87] IRJ - International Railway Journal. Union pacific: on the rail road to ATO, 2019. `https://www.railjournal.com/in_depth/union-pacific-ato` [Online. Last Visited: 29 December, 2021].

[88] M. Isaai and M. Singh. An intelligent constraint-based search method for a single-line passenger-train scheduling problems. page 79 – 91, 2000.

[89] J. Jespersen-Groth and J. Clausen. Optimal reinsertion of cancelled train line. Technical report, Denmark & Informatics and Mathematical Modelling, The Technical University of Denmark, aug 2006.

[90] J. Jespersen-Groth, D. Potthoff, J. Clausen, D. Huisman, L. G. Kroon, G. Maróti, and M. N. Nielsen. *Disruption Management in Passenger Railway Transportation*, pages 399–421. Springer Berlin Heidelberg, 2009.

[91] S. Josyula and J. Törnquist. Passenger-oriented railway traffic re-scheduling: A review of alternative strategies utilizing passenger flow data. In *7th International Conference on Railway Operations Modelling and Analysis, Lille*, 2017.

[92] S. Josyula, J. Törnquist, and L. Lundberg. An evaluation framework and algorithms for train rescheduling. *Algorithms*, 13, 12 2020.

[93] P. Kecman, F. Corman, A. D'Ariano, and R. Goverde. Rescheduling models for railway traffic management in large-scale networks. *Public Transport*, 5, 03 2013.

[94] A. Keha, K. Khowala, and J. Fowler. Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering*, 56:357–367, 02 2009.

[95] S. Kohli, A. Kumar, J. Easton, and C. Roberts. *Innovative Applications of Big Data in the Railway Industry.* IGI Global, Philadelphia, PA, USA, 2017.

[96] N. Kumar and A. Mishra. *Role of Artificial Intelligence in Railways: An Overview*, pages 323–330. Springer Nature, 02 2021.

[97] F. Lagos, N. Boland, and M. Savelsbergh. The continuous-time inventory-routing problem. *Transportation Science*, 01 2020.

[98] L. Lamorgese and C. Mannino. An exact decomposition approach for the real-time train dispatching problem. *Operations Research*, 63(1):48–64, 2015.

[99] L. Lamorgese and C. Mannino. A noncompact formulation for job-shop scheduling problems in traffic management. *Oper. Res.*, 67:1586–1609, 2019.

[100] L. Lamorgese, C. Mannino, D. Pacciarelli, and J. Törnquist. Train dispatching. In *Handbook of Optimization in the Railway Industry*, pages 265–283. Springer, 2018.

[101] L. Lamorgese, C. Mannino, and M. Piacentini. Optimal train dispatching by benders' like reformulation. *Transportation Science*, 50(3):910–925, 2016.

[102] J. Lenstra and A. Rinnooy Kan. Computational complexity of discrete optimization problems. In P. Hammer, E. Johnson, and B. Korte, editors, *Discrete Optimization I*, volume 4 of *Annals of Discrete Mathematics*, pages 121–140. Elsevier, 1979.

[103] I. Louwerse and D. Huisman. Adjusting a railway timetable in case of partial or complete blockades. *European Journal of Operational Research*, 235, 06 2014.

[104] R. Lusby, J. Larsen, D. Ryan, and M. Ehrgott. Routing trains through railway junctions: A new set-packing approach. *Transportation Science*, 45:228–245, 05 2011.

[105] R. M. Lusby, J. Larsen, M. Ehrgott, and D. M. Ryan. A set packing inspired method for real-time junction train routing. *Computers & Operations Research*, 40(3):713–724, 2013. Transport Scheduling.

[106] C. Mannino. Real-time traffic control in railway systems. In *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, pages 1–14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 01 2011.

[107] C. Mannino, L. C. Lamorgese, and M. Piacentini. Integer programming techniques for train dispatching in mass transit and main line. *Advances and trends in optimization with engineering applications*, 2017.

[108] C. Mannino and A. Mascis. Optimal real-time traffic control in metro stations. *Operations Research*, 57(4):1026–1039, 2009.

[109] C. Mannino and G. Sartor. The path&cycle formulation for the hotspot problem in air traffic management. In *ATMOS*, 2018.

[110] C. Mannino and G. Sartor. An exact (re) optimization framework for real-time traffic management. *optimization on line*, 2020.

[111] L. Marshall, N. Boland, M. Savelsbergh, and M. Hewitt. Interval-based dynamic discretization discovery for solving the continuous-time service network design problem. *Transportation Science*, 55(1):29–51, 2021.

[112] A. Mascis. *Optimization and simulation models applied to railway traffic.* PhD thesis, University of Rome "La Sapienza", Italy, 1997.

[113] A. Mascis and D. Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, 2002.

[114] A. Mascis, D. Pacciarelli, and M. Pranzo. *Scheduling Models for Short-Term Railway Traffic Optimisation*, volume 600, pages 71–90. Springer, 01 2008.

[115] A. Mees. Railway scheduling by network optimization. *Mathematical and Computer Modelling*, 15(1):33–42, 1991.

[116] L. Meng and X. Zhou. Robust single-track train dispatching model under a dynamic and stochastic environment: A scenario-based rolling horizon solution approach. *Transportation Research Part B: Methodological*, 45(7):1080–1102, 2011.

[117] L. Meng and X. Zhou. Simultaneous train rerouting and rescheduling on an n-track network: A model reformulation with network-based cumulative flow variables. *Transportation Research Part B: Methodological*, 67:208–234, 2014.

[118] S. Narayanaswami and N. Rangaraj. Scheduling and rescheduling of railway operations: A review and expository analysis. *Technology Operation Management*, 2, 12 2011.

[119] S. Narayanaswami and N. Rangaraj. Modelling disruptions and resolving conflicts optimally in a railway schedule. *Computers & Industrial Engineering*, 64(1):469–481, 2013.

[120] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

[121] L. Nielsen. *Rolling Stock Rescheduling in Passenger Railways. Applications in short-term planning and in disruption management.* PhD thesis, Erasmus University Rotterdam, Netherlands, 2011.

[122] L. K. Nielsen, L. Kroon, and G. Maróti. A rolling horizon approach for disruption management of railway rolling stock. *European Journal of Operational Research*, 220(2):496–509, 2012.

[123] E. Oliveira and B. M. Smith. A job-shop scheduling model for the single-track railway scheduling problem. Technical report, Problem, research report 2000.21, University of Leeds. Opentrack Railway Technology, Railway Simulation. http://www.opentrack.ch, 2000.

[124] J. Pachl. *Railway operation and control.* Mountlake Terrace, WA : VTD Rail Pub, 01 2002.

[125] P. Pellegrini, G. Marlière, and J. Rodriguez. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Methodological*, 59:58–80, 2014.

[126] E. R. Petersen, A. J. Taylor, and C. Martland. An introduction to computerassisted train dispatch. *Journal of Advanced Transportation*, 20:63–72, 1986.

[127] M. L. Pinedo. *Planning and scheduling in manufacturing and services.* Springer, New York, NY, 2009.

[128] M. L. Pinedo. *Scheduling. Theory, Algorithms, and Systems.* Springer International Publishing, 5 edition, 2016.

[129] A. Pritsker, L. J. Waiters, and P. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16:93–108, 1969.

[130] E. Quaglietta, P. Pellegrini, R. Goverde, T. Albrecht, B. Jaekel, G. Marlière, J. Rodriguez, T. Dollevoet, B. Ambrogio, D. Carcasole, M. Giaroli, and G. Nicholson. The on-time real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. *Transportation Research Part C: Emerging Technologies*, 63:23–50, 02 2016.

[131] M. Queyranne and A. S. Schulz. *Polyhedral approaches to machine scheduling.* Citeseer, 1994.

[132] RAS - The Institute for Operations Research and the Management Sciences. The 2021 railroad problem solving competition, 2021. `https://connect.informs.org/railway-applications/new-item3/problem-solving-competition681` [Online. Last Visited: 29 December, 2021].

[133] E. Reynolds, M. Ehrgott, S. J. Maher, A. Patman, and J. Y. Wang. A multicommodity flow model for rerouting and retiming trains in real-time to reduce reactionary delay in complex station areas. *Optimization Online*, 2020.

[134] M. Riedler, M. Ruthmair, and G. Raidl. *Strategies for Iteratively Refining Layered Graph Models*, pages 46–62. Springer, 01 2019.

[135] J. Rodriguez. Scheduling theory and constraint programming applied to rail traffic management. Technical report, TECHNICAL UNIVERSITY DELFT, 2012.

[136] B. Roy and B. Sussmann. Les problemes d' ordon ordonnancement avec constraints disjunctives. Technical report, 1964.

[137] D. RöSSler, J. Reisch, F. Hauck, and N. Kliewer. Discerning primary and secondary delays in railway networks using explainable ai. *Transportation Research Procedia*, 52:171–178, 2021. 23rd EURO Working Group on Transportation Meeting, EWGT 2020, 16-18 September 2020, Paphos, Cyprus.

[138] G. Sahin, R. Ahuja, and C. B. Cunha. Integer programming based solution approaches for the train dispatching problem. 2010.

[139] M. Samà, A. D'Ariano, and D. Pacciarelli. Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports. *Procedia-Social and Behavioral Sciences*, 80:531–552, 2013.

[140] Y. O. Scherr, M. Hewitt, B. A. N. Saavedra, and D. C. Mattfeld. Dynamic discretization discovery for the service network design problem with mixed autonomous fleets. *Transportation Research Part B: Methodological*, 141:164–195, 2020.

[141] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency.* Springer, 2003.

[142] S. Shen and N. H. Wilson. *An Optimal Integrated Real-time Disruption Control Model for Rail Transit Systems*, pages 335–363. Springer Berlin Heidelberg, 2001.

[143] J. P. Sousa and L. A. Wolsey. A time indexed formulation of non-preemptive single machine scheduling problems. LIDAM Reprints CORE 984, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 1992.

[144] South Western Railway (SWR). Machine learning deployed to support train operations decision-making, 2021. `https://www.railwaygazette.com/uk/machine-learning-deployed-to-support-train-operations-decision-making/59691.article` [Online. Last Visited: 29 December, 2021].

[145] B. Szpigel. Optimal train scheduling on a single line railway. *Operations Research '72*, pages 343–352, 1973.

[146] A. Tesch. A polyhedral study of event-based models for the resource-constrained project scheduling problem. *Journal of Scheduling*, 23, 04 2020.

[147] A. Toletti and U. Weidmann. Modelling customer inconvenience in train rescheduling. In *Swiss Transport Research Conference (STRC2016)*, 05 2016.

[148] J. Törnquist. Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. *OpenAccess Series in Informatics*, 2, 01 2005.

[149] S. Tschirner, B. Sandblad, and A. W. Andersson. Solutions to the problem of inconsistent plans in railway traffic operation. *Journal of Rail Transport Planning & Management*, 4(4):87–97, 2014.

[150] UIC - Union internationale des chemins de fer. Uic leaflet 406, 2004. `https://tamannaei.iut.ac.ir/sites/tamannaei.iut.ac.ir/files/files\char'_course/uic406\char'_2013.pdf` [Online. Last Visited: 29 December, 2021].

[151] University of Hong Kong. Safety, reliability, and disruption management of high speed rail and metro systems, 2016. `http://www.cityu.edu.hk/csie/TBRS/` [Online. Last Visited: 29 December, 2021].

[152] S. Van Aken, N. Beinovi, and R. Goverde. Designing alternative railway timetables under infrastructure maintenance possessions. *Transportation Research Part B: Methodological*, 98:224–238, 04 2017.

[153] S. Van Aken, N. Beinovi, and R. Goverde. Solving large-scale train timetable adjustment problems under infrastructure maintenance possessions. *Journal of Rail Transport Planning & Management*, 7, 07 2017.

[154] L. Veelenturf, M. Kidd, V. Cacchiani, L. Kroon, and P. Toth. A railway timetable rescheduling approach for handling large scale disruptions. *Transportation Science*, 50:841–862, 01 2016.

[155] D. M. Vu, M. Hewitt, N. Boland, and M. Savelsbergh. Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. *Transportation Science*, 54(3):703–720, 2020.

[156] R. Vujanic and A. Hill. Computationally efficient dynamic traffic optimization of railway systems. *ArXiv*, abs/2105.03924, 2021.

[157] X. Wang and A. C. Regan. Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological*, 36(2):97–112, 2002.

[158] X. Wang and A. C. Regan. On the convergence of a new time window discretization method for the traveling salesman problem with time window constraints. *Computers & Industrial Engineering*, 56(1):161–164, 2009.

[159] C. Wen, P. Huang, Z. Li, J. Lessan, L. Fu, C. Jiang, and X. Xu. Train dispatching management with data-driven approaches: a comprehensive review and appraisal. *IEEE Access*, 7:114547–114571, 2019.

[160] S. Zhan, L. Kroon, L. Veelenturf, and J. Wagenaar. Real-time high-speed train rescheduling in case of a complete blockage. *Transportation Research Part B: Methodological*, 78, 08 2015.

[161] S. Zhan, L. G. Kroon, J. Zhao, and Q. Peng. A rolling horizon approach to the high speed train rescheduling problem in case of a partial segment blockage. *Transportation Research Part E: Logistics and Transportation Review*, 95:32–61, 2016.

[162] S. Zhan, S. C. Wong, Q. Peng, and S. M. Lo. Train stop deployment planning in the case of complete blockage: An integer linear programing model. *Journal of Transportation Safety & Security*, 0(0):1–27, 2019.

[163] Y. Zhou, J. Wang, and H. Yang. Resilience of transportation systems: Concepts and comprehensive review. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4262–4276, 2019.

[164] Y. Zhu and R. Goverde. Railway timetable rescheduling with flexible stopping and flexible short-turning during disruptions. *Transportation Research Part B Methodological*, 123:149–181, 04 2019.

[165] Y. Zhu and R. Goverde. Dynamic and robust timetable rescheduling for uncertain railway disruptions. *Journal of Rail Transport Planning & Management*, 04 2020.

[166] Y. Zhu and R. M. Goverde. Dynamic railway timetable rescheduling for multiple connected disruptions. *Transportation Research Part C: Emerging Technologies*, 125:103080, 2021.

[167] A. A. Zilko, D. Kurowicka, and R. Goverde. Modeling railway disruption lengths with copula bayesian networks. *Transportation Research Part C-emerging Technologies*, 68:350–368, 2016.

Ringraziamenti

Lao Tzu scrisse "I nuovi inizi sono spesso mascherati da finali dolorosi". O più verosilmente "新的開始往往被痛苦的結局所掩蓋。". In ogni caso sono d'accordo con lui.
Era ottobre 2018 e dopo quasi un anno di lavoro in consulenza, senza saperlo, non riconoscevo più me stessa. Non voglio dire che è stato doloroso, straziante e avvilente, perché a dir la verità non me la cavavo neanche male. Voglio dire che quel periodo per me è stato snaturante e sono riuscita a capirlo solo dopo aver cominciato questo percorso di crescita, accademica e personale, durato 3 anni (+ proroga). Ora credo di essere nel posto in cui dovrei stare e vorrei ringraziare tutte le persone che mi ci hanno guidato.

Per questo vorrei dedicare qualche riga ai miei colleghi, o meglio amici, con i quali ho trascorso la maggiorparte delle mie giornate in dipartimento, al cinema, alle *stand up comedy* o nelle *escape room*, da Mario, da Chamei o dai cinesi. Dal primo giorno al DIAG ho capito di essere entrata in una nuova squadra, e che A114 e A115 erano solo un enorme spogliatoio travestito da Aula Dottorandi. Tra i miei colleghi c'è chi mi ha sostenuto e spronato prima che io imparassi a farlo da me. Non smetterò mai di ringraziare Tommaso Giovannelli per quel mercoledì sera in cui è riuscito a convincermi a presentarmi all'esame orale di ammissione. Un grazie speciale va a Edoardo, un completo sconosciuto che in pochi mesi è diventato come un fratello. A volte sembra il maggiore, quando mi protegge e cerca di mettermi in guardia dalle situazioni più indecifrabili, altre il minore, quando ha bisogno di essere guidato e calmato prima che prenda una delle sue decisioni dell'ultimo minuto, convinta e completamente insensata. Lo ringrazio per tutto ciò che ha condiviso con me e con gli altri, per aver creato sempre un clima di profonda amicizia e mai di competizione. Ringrazio Marco, per le mille curiosità e la cultura diffusa con ogni suo discorso, per i sondaggi e per le domande pungenti sui sentimenti che neanche lui sa di saper fare. Ringrazio Giorgio per aver semplicemente deciso di mostrarsi per la persona splendida qual è. Ci è voluto un mese a farlo sciogliere durante il freddo inverno norvegese, ma sono felice che alla fine abbia deciso di aprire il suo enorme cuore. Ringrazio Davide, per le notti da ventisettenni suonati in stile *The Sleepover Club* e per il brio portato in dipartimento. Ringrazio Martina, per essersi prestata a condividere i più disparati interessi, ma anche per tutti i racconti delle improbabii situazioni in cui inciampa ogni giorno. Ringrazio Giulia, per il suo spontaneo e disinteressato supporto, per i suoi piani, dalla A alla Z, e per i suoi incoraggiamenti. Grazie a Riccardo per gli aperitivi e le incursioni serali in A114, per i mille aneddotti divertenti e per gli sfottò, ma anche per la sua capacità di ascolto. Ringrazio Chiara per avermi fatto sentire da subito inglobata e accolta in una nuova realtà, per le mille domande curiose, e facilemente risolvibili, che sente la necessità di chiedere. Ringrazio Mirko per aver condiviso la sua passione per la fotografia culinaria, per la sua rallegrante e sempre serena presenza. Grazie a Luca per avermi perdonato dopo tre anni per il famoso lancio delle pietre. Ancora grazie a Ruggiero, Marianna, Tiziana, Alessandro, Annarelli, Alice, Marta, Valerio, Andrea, per i pranzi, le risate, le chiacchiere e per rendere un ambiente di lavoro una seconda casa.

Ringrazio i miei tutor, il Prof. Carlo Mannino e il Prof. Paolo Ventura, che mi hanno espressamente richiesto dei ringraziamenti sarcastici. Grazie per i loro consigli, per le mille videochiamate su meet ma soprattutto per la condivisione della loro conoscenza e cultura in merito a *fun facts*. Grazie perchè ora so la vera storia di alcune espressioni come "spezzare una lancia" o "essere sano come un pesce". Lasciando da parte l'umorismo, che ci accomuna, vorrei ringraziare Carlo per la sua fondamentale guida e le sue sempre calzanti ed efficaci direttive. Grazie a Paolo per aver messo sempre a disposizione la sua esperienza e presenza, e per la sua disponibilità al confronto.
Vorrei aggiungere un grande ringraziamento alla Prof.ssa Laura Palagi, la mia tutor informale, che ha sempre creduto nelle mie potenzialità e mi ha dato la possibilità di dimostrarle più volte.
Un ringraziamento va anche al DIAG in quanto istituzione, per aver finanziato le sempre magnifiche e mai ordinarie missioni, in Italia e all'estero. Grazie per Ischia, Firenze, Lisbona, Bolzano, Oslo e Atene.

Vorrei inoltre ringraziare chi c'è sempre stato, le mie mine vaganti che (r)esistonoo da sempre. Un grazie a Valentina, sono felice di aver percorso anche questa strada al suo fianco, spronandoci a vicenda. Siamo cresciute insieme in questi anni, forse più di quanto vogliamo ammettere (ad oggi lei riesce addirittura a mandarmi dei cuori su Whatsapp), e non posso che esserle grata di questo. Grazie a Maria, per la sua sensibilità e la sua ben nascosta gentilezza, un grazie per tutte le volte in cui mi ha stupito con le sue parole, in cui non è mancata dimostrando in ogni situazione la sua sincera amicizia. Grazie a Federica e Bea per la loro generosità, ospitalità e abilità nel far sentire chiunque parte di un'unica famiglia di pari. Ringrazio ancora una volta Bea e Assenzio per tutte le uscite delle ore 19, per gli sfoghi e le risate sulle panchine di piazza Massa Carrara. Grazie a Marco, per la sua empatia sentita, la sua immancabile presenza e i suoi consigli. Grazie a Valeria (e alla sua splendida famiglia) per il loro dolce sostegno. Un grazie a Minni e Billi per le spumeggianti feste in terrazza e la loro, ormai proverbiale, spensieratezza. Grazie a Peppe, Lorenzo, Francesca e Marco per le serate in pista e per tutte quelle battute apparentemente stupide che hanno il potere di farti sentire, strappandoti un sorriso, finalmente giusta e nel posto giusto.

Ringrazio Marta, per tutti quei momenti in cui siamo esistite soltanto noi, nel bel mezzo di una pandemia a migliaia di km di distanza o senza un centimetro di spazio tra di noi.

Grazie alla mia famiglia, a mia madre e mio padre che hanno reso possibile tutto questo. Ringrazio mia sorella per essere capace di dimostrarmi sempre il suo afetto, nonostante io continui a "fare l'orso" con lei. Sono grata a tutti loro per dimostrarmi senza l'ausilio di parole l'insegnameento più importante, quello di essere sempre più grandi delle nostre sofferenze.

Ci sono altre persone che vorrei ringraziare, che sono passate nella mia vita e che nonostante siano fisicamente distanti riescono comunque a essere presenti. Grazie a Lucie, Alessia, Giulia, Vincenzo, Noemi, Gianmarco, Davide e Antonella.
Per la terza volta in una tesi ringrazio la mia squadra del cuore rosso-blu, che rimane come una melodia di sottofondo, pronta a incoraggiarti e a farti sapere che ti sta guardando. Ringrazio la mia attuale squadra, il mio mister e le mie compagne, che da settembre fanno parte delle mie settimane e con le quali condivido emozioni sempre più forti.

So di avere ancora molto da impare e sono felice di questo. Grazie ancora una volta a tutti. Mi piace immaginarvi tutti insieme, in stile Harry Potter e i doni della morte - parte II, nella scena in cui Harry ha in mano la pietra della resurrezzione. Per i fan questa immagine vale più di tutto quello che ho appena detto. Per chi non sa di cosa sto parlando, punto primo sono veramente delusa!, punto secondo il succo è che vi ringrazio per avermi fatto sentire amata e per essere stati al mio fianco. Ci sarà sempre un pezzo di ognuno di voi in tutto quello che farò. Vi voglio bene.

La vostra AnnaLove di dipartimento
Roma, Dicembre 2021

Train scheduling is a critical activity in rail traffic management, both off-line (timetabling) and on-line/in real-time (dispatching). This research aims at the design and development of advanced optimization models and methods for the real-time re-scheduling problem. The objective is twofold: managing the train service plans in an increasingly dynamic scenario; and ensuring the safety and reliability of the railway system in case of major disruptions.

The projects outputs are very innovative and off the beaten track.